Preamble to Video Services Forum (VSF) Technical Recommendation TR-06-3:2022

September 8, 2022

The Reliable Internet Stream Transport (RIST) project was initiated as an Activity Group under the auspices of the Video Services Forum in 2017. To date, the group has produced three specifications, released as TR-06-1 (RIST Simple Profile, published in 2018 and updated in 2020), TR-06-2 (RIST Main Profile, published in 2020 and updated in 2021 and 2022), and TR-03-3 (RIST Advanced Profile, published in 2021).

The attached document is a minor update to the RIST Advanced Profile Specification. It corrects some minor editorial issues, and includes a reference to the EAP SHA256-SRP6a Authentication Protocol, which is now documented in Annex D of TR-06-2:2022. Additionally, the encoding details of the Payload Format Descriptor moved from TR-06-3 to the Registration Repository on GitHub since these are subject to change.

The primary purpose of RIST Advanced Profile is to create a specification for a generic tunnel with packet loss recovery, similar to what is defined in RIST Simple Profile. This way, any non-RIST protocol can benefit from RIST packet recovery features. RIST Advanced Profile also includes support for optional lossless data compression, fragmentation (with fragment-level packet recovery), as well as additional ciphers and data integrity options in Pre-Shared Key (PSK) mode.

RIST Advanced Profile also includes a very flexible Payload Format Descriptor. This can be combined with a Direct Payload mode to offer low-overhead and low-latency transport modes. The Payload Format Descriptor registration is hosted on the following GitHub repository:

https://github.com/vsf-tv/rist-pfd

The repository includes a list of registered format descriptors and instructions on how to submit new entries, as well as the encoding rules.

Work continues within the group towards developing additional RIST specifications that include additional features. As the Activity Group develops and reaches consensus on new functions and capabilities, these documents will also be released in support of the RIST effort. For additional information about the RIST Activity group, or to find out about participating in the development of future specifications, please visit http://vsf.tv/RIST.shtml

Video Services Forum (VSF)
Technical Recommendation TR-06-3

Reliable Internet Stream Transport (RIST) Protocol Specification – Advanced Profile

Approved September 8, 2022

INTELLECTUAL PROPERTY RIGHTS

THIS RECOMMENDATION IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NONINFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS RECOMMENDATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY MPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS RECOMMENDATION.

LIMITATION OF LIABILITY

VSF SHALL NOT BE LIABLE FOR ANY AND ALL DAMAGES, DIRECT OR INDIRECT, ARISING FROM OR RELATING TO ANY USE OF THE CONTENTS CONTAINED HEREIN, INCLUDING WITHOUT LIMITATION ANY AND ALL INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS, LOSS OF PROFITS, LITIGATION, OR THE LIKE), WHETHER BASED UPON BREACH OF CONTRACT, BREACH OF WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE FOREGOING NEGATION OF DAMAGES IS A FUNDAMENTAL ELEMENT OF THE USE OF THE CONTENTS HEREOF, AND THESE CONTENTS WOULD NOT BE PUBLISHED BY VSF WITHOUT SUCH LIMITATIONS.

Executive Summary

This Technical Recommendation defines a base protocol that can be used both for low-latency media delivery, and as a delivery mechanism for existing legacy protocols. This base protocol includes features such as packet loss recovery, fragmentation support, lossless data compression, content identification, security and authentication. This document is part of a series of open and collaboratively developed protocol specifications that provide reliable streaming over the Internet.

Many proprietary solutions for streaming video and audio exist in the marketplace. These solutions all use the same types of techniques, but with the exception of implementations based upon this series of Technical Recommendations, they do not interoperate with each other.

Recipients of this document are invited to submit technical comments. The VSF also requests that recipients notify us of any relevant patent claims or other intellectual property rights of which they may be aware, that might be infringed by any implementation of the Recommendation set forth in this document, and to provide supporting documentation.

Table of Contents

1 Introduction (Informative)

As broadcasters and others increasingly utilize unconditioned Internet circuits to transport high-quality video, the demand grows for systems that can compensate for the packet losses and delay variation that often affect these streams. A variety of solutions are currently available on the market; however, incompatibilities exist between devices from different suppliers.

The Reliable Internet Stream Transport (RIST) project was launched specifically to address the lack of compatibility between devices, and to define a set of interoperability points through the use of existing or new standards and recommendations.

This Specification defines a method and protocol that provides low-latency reliable transport for both media and existing legacy protocols. It also includes support for fragmentation (with fragment-level packet recovery), lossless data compression, payload identification, authentication, and security.

1.1 Contributors

The following individuals participated in the Video Services Forum RIST working group that developed this technical recommendation.

Merrick Ackermans (CBS)	Sergio Ammirata (SipRadius)	Paul Atwell (Media Transport Solutions)
John Beer (QVidium)	Alex Converse (AWS/Twitch)	Eric Fankhauser (Evertz)
Ronald Fellman (QVidium)	Michael Firth (Nevion)	Oded Gants (Zixi)
Chris Homer (NET)	Ciro Noronha (Cobalt Digital)	Hermann Popp (Arri)
Adi Rozenberg (VideoFlow)	Wes Simpson (LearnIPVideo)	Mikael Wånggren (Net Insight)

This technical recommendation builds upon VSF TR-06-1 and VSF TR-06-2. The list of contributors to these documents can be found in section 1.1 of each document.

1.2 About the Video Services Forum

The Video Services Forum, Inc. (www.videoservicesforum.org) is an international association dedicated to video transport technologies, interoperability, quality metrics and education. The VSF is composed of service providers, users and manufacturers. The organization's activities include:

- providing forums to identify issues involving the development, engineering, installation, testing and maintenance of audio and video services;
- exchanging non-proprietary information to promote the development of video transport service technology and to foster resolution of issues common to the video services industry;

- identification of video services applications and educational services utilizing video transport services;
- promoting interoperability and encouraging technical standards for national and international standards bodies.

The VSF is an association incorporated under the Not For Profit Corporation Law of the State of New York. Membership is open to businesses, public sector organizations and individuals worldwide. For more information on the Video Services Forum or this document, please call +1 929-279-1995 or e-mail opsmgr@videoservicesforum.org.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except the Introduction and any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; Tables shall be next; followed by formal languages; then figures; and then any other language forms.

3 References

VSF TR-06-1:2020, Reliable Internet Stream Transport (RIST) Protocol Specification – Simple Profile

VSF TR-06-2:2022, Reliable Internet Stream Transport (RIST) Protocol Specification – Main Profile

IETF RFC 2784, Generic Routing Encapsulation (GRE)

IETF RFC 3173, IP Payload Compression Protocol (IPComp)

IETF RFC 3550, RTP: A Transport Protocol for Real-Time Applications

IETF RFC 4566, SDP: Session Description Protocol

IETF RFC 5054, Using the Secure Remote Password (SRP) Protocol for TLS Authentication

IETF RFC 6347, Datagram Transport Layer Security Version 1.2

IETF RFC 8018, PKCS #5: Password-Based Cryptography Specification Version 2.1

IETF RFC 8259, The JavaScript Object Notation (JSON) Data Interchange Format

SMPTE ST 2022-1:2007, Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks

SMPTE ST 2022-5:2013, Forward Error Correction for Transport of High Bit Rate Media Signals over IP Networks (HBRMT)

J. Carlson, B. Aboba, and H. Haverinen, **EAP SRP-SHA1 Authentication Protocol**, retrieved from https://tools.ietf.org/html/draft-ietf-pppext-eap-srp-03, retrieved August 31, 2021

T. Wu, **SRP Protocol Design**, retrieved from http://srp.stanford.edu/design.html, retrieved August 31, 2021

Any mention of references throughout the rest of this document refers to the versions described here, unless explicitly stated otherwise.

4 RIST Profiles (Informative)

RIST has multiple operational profiles, corresponding to increasing levels of complexity and functionality. Higher profiles include all the features and functionality of the lower profiles. This document defines RIST Advanced Profile, which adds the following features to VSF TR-06-2 RIST Main Profile:

- Tunnel Level Enhancements, including:
    1. Packet recovery
    2. Transparent fragmentation support, with fragment-level recovery
    3. Lower-overhead transport options
    4. Pre-Shared Key operation enhancements, including packet integrity hashes
    5. Support for optional lossless packet payload data compression

A RIST profile roadmap is included in TR-06-1.

5 Tunnel Level Enhancements

The primary objective of the Tunnel Level Enhancements in RIST Advanced Profile is to create a specification for a generic tunnel with packet loss recovery, similar to what is defined in RIST Simple Profile. This way, any non-RIST protocol can benefit from RIST packet recovery features. The tunnel also provides optional features such as fragmentation and lossless packet payload compression. The tunnel definition is compatible with SMPTE ST 2022-1 and ST 2022-5 FEC, which may be combined with ARQ for packet recovery.

5.1 Tunnel Level Features (Informative)

This Specification includes support for the following tunnel-level features:

- Three modes of operation:
    1. Operation in the clear, with no encryption and no authentication.
    2. Pre-Shared Key (PSK) operation, with multiple encryption and hashing options.
    3. DTLS operation, with the same set of cipher suites as TR-06-2.
- Tunnel-Level packet recovery, using both ARQ and FEC methods. This allows the tunnel to transparently provide low-latency packet recovery and reordering to any protocol being transported.
    1. ARQ is provided using the same methods as TR-06-1 RIST Simple Profile, with Bitmask and Range NACKs. These messages are defined in sections 5.3.2 and 5.3.3.
    2. ST 2022-1 and ST 2022-5 FEC can be added to the tunnel in two ways:
        - Since the tunnel top-level uses RTP, it is possible to add FEC exactly as described in ST 2022-1 or ST 2022-5, using two additional UDP ports. However, if using DTLS, this will require additional sessions for the FEC packets.
        - Section 5.3.5 contains a method of carrying ST 2022-1 and ST 2022-5 FEC packets as tunnel control messages. Using this method allows a single UDP port for the tunnel, and a single DTLS session.
- Support for multiplexing data and control packets in the same tunnel, using a single UDP port. This simplifies IT operations by combining all flows, NACKs, FEC packets,

and control into a single port. This feature is described in section 5.2.1, and the control packets are described in section 5.3.

- Support for JSON-based media description, described in section 5.4. This feature is similar to SDP, allowing media details and human-readable labels to be transmitted in the tunnel and associated with specific data flows.
- Support for payload identification codes, to allow for fast packet content classification. This feature is provided by the Payload Format Descriptor field, described in section 5.2.7.
- Support for logical flow grouping, which allows fast identification of related flows. This feature is provided by the Flow ID field, described in section 5.2.4. The Flow ID field allows gateway devices to de-multiplex and multiplex individual related groups of flows (e.g., audio and video) without the need for deeper packet inspection.
- Support for tunnel-level fragmentation and reassembly, allowing the tunnel to transparently transport packets of arbitrary sizes. Since fragmentation happens after lost packet recovery and re-ordering, it is implemented by simply using two flags, described in section 5.2.3. A fragmentation example is presented in Appendix B.
- Support for tunnel-level lossless payload compression, as described in section 5.2.6.
- Support for tunnel-level multi-link operation, through the use of a standard RTP header. The tunnel is compatible with ST 2022-7 seamless switching, as well as bonding.
- PSK mode enhancements over TR-06-2 RIST Main Profile:
    1. Additional cipher support.
    2. Data authentication support, to protect against data corruption and replay attacks.

  These are encoded in the **PSK** flags defined in section 5.2.3.

5.2 Top-Level Tunnel Packet Format

The tunnel senders shall use RTP packets, with the extensions shown in Figure 1 and described in this section.

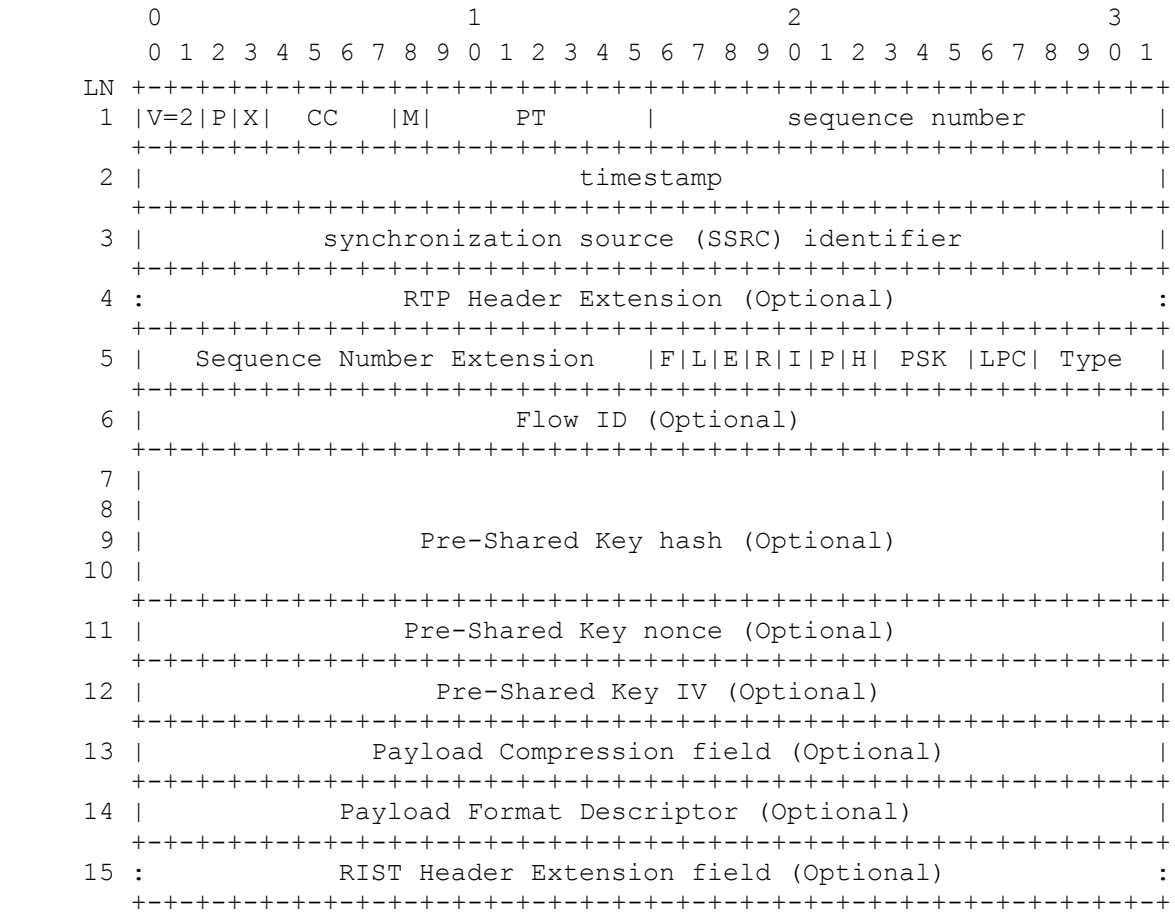


Figure 1: RTP Header for Tunnel Packets

The header is divided into three groups of fields:

1. Lines 1 through 4 in Figure 1 are a standard RTP header. This header may include an extension immediately after the SSRC field, as per RFC 3550 (line 4). The RTP header defined here in Figure 1 constitutes a new RTP profile, with its own timestamp frequency defined in Section 5.2.1.
2. Lines 5 through 15 in Figure 1 are a profile-defined RTP header, compliant with RFC 3550. This profile-defined RTP header is further divided into the following sections:
    a. The top 16 bits of line 5 (MSB in network byte order) are a sequence number extension compatible with the ST-2110 over WAN definitions.
    b. The lower 16 bits of line 5 (LSB in network byte order) are a set of flags that provide further functionality. They are described later in this section.
    c. Lines 6 through 15 are optional headers that may or may not be present depending on the flags in the lower 16 bits (LSB) of line 5.

5.2.1 Standard RTP Header Fields

The standard RTP header fields in lines 1 through 4 of Figure 1 shall be set by senders as follows:

- **Version (V):** 2 bits
  RTP version, set to '2'.
- **Padding (P):** 1 bit
  Set according to RFC 3550.
- **Header Extension (X):** 1 bit
  If this bit is set, a header extension is added as indicated in Figure 1. The format of the header extension is defined in RFC 3550 section 5.3.1, and reproduced in Figure 2.

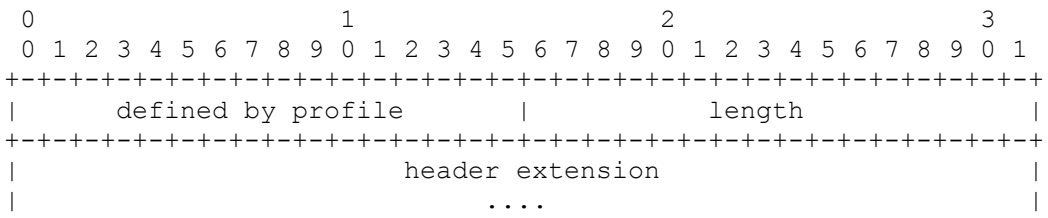


Figure 2: Header Extension Format

- **CSRC Count (CC):** 4 bits
  Set to zero, as the tunnel will not have a CSRC count.
- **Marker (M):** 1 bit
  Set to zero, as the tunnel will not use the marker bit.
- **Payload Type (PT):** 7 bits
  RTP dynamic payload types are usually signaled through SDP. This specification does not require the use of SDP. If SDP is not in use, the fixed Payload Type of 127 shall be used. If SDP is in use, the Payload Type shall be signaled by the SDP message.
- **Sequence Number:** 16 bits
  The sequence number shall increment by one for each RTP data packet sent.
- **Timestamp:** 32 bits
  The timestamp is associated with the first bit of the RTP packet. The initial value of the timestamp should be random, as indicated in RFC 3550. If SDP is not in use, senders and receivers shall use the default timestamp frequency of 1 MHz. If SDP is in use, senders should use this timestamp frequency. If a different frequency is used, that frequency shall be specified in the SDP. If the SDP data specifies a timestamp frequency, receivers shall use this specified frequency. If the SDP data does not specify a timestamp frequency, receivers shall use the default frequency of 1 MHz.
- **SSRC:** 32 bits
  This identifies the source and may be used to multiplex multiple tunnels in the same port.

As indicated in RFC 3550, the SSRC value should be chosen randomly, with the restriction defined below regarding the last bit of the SSRC.

A RIST Advanced Profile flow shall be composed of two consecutive SSRC values, an even SSRC (last bit set to zero) and an odd SSRC (last bit set to '1'), with the remaining 31 bits being the same. The usage of these two SSRCs is as follows:

- Packets with the last bit of the SSRC set to zero shall be denoted as the Protected packets. Recovery mechanisms such as FEC and ARQ may be applied to these packets.
- Packets with the last bit of the SSRC set to one shall be denoted as the Unprotected packets. No recovery mechanism shall be applied to these packets.

Multiple RIST Advanced Profile flows may be combined into a single UDP port using SSRC multiplexing. Support for this mode of operation is optional. A more detailed description can be found in Section 6.

As determined by RFC 3550, the sequence numbers for the Protected and Unprotected packets shall be independent.

5.2.2 Sequence Number Extension

The 16-bit Sequence Number Extension field shall be combined with the RTP sequence number to form a 32-bit sequence number. The sequence number extension shall correspond to the most significant bits of this 32-bit sequence number, as shown in Figure 3.

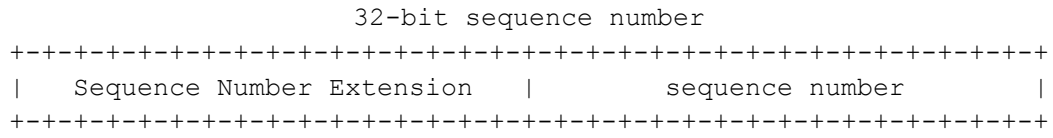


Figure 3: Forming the 32-bit sequence number

5.2.3 Flags

The flags in Figure 1 shall be defined as follows:

- **F (bit 16):** Senders shall set this bit to indicate that the packet is the first fragment of a packet.
- **L (bit 17):** Senders shall set this bit to indicate that the packet is the last fragment of a packet.
  Note: single-fragment packet will have both the **F** and **L** bits set.
- **E (bit 18): E**xpedite bit. Senders shall set this bit to request expedited processing of the packet at the receiver. Receivers shall immediately output or process packets with this bit set.

- **R (bit 19): R**e-transmit bit. Senders shall set this bit to indicate that the packet is a re-transmitted packet. Retransmitted packets shall only be sent as Protected packets, with the last bit of the SSRC set to zero as indicated in Section 5.2.1.
- **I (bit 20):** Flow **I**D present bit. Senders shall set this bit to indicate that the packet header includes the Flow ID field. Usage of the Flow ID field is defined in section 5.2.4.
- **P (bit 21): P**ayload Format Descriptor present bit. Senders shall set this bit to indicate that the packet header includes Payload Format Descriptor field. Usage of the Payload Format Descriptor is defined in section 5.2.7.
- **H (bit 22):** RIST **H**eader Extension present bit. Senders shall set this bit to indicate that the packet header includes the RIST Header Extension field. This this field has the same format as a standard RTP header extension, as per RFC 3550 and illustrated in Figure 2.
- **PSK** field, **bits 23-25**: Senders shall set these bits to indicate Pre-Shared Key operation, as follows:
    - 0 (000): No Encryption
    - 1 (001): AES-CTR (Same as RIST Main Profile)
    - 2 (010): HMAC-SHA256 (no encryption)
    - 3 (011): AES-CTR-HMAC-SHA256
    - 4 (100): AES-GCM
    - 5 (101): CHACHA20-POLY1305
    - 6 (110): User defined/future expansion, no Hash
    - 7 (111): User defined/future expansion, with Hash header

  The presence of the Hash, Nonce and IV fields in the header shall be determined from the value of the **PSK** field as indicated in Table 1.

Table 1: Presence of Hash, Nonce and IV based on PSK field

PSK Field	Hash	Nonce	IV
0			
1		✓	✓
2	✓	✓	
3	✓	✓	✓
4	✓	✓	✓
5	✓	✓	✓
6		✓	✓
7	✓	✓	✓

- **LPC** (Lossless Payload Compression) field, **bits 26-27**: Senders shall set these bits to indicate the lossless payload compression mode in use, as follows:
    - 0 (00): No Compression
    - 1 (01): LZ4 Compression
    - 2 (10): Reserved for future use

    - 3 (11): Indicates that the Payload Compression field is present in the header, and this function is controlled by the value of that header. The Payload Compression field is described in section 5.2.6.
- **Type** field, **bits 28-31**: Senders shall set these bits to describe the format of the payload, as follows:
    - 0 (0000): Reserved.
    - 1 (0001): IPv4 packet
    - 2 (0010): IPv6 packet
    - 3 (0011): Reduced header UDP packet, as defined in TR-06-2.
    - 4 (0100): Control packet. The **E**xpedite bit shall be set for all Control packets. The format for control packets is described in section 5.3.
    - 5 (0101): Direct Payload (without IP and UDP headers). The **P**ayload Format Descriptor field shall be included for all Direct Payload packets to indicate their contents.
    - 6 (0110): Layer 2 Ethernet Frame. The frame shall not include the 4-byte FCS at the tail.
    - 7 (0111): GRE packet, as per RFC 2784. No assumptions are made about the contents of such packets.
    - 8 (1000): RIST Main Profile GRE packets, with the GRE header defined in TR-06-2.
    - 9-15 (1001-1111): Reserved for future use.

Senders shall not compress Unprotected packets. For such packets, senders shall set the LPC field to '00'.

If lossless payload compression and fragmentation are both in use, senders shall apply compression first, followed by fragmentation of the resulting compressed payload as needed. Senders shall set the LPC field of all the fragments of a packet to the same value.

Lossless payload compression shall not be used by a sender unless it receives a keep-alive message from the remote end with the **C** bit set as per section 5.3.6, indicating that the remote end is capable of compression.

Senders may add either an RTP Header Extension or a RIST Header Extension or both to a packet, depending on the application requirements.

An example of each type of encapsulation is presented in Appendix A.

5.2.4 Flow ID Field

The Flow ID field (4 bytes) can be used to multiplex multiple flows into the same tunnel when another mechanism is not available (e.g., for Direct Payload packets). Flow IDs can also be used

as an aid to routing and to apply priorities to the tunnel. The layout of the Flow ID field is shown in Figure 4.

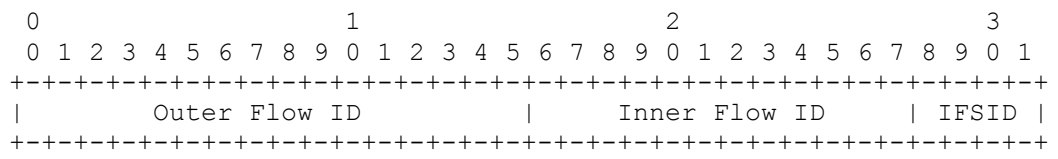


Figure 4: Flow ID Field

Senders shall set the fields in Figure 4 as follows:

- **Outer Flow ID:** 16 bits
  Senders shall use this field to differentiate multiple tunnels in the same SSRC flow. The Outer Flow ID is used to combine multiple logical tunnels that share a single packet loss recovery mechanism and bandwidth management. Alternatively, multiple tunnels may be combined using SSRC multiplexing, as described in section 6. If not used, Outer Flow ID shall be set to zero.
- **Inner Flow ID:** 12 bits
  Senders shall use this field to group multiple flows that logically belong together. In this case, the Inner Flow ID of every flow in the group shall be set to the same value.
- **Inner Flow Sub ID:** 4 bits
  Senders shall use this field to differentiate between individual flows that have been grouped together using the Inner Flow ID. For such flows, senders shall set the same Inner Flow ID and different Inner Flow Sub IDs.

If the Flow ID field shown in Figure 4 is not present in a packet, a receiver shall assume that the Outer Flow ID, the Inner Flow ID and the Inner Flow Sub ID are set to zero.

Example**:** consider a ST 2110 session composed of one video service, two audio services, and one ancillary data service, using Direct Payload Encapsulation (Type field in section 5.2.3 set to 5). All flows have the same Inner Flow ID value. Each flow has a different Inner Flow Sub ID to allow the packets to be properly demultiplexed. It is possible to distinguish the video and ancillary data services from each other using the Payload Format Descriptor, but it is likely that the two audio services will have the same value for this field, and so cannot be demultiplexed correctly without using the Inner Flow Sub ID.

5.2.5 Pre-Shared Key Fields

The fields below are used in Pre-Shared Key operation, and their presence depends on the state of the **PSK** field, as described in Table 1. If PSK is not being used, senders shall set the PSK field to ‘000’ and shall not include the Pre-Shared Key fields in the packet header. The specific usage of the fields by senders shall be:

- **Pre-Shared Key Hash:** (16 bytes)
  Senders shall use this field to convey a hash to ensure packet integrity. For methods that create hashes that are longer than 16 bytes, senders shall set this field to the most significant 16 bytes of the hash.
- **Pre-Shared Key Nonce:** (4 bytes)
  Senders shall use this field to convey the Nonce for the key generation.
- **Pre-Shared Key IV:** (4 bytes)
  Sender shall use this field to convey the IV for PSK operation. It shall be incremented at every packet.

The method for computing the values of these fields is described in Section 8.

5.2.6 Payload Compression Field

The 32-bit Payload Compression Field shall be included by senders if the **LPC** flags are set to 11. This field shall follow the definitions from RFC 3173, as shown in Figure 5.

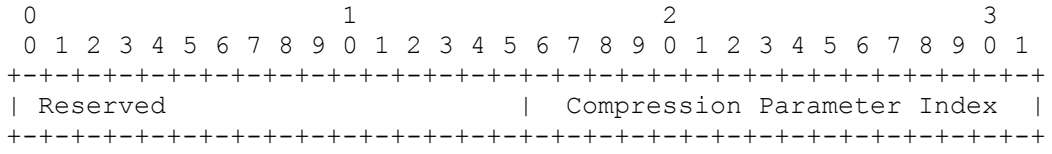


Figure 5: The Payload Compression Field (from RFC 3173)

The fields are:

- Reserved: 16 bits
  Senders shall set these bits to zero, and receivers shall ignore these bits.
- Compression Parameter Index (CPI): 16 bit-index
  Senders shall set the CPI to indicate the compression algorithms, as per RFC 3173. Values 0-63 (in network transmission order) designate well-known compression algorithms from RFC 3173, which require no additional information, and are used for manual setup.

5.2.7 Payload Format Descriptor Field

The Payload Format Descriptor Field is used to indicate the content of the payload. Senders shall always include this field when the Type field is set to 5 (0101). If the Type field is not set to 5 (0101), senders may include the Payload Format Descriptor field to indicate the payload content. The coding of the Payload Format Descriptor field is designed as a pointer to a standard, recommendation or specification that defines the packet payload. The format of the Payload Identification Field is shown in Figure 6.

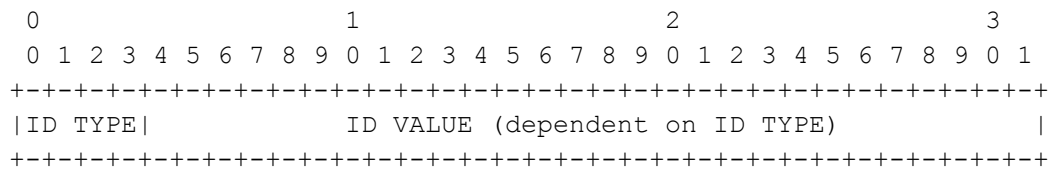


Figure 6: The Payload Format Descriptor Field

Senders shall set the fields of the Payload Format Descriptor Field as follows:

- **ID Type:** 4 bits
  Senders shall set this field to identify a standards body category, and to define the format of the **ID Value** field. The following ID Types are defined:
  - 0: VSF Category.
  - 1: RFC Category.
  - 2: SMPTE Category.
  - 3: IEEE Category.
  - 4: ISO/IEC Category.
  - 5: AES Category.
  - 6: ATSC Category.
  - 7: ITU Category.
  - 8: ETSI/DVB Category.
  - 9: EBU Category.
  - 10-14: Reserved for future use.
  - 15: Vendor defined type: reserved for private allocation to vendors. Note that interoperability is not guaranteed in this case.
- **ID VALUE:** 28 bits
  Senders shall use this value to identify the exact Standard, RFC or Recommendation. The formatting of this field is dependent on the ID TYPE field.

If more than one type of payload is encapsulated in a packet, the Payload Format Descriptor shall indicate the outermost format.

If more than one possible standard applies to the payload, the Payload Format Descriptor shall indicate the most restrictive one that applies.

Receivers may discard packets with Payload Format Descriptor values signaling formats that it is not prepared to process or accept.

The Video Services Forum has established a Payload Format Descriptor Registry and a registration procedure, to provide a canonical repository of the acceptable values of this field. Only the values present in the registry are shall be used for the Payload Format Descriptor. The exact assignment of the ID VALUE bits can be found in the Registry as well. The Registry can be accessed using the following link:

https://github.com/vsf-tv/rist-pfd

5.3 Tunnel Control Packets

The format of the Tunnel Control Packets is shown in Figure 7, where the Control Packet is highlighted in gray (immediately after the Figure 1 header). The fields are:

- **Control Index:** 16 bits
  Senders shall use this field to identify the type of Control Packet, as indicated in Table 2.
- **Length:** 16 bits
  Senders shall use this field to indicate the size of the Advanced Profile Control Message following the Length field, in bytes. There is no requirement that the size of the message be a multiple of 32 bits. Unlike RTCP packets, senders shall not stack Control Messages, i.e., a Tunnel Control Packet shall have only one Control Message

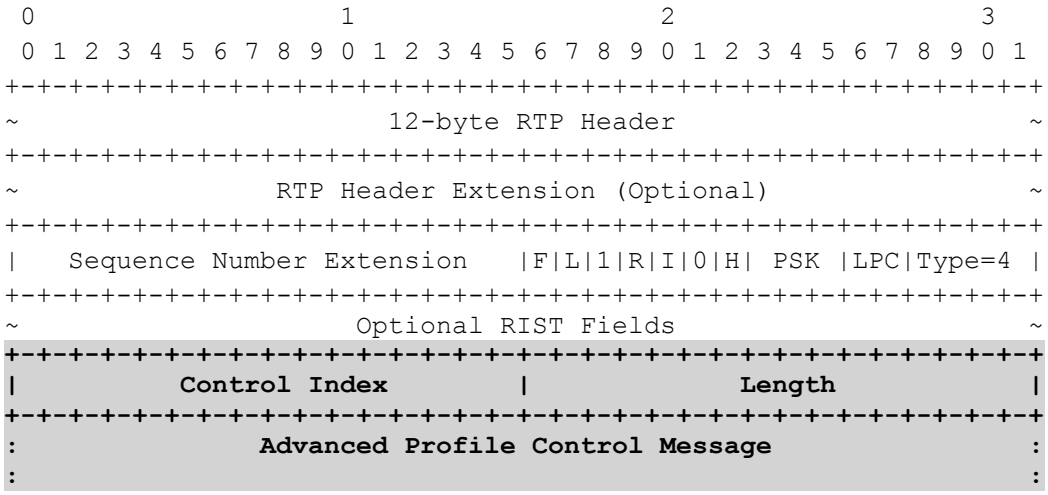


Figure 7: Control Message Format

The Control Index values shall be set by senders as described in Table 2.

All devices compliant with this Specification shall support the messages marked as mandatory in Table 2.

Devices compliant with this Specification may implement packet loss recovery using ARQ. Such devices shall accept and process both the NACK Bitmask (Section 5.3.2) and the NACK Range (Section 5.3.3) messages.

Devices compliant with this Specification may implement packet loss recovery using FEC, with the FEC packets being transmitted as control packets. If such devices support ST 2022-5 or ST 2022-1, they shall use the corresponding message from Section 5.3.5.

Table 2: Control Index Values

Control Index	Message Type	Mandatory
0x0000	NACK Bitmask	
0x0001	NACK Range	
0x0002-0x000F	Reserved for future NACK messages	
0x0010	RTT Echo Request	
0x0011	RTT Echo Response	Yes
0x0012-0x001F	Reserved for future RTT messages	
0x0020	ST 2022-5 FEC Row Packet	
0x0021	ST 2022-5 FEC Column Packet	
0x0022	ST 2022-1 FEC Row Packet	
0x0023	ST 2022-1 FEC Column Packet	
0x0024-0x002F	Reserved for future FEC messages	
0x0030-0x77FF	Reserved for future control messages	
0x7800-0x7FFF	Reserved for private vendor use	
0x8000	RIST Main Profile Keep-Alive message	Yes
0x8001	Flow Attribute message	
0x8002-0x800F	Reserved for future tunnel messages	
0x8010	Advanced Profile SRP authentication for PSK sessions	
0x8011	PSK Future Nonce Announcement Message	
0x8012-0x801F	Reserved for future authentication messages	
0x8020	Control Message Unsupported Response	
0x8021-0xF7FF	Reserved for future control messages	
0xF800-0xFFFF	Reserved for private vendor use	

Control messages with Control Index between 0x0000 and 0x7FFF shall only be sent as Unprotected packets (odd SSRC). Control messages with Control Index between 0x8000 and 0xFFFF may be sent either as Protected or Unprotected packets.

Note: some control messages in this list have corresponding RTCP packets in TR-06-1 and TR-06-2. Even though the functionality is the same, the actual message format will be different. In particular, NACK messages include the full 32-bit sequence number.

5.3.1 RIST Advanced Profile Flags for Control Messages

Control message senders shall set the flags defined in section 5.2.3 as follows:

- **F**, **L** flags:
  - Senders shall not fragment control messages with Control Index between 0x0000 and 0x001F. For such messages, senders shall set the F and L flags to '1'.
  - Senders may fragment control messages with other Control Index values. For such messages, senders shall set these flags per section 5.2.3.
- **E** flag: Senders shall mark all control messages as expedited by setting the E flag to '1'.
- **R** flag:

    - For control messages sent as Unprotected packets, senders shall always set the R flag to ‘0’ as they are not subject to retransmission.
    - Receivers may request retransmission of control messages sent as Protected packets. Senders shall set the R bit to ‘1’ in such retransmissions.
- **I** flag: control messages may include the optional Flow ID field. Senders shall signal the inclusion of the Flow ID field using the I flag as per section 5.2.3.
- **P** flag: control messages shall not have the Payload Format Descriptor field. Senders shall set the P flag to ‘0’ and shall not include the Payload Format Descriptor field in the header.
- **H** flag: control messages may include the optional RIST Header Extension field. Senders shall signal the use of the RIST Header Extension field by using the H flag as per section 5.2.3.
- **PSK** field: control messages may be encrypted using PSK. Senders shall set the PSK field as per section 5.2.3.
- **LPC** field: Senders shall not compress control messages with control index in the 0x0000-0x7FFF range, and shall set the LPC field for such messages to ‘00’. Senders may compress control messages with control index in the 0x8000-0xFFFF range if the remote end signals support for lossless compression.

5.3.2 NACK Bitmask Control Message Format

The NACK Bitmask control message may be used by a receiver to request retransmission one or more sets of up to thirty-three lost packets. The message format is shown in Figure 8. Senders implementing ARQ shall support this message. This message shall never be fragmented. If the number of lost packet sets to be communicated to the sender is such that the resulting message exceeds the link MTU, the receiver shall use multiple individual messages with sizes less than the link MTU.

The receiver shall set the fields in the NACK Bitmask Control Message as follows:

- **SSRC of Media Source:** 32 bits
  The receiver shall set this field to the SSRC of the media source (incoming stream) to which this NACK message is related.
- **Packet Sequence Start (PSS):** 32 bits
  The receiver shall set this field to the extended 32-bit RTP sequence number of the first packet lost.
- **Bitmask of following Lost Packets (BLP)**: 32 bits
  The receiver shall use the BLP field to report losses of any of the 32 packets immediately following the packet indicated by the PSS. Denoting the BLP's least significant bit as bit 1, and its most significant bit as bit 32, the receiver shall set bit i of the bit mask to ‘1’ to request retransmission of packet number (PSS+i) (modulo 2^32). If BLP is zero, then only one packet (with sequence number PSS) is being requested by the receiver. Senders

shall not assume that a receiver has received a packet because its corresponding bit mask in the BLP was set to 0.

Example: the least significant bit of the BLP would be set to '1' if the packets corresponding to PSS and PSS+1 been lost. However, senders cannot infer that packets PSS+2 through PSS+32 have been received simply because bits 2 through 32 of the BLP are '0'; all senders know is that the receiver has not reported them as lost at this time.

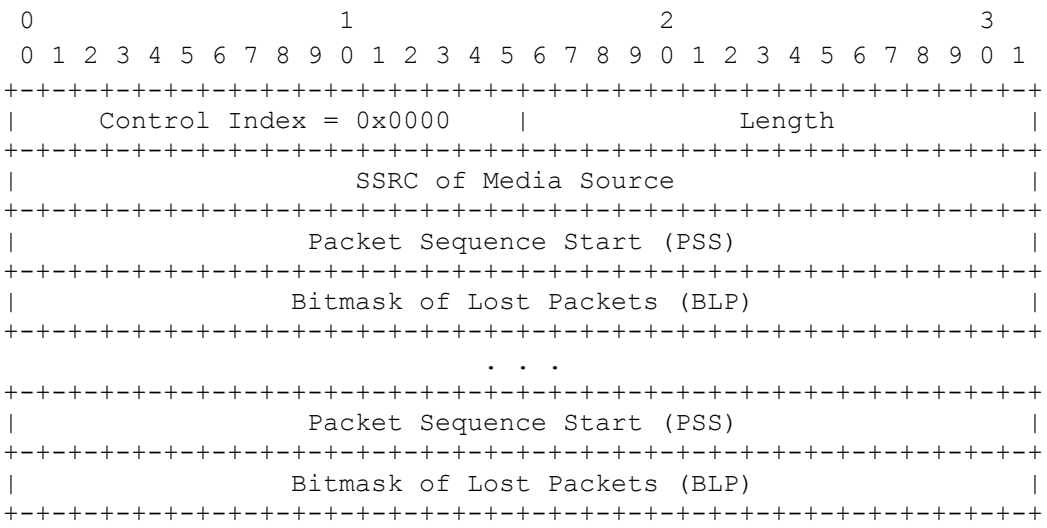


Figure 8: NACK Bitmask Control Message

5.3.3 NACK Range Control Message Format

The NACK Range control message may be used by a receiver to request retransmission one or more contiguous ranges of lost packets. The message format is shown in Figure 9. Senders implementing ARQ shall support this message. This message shall never be fragmented. If the number of lost packet ranges to be communicated to the sender is such that the resulting message exceeds the link MTU, the receiver shall use multiple individual messages with sizes less than the link MTU.

The receiver shall set the fields in the NACK Range Control Message as follows:

- **SSRC of Media Source:** 32 bits
  The receiver shall set this field to the SSRC of the media source (incoming stream) this NACK message is related to.
- **Packet Sequence Start (PSS):** 32 bits
  The receiver shall set this field to the extended 32-bit RTP sequence number of the first packet lost.
- **Number of Additional Lost Packets (NALP):** 32 bits
  The receiver shall set this field to the number consecutive packets being requested **after** the packet identified by the PSS.

Example: the if PSS is **N** and NALP is **A**, this indicates that packets from **N** to **N+A** inclusive have been lost. If NALP is zero, then only one packet (with the sequence number given by PSS) is being requested.

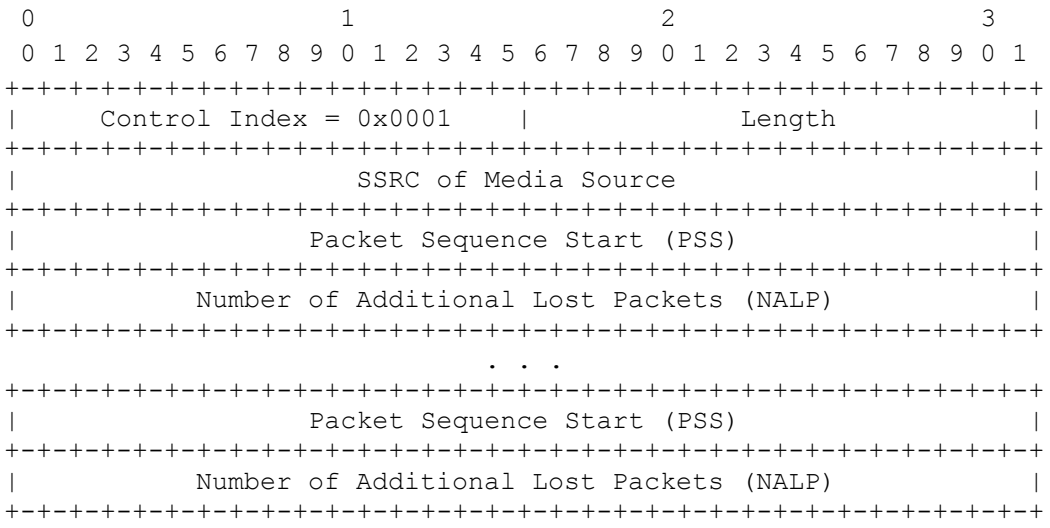


Figure 9: NACK Range Control Message

5.3.4 RTT Echo Request/Response Control Message Format

RTT Echo Request/Response Control Messages are used to measure the Round Trip Time (RTT) between two devices. All devices shall accept RTT Echo Request messages and shall respond with the RTT Echo Response message described in this section. Devices may impose a minimum interval of 100 milliseconds between responses.

The format of the RTT Echo Request/Response control messages is shown in . If the message sender uses padding, it shall select the size of the padding to ensure that the messages are not fragmented.

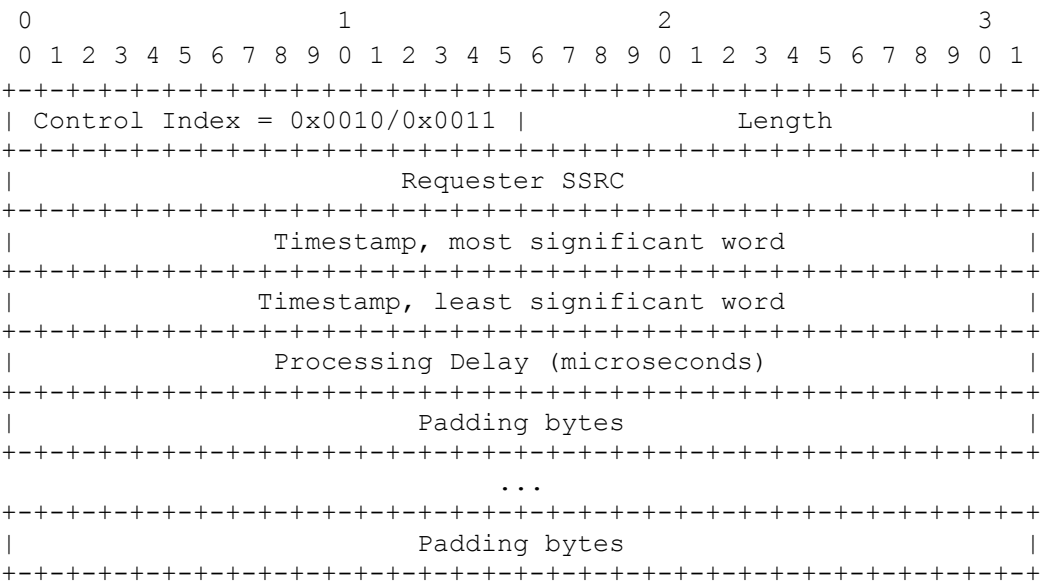


Figure 10: RTT Echo Request/Response Control Message

Senders shall set the fields in the message as follows:

- **Control Index:** Senders shall set the Control Index to 0x0010 for the RTT Echo Request Message, and to 0x0011 for the RTT Echo Response Message, as indicated in Table 2.
- **Length:** as defined in section 5.3, senders shall set this field to the size of the control message that follows. The length of RTT Echo messages shall be a multiple of 4 bytes.
- **Requester SSRC:** 32 bits
  The originator of the RTT Echo Request message shall set this field to its own SSRC (the same value used in the RTP header for this packet). In the RTT Echo Response message, the sender shall set this field to the SSRC value received in the corresponding RTT Echo Request message.
- **Timestamp:** 64 bits
  The originator of the RTT Echo Request message shall fill in an arbitrary value in this field, and the recipient of the message shall echo it back in the RTT Echo Response Message. In order to aid debugging, the timestamp may be in NTP format: the Timestamp most significant word may be a value in seconds, and the Timestamp least significant word may be the fractional part. There is no requirement that this be the actual NTP time or that the nodes be NTP synchronized.
- **Processing Delay:** 32 bits
  In RTT Echo Request messages, senders shall fill this field with zeros, and receivers shall ignore the value of this field. In RTT Echo Response messages, senders shall code their processing time in microseconds, expressed as a 32-bit unsigned integer in network byte order. The processing time is defined as the interval between the instant the RTT Echo Request message is received and the RTT Echo Response message is transmitted. The following considerations apply:

- Devices that respond as soon as possible may code zero in this field.
- Devices that purposefully delay a response shall use this field to indicate the amount of such delay.

Note: Devices are not required to have microsecond-precision in the delay measurement. They can use whatever clock precision they have, and express the final result in microseconds.

- **Padding:** n*32 bits
  The RTT Echo Request sender may want to measure the RTT for a packet of a certain size, so it may pad the packet with a number of additional bytes, with arbitrary content. The number of padding bytes inserted by the RTT Echo Request sender shall be a multiple of 4, and the resulting packet shall not exceed the link MTU. If the link MTU is symmetric (i.e., the same in both directions), RTT Echo Response packet shall have the same padding size as the corresponding RTT Echo Request, and the content of the padding bytes should be the same. If the link MTU is asymmetric, the size of the padding field shall be selected by the RTT Echo Response sender to ensure that the packet is not fragmented.

RTT Echo Response messages shall always be transmitted by senders as unicasts directed to the source IP address of the RTT Echo Request, even if the RTT Echo Request was received as a multicast.

5.3.5 SMPTE ST 2022 FEC Control Message Format

SMPTE ST 2022-1 and ST 2022-5 FEC use two additional UDP ports, one for the Column FEC packets and another for the Row FEC packets. It is possible to use ST 2022-1 or ST 2022-5 on the Protected packets and carry the FEC packets on separate UDP ports. However, it may be desirable to use a single UDP port for all the RIST Advanced Profile traffic. The SMPTE ST 2022 FEC control messages are provided as a means to carry the FEC packets in the same UDP port as the data. Devices supporting ST 2022-1 and/or ST 2022-5 FEC may use these messages to carry the FEC packets.

The SMPTE 2022 FEC control message is shown in Figure 11.

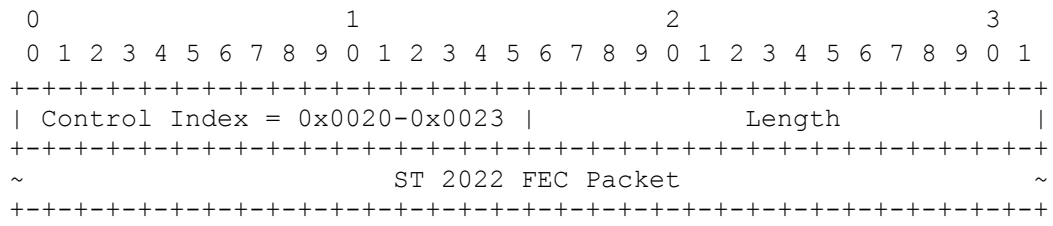


Figure 11: ST 2022 FEC Control Message

Senders shall use the field indicated as **ST 2022 FEC Packet** as follows:

- For ST 2022-5 FEC packets (Control Index = 0x0020/0x0021), senders shall set this field to include the FEC header from ST 2022-5 section 7.3, followed by the FEC data.
- For ST 2022-1 FEC packets (Control Index = 0x0022/0x0023), senders shall set this field to include the FEC header from ST 2022-1 section 8.4 followed by the FEC data.

In both cases, the FEC packets shall be computed over the payload of the final RTP packets in the Protected flow, and shall include the full RIST Advanced Profile header defined in section 5.2.

If the packets are PSK-encrypted, the FEC packets shall be computed **after** encryption.

Since FEC control messages will be larger than the packets they protect, these messages can exceed the MTU size. In this case, they shall be fragmented as described in section 5.2.3.

While ST-2022-1 and ST 2022-5 permit a number of different matrix sizes, implementations compliant with this document shall limit the matrix size as described below. 'L' denotes the number of columns, and 'D' denotes the number of rows:

- For ST 2022-1 implementations:
  - Column Only FEC
    - $1 \leq L \leq 20$
    - $4 \leq D \leq 20$
  - Column and Row FEC
    - $4 \leq L \leq 20$
    - $4 \leq D \leq 20$
  - Maximum matrix size: $L \times D \leq 100$
- For ST 2022-5 implementations:
  - Column Only FEC
    - $1 \leq L \leq 1020$
    - $4 \leq D \leq 255$
  - Column and Row FEC
    - $4 \leq L \leq 1020$
    - $4 \leq D \leq 255$
  - Maximum matrix size: $L \times D \leq 6000$

When using PSK, senders shall perform the FEC operation after PSK encryption. Senders may encrypt the FEC packets. If senders encrypt the FEC packets, receivers shall handle the case where the Nonce changes during a FEC matrix, and cache the current and previous keys as they will be required to decrypt packets in the FEC matrix.

Note: it is not technically necessary to encrypt the FEC packets, as they are computed over the already-encrypted content.

If AES-CTR-HMAC encryption is used for the non-FEC packets, senders should use HMAC-SHA256 for the FEC packets in order for the receiver to authenticate the FEC packet.

5.3.6 RIST Main Profile Keep-Alive Control Message Format

The purpose of this message is to encapsulate a Main Profile Keep-Alive message, without a GRE header. The format is shown in Figure 12. The definition of the message fields follows TR-06-2 section 5.5.3. Devices shall generate Keep-Alive messages once every 1 to 10 seconds, as defined by TR-06-2.

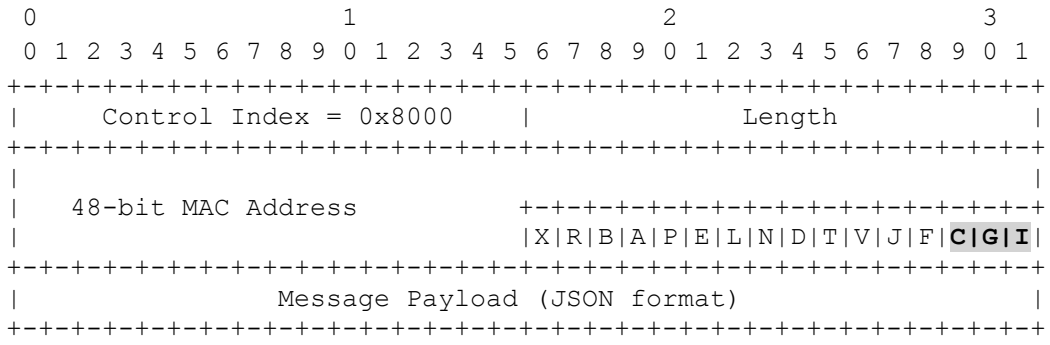


Figure 12: Main Profile Keep Alive Control Message

Three additional capability flags in the Main Profile Keep-Alive message are defined in this document. These bits are highlighted in gray in Figure 12. Senders shall set these bits as follows:

- **C** (bit 29): senders shall set this flag if it supports RIST Advanced Profile compression.
- **G** (bit 30): senders shall set this flag if it supports RIST Advanced Profile fragmentation.
- **I** (bit 31): senders shall set this flag if it supports RIST Advanced Profile operation. Usage of this bit is as follows:
  - If the Keep-Alive message is encapsulated in a RIST Advanced Profile Control Message as indicated in Figure 12, this bit shall be set to '1'.
  - If the Keep-Alive message is sent as a GRE packet as described in TR-06-2 section 5.5.3, this bit shall be set to '1' if the sending device supports TR-06-3 operation.

  Note: Devices compliant with TR-06-2 set the above bits to zero on transmission, and ignore them on reception. Therefore, the usage described in this section is backward-compatible with TR-06-2 devices.

  Note: Section 9 describes an optional mechanism for a sender to negotiate TR-06-3 operation with a remote device using the I bit.

Senders shall terminate the Message Payload field in Figure 12 with a byte set to zero. Receivers shall accept Message Payloads with or without the zero termination.

Senders may transmit Keep-Alive messages with no Message Payload.

5.3.7 Advanced Profile Flow Attribute Control Message Format

Senders may use Advanced Profile Flow Attribute Control Messages to describe the various flows present in the Advanced Profile RTP flow. The content of these messages is in JSON format and is described in section 5.4.

The format for Advanced Profile Flow Attribute Control Messages is shown in Figure 13. The JSON message defined in section 5.4 is the payload of the control message.

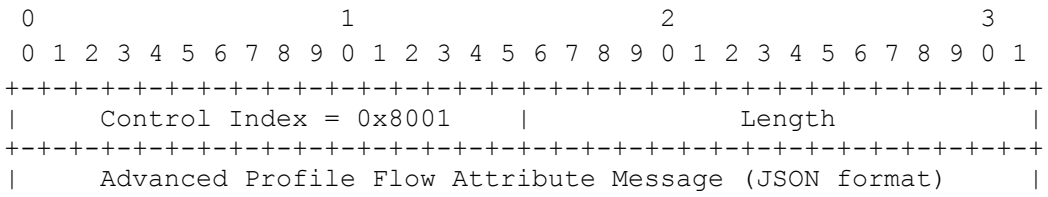


Figure 13: Advanced Profile Flow Attribute Control Message

Senders shall terminate the Advance Profile Flow Attribute Message (JSON format) with a byte set to zero. Receivers shall accept Advance Profile Flow Attribute Messages with or without the zero termination.

5.3.8 Advanced Profile EAP-SRP Authentication for PSK Sessions

Devices supporting PSK operation may use SRP for authentication, as described in Section 8.5. If used, the SRP messages shall be encapsulated into Advanced Profile Control Messages as shown in Figure 14.

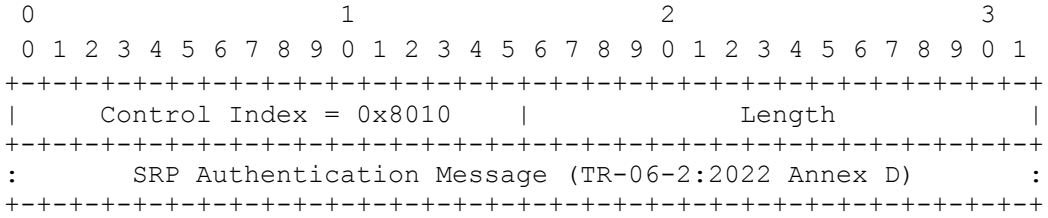


Figure 14: Advanced Profile EAP-SRP Authentication Message

5.3.9 PSK Future Nonce Announcement Message

In PSK mode, the receiving device needs to re-compute the AES decryption key from the passphrase once the Nonce changes. This operation is CPU-intensive, and it may take a

significant amount of time to complete. This can become an issue in high bit rate situations. In such cases, senders may use the PSK Future Nonce Announcement Message to inform the receiving node of the next Nonce value, so it can pre-compute and cache the AES key. If a sender makes use of the PSK Future Nonce Announcement Message, the sender shall format the message as follows:

- Senders shall transmit the PSK Future Nonce Announcement Message prior to a change in Nonce. The period of time between the transmission of this message and the Nonce change shall be no less than one second.
- Senders may transmit this message multiple times to guarantee delivery.
- Senders may transmit this message in the protected flow.
- After sending a message with a given Nonce value, senders shall not send a Future Nonce Announcement Message with a new Nonce value until the announced Nonce is in use for encryption.

The format for the PSK Future Nonce Announcement Message is shown in Figure 15.

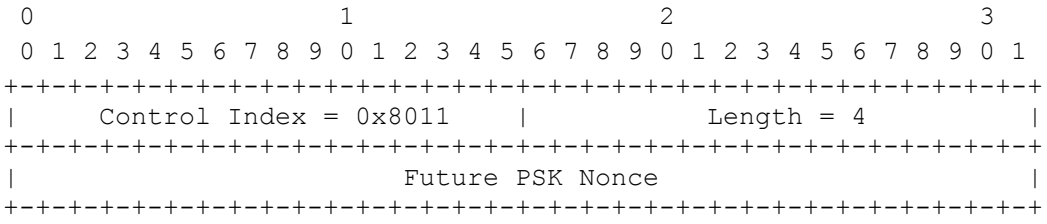


Figure 15: PSK Future Nonce Announcement Message

Senders shall set the **Future PSK Nonce** field in Figure 15 to the next Nonce to be used in PSK operation.

5.3.10 Control Message Unsupported Response

If a device compliant with this specification receives a control message that it does not support or recognize, it may send a Control Message Unsupported Response to indicate this fact. Control Message Unsupported Responses shall not be sent more often than once a second. The format for the Control Message Unsupported Response is shown in Figure 16.

Senders shall set the fields in the message as follows:

- **SSRC of Control Source:** 32 bits
  The sender shall set this field to the SSRC of the incoming control packet for which response is being sent.
- **Incoming Control Index:** 16 bits
  Senders shall set this field to the Control Index of the unsupported message. This allows the receiver to identify which message is unsupported.

- **First 48 bits of payload:** 48 bits
  Senders shall set this field to the first 48 bits (6 bytes) of the payload from the unsupported control message. If the payload of the message is less than 6 bytes, senders shall pad the remaining bytes in this field with zero.

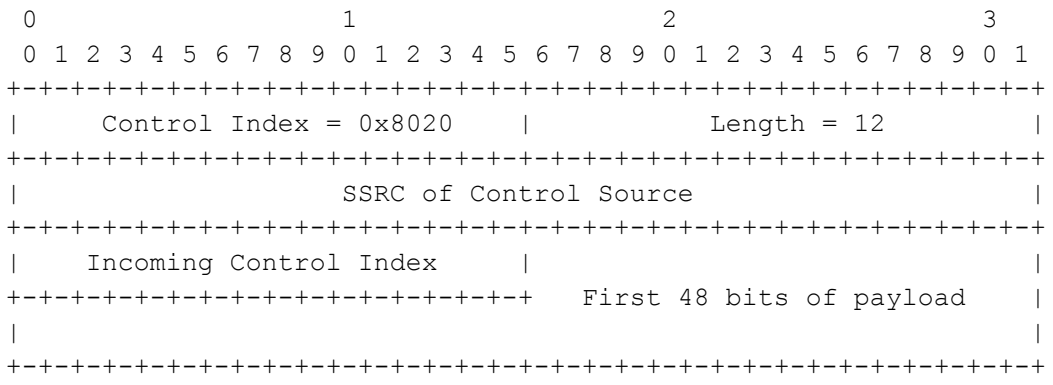


Figure 16: Control Message Unsupported Response

This message shall not be sent in response to control messages with syntax errors. It shall only be sent in situations where the Control Index is not recognized.

5.3.11 Vendor Private Control Message Format

This specification defines two ranges of control indexes for Vendor Private messages:

- 0x7800-0x7FFF for messages that can only be sent in as Unprotected packets.
- 0xF800-0xFFFF for messages that can be sent either as Protected or Unprotected packets.

Vendor Private messages are, by definition, not interoperable. However, it is necessary to provide a mechanism whereby Vendor Private messages from different vendors can coexist. This is achieved by requiring an ISO/IEC 11578:1996 Annex A UUID in in the header of each message. The format is shown in Figure 17.

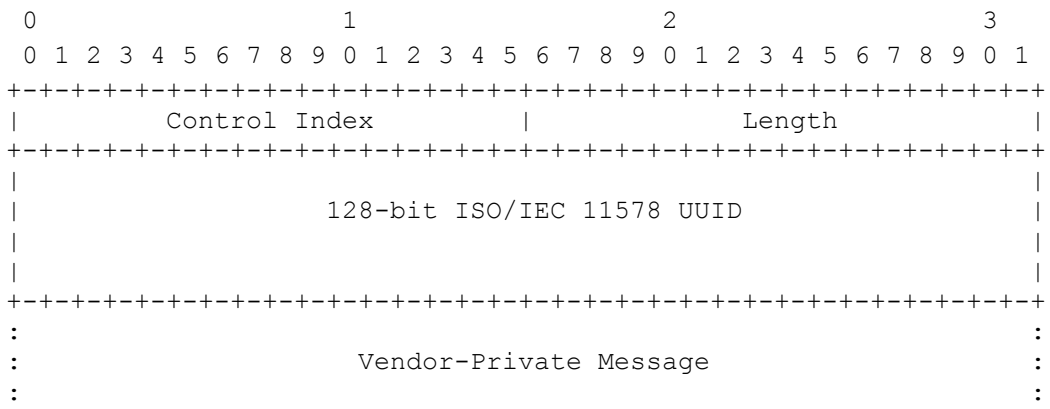


Figure 17: Vendor Private Control Message Format

As indicated in Figure 17, senders shall use first 16 bytes of every Vendor Private message to carry an UUID generated by the vendor. Using this method, a vendor can recognize their messages and the likelihood of two vendors selecting the same UUID is negligible.

Devices compliant with this Specification may discard all Vendor Private messages.

5.4 Advanced Profile Flow Attribute Messages

5.4.1 Message Definition

The Advanced Profile Flow Attribute Message is defined as follows:

```
{
    "sessions": [
        {
            "version": “1611655102:012345678”,
            “sid”: “8612b64f-77a2-4cd7-a1ca-6dfef9b0c012”,
            "flows": [
                {
                    "fid": "<INNER-FLOW-ID>",
                    "name": "<STRING>",
                    "description": "<STRING>",
                    "priority": "<NUMBER>",
                    "bitrate": "<NUMBER>",
                    "sdp": "<STRING>",
                    "subflows": [
                        {
                            "id": "<INNER-FLOW-SUB-ID>",
                            "fmt_des": "<NUMBER>",
                            "name": "<STRING>",
                            "description": "<STRING>",
                            "priority": "<NUMBER>",
                            "bitrate": "<NUMBER>",
                            "sdp": "<STRING>",
                            "<LABEL>": "<VALUE> (Optional payload properties)"
                        }
                    ],
                    "<LABEL>": "<VALUE> (Optional flow properties)"
```

```
                }
            ],
            "<LABEL>": "<VALUE> (Optional session properties)"
        }
    ]
}
```

Senders shall set the fields as follows:

- The “version” attribute shall be used to represent time in seconds and nanoseconds. A colon is used instead of a period to indicate that “:1” represents 1 nanosecond, not “.1” representing 100,000,000 nanoseconds (0.1 seconds).
- The “sid” field at the session level shall be a unique ID field for each session. The “sid” field may hold a UUID as shown in the above example, or any other unique ID generated by senders.
- The “sid” field shall remain constant while the session it identifies is present in the tunnel.
- The receiver shall compare the “version” and “sid” fields between two messages by a string comparison.
- The “subflows” nodes in the JSON message shall always be present to describe the details of the specific payload type for each flow or the details of some other custom multiplexing mechanism the implementer has used.
- The subflows "name" and "descriptions" shall be more specific than the same values at the flow level.
- The priority value at the subflows level shall represent the priority of that particular subflow within its flow. The priority value at the flow level shall represent the priority of that particular flow among other flows. Priority values from flows and subflows are independent and unrelated.
- SDP strings are optional and may be included at the flow or subflow level.
- SDP content shall mapped into JSON by taking the SDP data, including all whitespace, and encapsulate that into a string as per section 7 of RFC 8259.
- RFC 4566 mandates that SDP messages use DOS line endings (“\n\r”). These line endings shall be encoded with the above escape sequences in the JSON string.
- The “fmt_des” field shall include the value of the Payload Format Descriptor (defined in section 5.2.7). For packets where the Payload Format Descriptor is required, this field shall be included. Otherwise, if the Payload Format Descriptor is present in the packet flow, this field should be included.
- The receiver should only reparse the packet when the version number changes.
- JSON packets shall be transmitted at a rate of at least 2 Hz and no more than 10 Hz.

In high bandwidth links, a large number of flows and sessions may be present. In this case, multiple JSON packets may be generated by senders, each with a subset of the sessions. If such a mechanism is used, it shall be implemented as follows:

- The flows for a single session shall be sent in the same JSON packet. Senders shall not split data for a session across multiple packets.
- Notwithstanding the previous bullet, JSON packets should be the size of the MTU of the link or smaller.
- If a single session's data is larger than the MTU, then the payload fragmentation method described in section 5.2.3 shall be used to split one large JSON packet for the session into fragments.
- Senders may send one session per JSON packet.
- Senders may group multiple sessions per JSON packet, provided the combined packet remains below the link MTU.
- Wherever possible, senders should retain the same grouping of sessions into packets.
- If a session is terminated, senders should not reshuffle later session data into the packet that becomes smaller, and instead keep the existing packet structure for other sessions.
- Data for a new session may be added to any JSON packet that has available space within the link MTU.
- Data for all sessions shall be sent at least twice per second, provided the total rate for the session data does not exceed 10% of the available link capacity. If sending the data twice per second would exceed 10% of the link capacity, the data shall be sent at a rate that is approximately 10% of the link capacity.
- If the data for a session changes, the version field on that session shall be updated to the current timestamp. The data for the JSON packet that includes that session should be sent ten times per second for a period of one second after the change.
- If data for a given session is not received for 2 seconds (4 packet times), then the receiver shall assume this session has been terminated and shall remove the session from its table of sessions (or equivalent).

5.4.2 Usage of the Outer Flow ID Field

The Outer Flow ID field (see Section 5.2.4 and Figure 4) is used by senders to combine multiple logical tunnels into a single SSRC. In this case, each logical tunnel will have its own independent set of Inner Flow IDs and Inner Flow Sub-IDs.

If such a scheme is used, the packets containing the Advanced Profile Flow Attribute messages shall include the Flow ID field in the header. Such packets shall have the Inner Flow ID and Inner Flow Sub-ID fields set to zero. A given Advanced Profile Flow Attribute message with a given Outer Flow ID shall only apply to the packets with the same Outer Flow ID.

Note: The Outer Flow ID is not included in the JSON payload to avoid the need for rewriting Flow Attribute Messages when tunnels are combined.

5.4.3 JSON Schema

Senders shall use the JSON Schema below for the Advanced Profile Flow Attribute Message:

```
{
    "$schema": "https://json-schema.org/draft/2019-09/schema",
    "$id": "https://rist.tv/flow-attribute-schema.json",
    "title": "Flow Attribute Packet",
    "description": "RIST Advanced Profile Flow Descriptor Packet",
    "properties": {
        "sessions": {
            "items": {
                "required": [ "version", "sid" ],
                "properties": {
                    "version": { "type": "string" },
                    "sid": { "type": "string" },
                    "name": { "type": "string" },
                    "description": { "type": "string" },
                    "flows": {
                        "items": {
                            "required": [ "fid", "subflows" ],
                            "properties": {
                                "fid": { "type": "number" },
                                "name": { "type": "string" },
                                "description": { "type": "string" },
                                "priority": { "type": "number" },
                                "bitrate": { "type": "number" },
                                "sdp": { "type": "string" },
                                "subflows": {
                                    "items": {
                                        "required": [ "id" ],
                                        "properties": {
                                            "id": { "type": "number" },
                                            "fmt_des": { "type": "number" },
                                            "name": { "type": "string" },
                                            "description": { "type": "string" },
                                            "priority": { "type": "number" },
                                            "bitrate": { "type": "number" },
                                            "sdp": { "type": "string" }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

5.4.4 Receiver Operation (Informative)

It is suggested that receivers operate in the following way:

1. The receiver starts with an empty table for session information. It might restore a previous table if it is restarting after less than two seconds since it was previously running.

2. For each JSON flow descriptor packet received, the receiver iterates across all sessions within the session array in the received packet. The suggested processing for each session is:
    a. Search within its table for the ID of the session.
    b. If the ID is found, compare the version number that has been received with the stored version number.
    c. If they match, there is nothing more to do for this session.
    d. If they are different, update the data in the table with the new data from the received packet.
    e. If the ID was not found, add the data from the received packet to the table, keyed by the session ID.
3. Separately, the receiver holds a timer for when it last received a packet for each session. If no data has been received in a session for two seconds, then the session is deemed to have been terminated, and is removed from the table.
4. If the link bandwidth is low, then the receiver might need to detect that the data is being received less than twice per second for each session, and increase the timeout in point (3) above accordingly.

6 SSRC Multiplexing (Informative)

This Specification allows multiple tunnels to be multiplexed into the same UDP port. Each tunnel is defined by two consecutive SSRC values. Protected packets are transmitted using the even SSRC, and Unprotected packets are transmitted using the odd SSRC, as described in section 5.2.1. Support for this mode of operation is optional.

The following features of this Specification are available to implementers who desire to support SSRC multiplexing and provide additional functionality (in a non-interoperable way):

- If a "path-level" RTT Echo message is desired, the implementer might create an extra tunnel, and use the standard RTT Echo Request/Response messages defined in section 5.3.4 in this tunnel. This provides an RTT reading for the group of tunnels and obviates the need to have independent RTT Echo messages on each of the tunnels.
- If some sort of tunnel description message is required (e.g., to list all the tunnels in the UDP port and/or their parameters and features), the Vendor Private Control Messages from section 5.3.10 might be used.

7 DTLS Support

RIST senders and receivers may use DTLS version 1.2 to secure their communication and authenticate the endpoints. RIST devices offering DTLS support shall implement the DTLS protocol according to RFC6347 with the additional restrictions and recommendations from this section.

7.1 Session Establishment

Endpoints shall create one single DTLS session to carry the RIST Advanced Profile tunnel packets described earlier in this document.

Once the session is established, the endpoints should ensure that the final tunneled and encrypted packet size does not exceed the path MTU. Endpoint should use the fragmentation mechanism described in section 5.2.3 for packets that exceed the MTU.

7.2 Supported DTLS Cipher Suites

The following cipher suites shall be supported by all RIST devices implementing DTLS:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_NULL_SHA256

RIST devices shall provide a means for the user to disable individual cipher suites to match their local policies.

The NULL cipher only provides authentication, not encryption. Devices should disable it by default.

RIST devices may include other cipher suites specified in DTLS in their implementations.

Note: It is understood that disabling individual ciphers might prevent two RIST devices from establishing communication, if there is no common cipher enabled.

7.3 Certificate Configuration

- The DTLS server should be configured with a certificate file – either issued by a certificate authority, or a self-signed one. There is no limitation to the certificate type, as long as it is readable by the DTLS library and compatible with the cipher suites available.
- The DTLS client may validate the authenticity of the certificate and may perform hostname validation. If offered, this shall be a user-configurable option.
- The DTLS client should be configured with a client-side certificate. This may be done using username/password.
- The DTLS server may validate the client certificate. If offered, this shall be a user-configurable option.
- Both client and server should use a certified list of CAs. One option is to use the public CA list from the following link:
  https://hg.mozilla.org/releases/mozilla-beta/file/tip/security/nss/lib/ckfw/builtins/certdata.txt
  Within a given organization, a more secure option is to use a private CA trusted by all the

devices in that organization. To support this, devices should allow users to add to or completely replace any default list of CAs.

- Endpoints may be configured to allow self-signed certificates.

The certificate management discussion presented in Appendix A of VSF TR-06-2 also applies to this Specification.

7.4 TLS-SRP Support

The TLS-SRP protocol, as described in RFC 5054, is used to securely provide username/password authentication between devices, as an alternative to using certificates. RIST devices may implement TLS-SRP as an authentication method when using DTLS. If they do so, the implementation shall follow RFC 5054, with the following modifications and restrictions:

- RIST devices implementing TLS-SRP shall support the following cipher suites:
  - TLS_SRP_SHA_WITH_AES_128_CBC_SHA
  - TLS_SRP_SHA_WITH_AES_256_CBC_SHA
- RIST devices implementing TLS-SRP may additionally support any of the other cipher suites listed in RFC 5054 section 2.7.
- RIST servers implementing TLS-SRP shall be configured with a certificate file. This certificate file may be self-signed.
  Note: RIST clients with TLS-SRP support can safely ignore the certificate expiration date without compromising security.
- In order to make TLS-SRP more secure, RIST servers should implement the following policies:
  - For a given server, user names should be unique across accounts.
  - Servers should limit the rate of authentication attempts from a particular IP address in order to reduce the risk of brute-force password attacks.
  - Servers should have reasonable password strength policies in order to reduce the risk of brute-force password attacks.

8 Pre-Shared Key Encryption Support

Advanced Profile RIST devices may use Pre-Shared Key Encryption to secure their communication and authenticate endpoints. When offering PSK encryption, the devices shall implement the protocol according to the recommendations of this section. The Pre-Shared Key header fields defined in section 5.2.5 contain the information used to generate and rotate keys and initialization vectors (IV), as well as an optional hash to authenticate the packet contents. The PSK flags defined in section 5.2.3 determine whether the packet is encrypted, and the algorithm used in the encryption. The presence of the Nonce, IV and Hash fields is determined by the same flags, as indicated in Table 1. Authentication, if present, shall cover the whole RTP packet, as shown in Figure 18. Encryption, if present, shall cover the part of the packet from the Pre-Shared IV field to the end of the payload, as indicated in Figure 18.

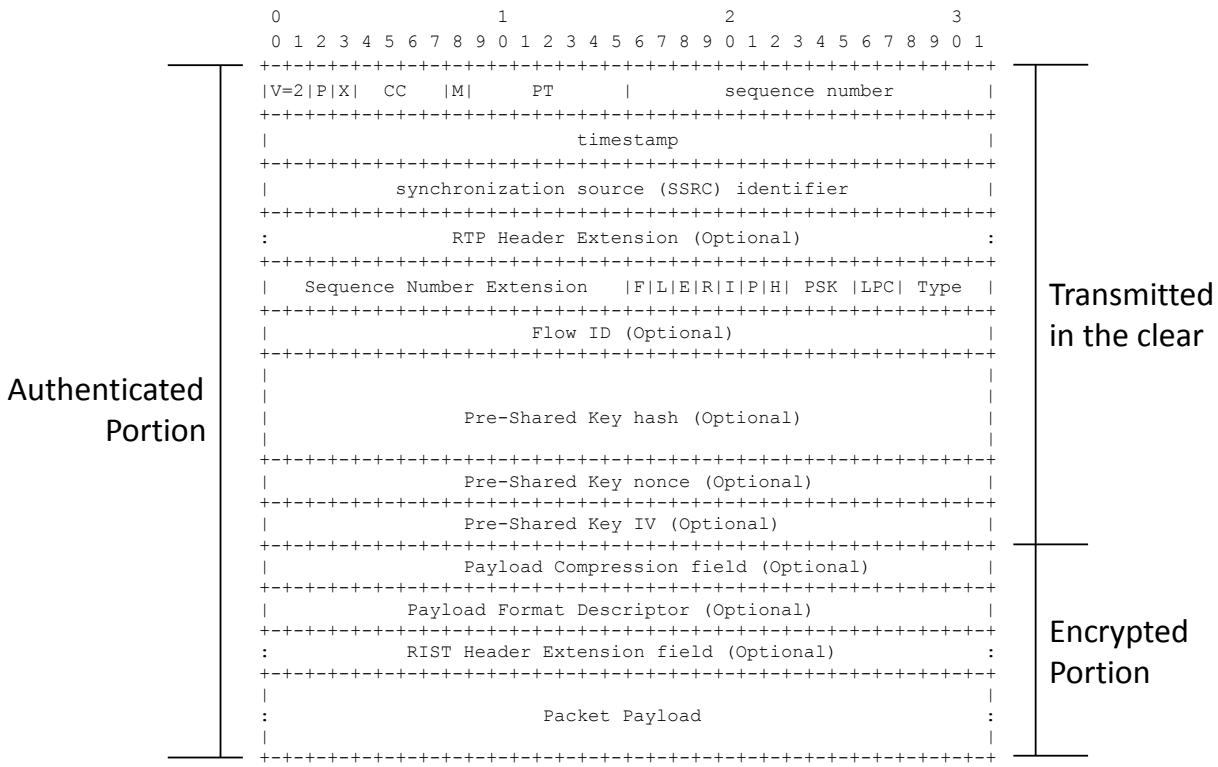


Figure 18: Clear, authenticated and encrypted fields using PSK

8.1 Authentication

The Pre-Shared Key hash provides authentication of the RTP header and payload, and it indirectly provides replay protection by authenticating the full 32-bit sequence number and RTP timestamp. If both encryption and authentication are applied, encryption shall be applied before authentication on the sender side.

A sender shall zero out the hash field before encrypting with AES-CTR-HMAC, AES-GCM and CHACHA20-POLY1305 or before authenticating with HMAC-SHA-256. A receiver shall copy the received hash value and zero out the hash field before performing the reverse action.

Senders shall calculate the Pre-Shared Key hash from the first byte of the RTP header until the last byte of the payload as shown in Figure 18.

8.2 Pre-Shared Key Nonce and IV

The Pre-Shared Key Nonce and IV fields shall be used as follows:

- The Pre-Shared Key Nonce field shall be set by senders to random, non-zero value. If PSK encryption is used in both the Protected and Unprotected packet flows (even/odd SSRCs), senders shall use different Nonce values for each flow.

- Senders shall increment the Pre-Shared IV field every time a packet is transmitted.
- Senders shall derive the 128-bit initialization vector (IV) for the encryption operation by using the 32-bit IV field as its most significant four bytes, followed by 12 bytes of zeros. This 128-bit quantity is the counter that generates the key stream. This process is illustrated in Figure 19.
- A new nonce shall be generated by senders at least every time the IV field wraps to zero. This ensures that the same IV + Nonce combination is never reused. The Future Nonce Announcement message described in section 5.3.8 may be used to pre-signal a new Nonce.
- The receiver shall inspect the Nonce field for every received packet, and shall re-generate the key any time it changes. The receiver may use the Future Nonce Announcement message from Section 5.3.9 to pre-generate the next key in advance of a Nonce change.

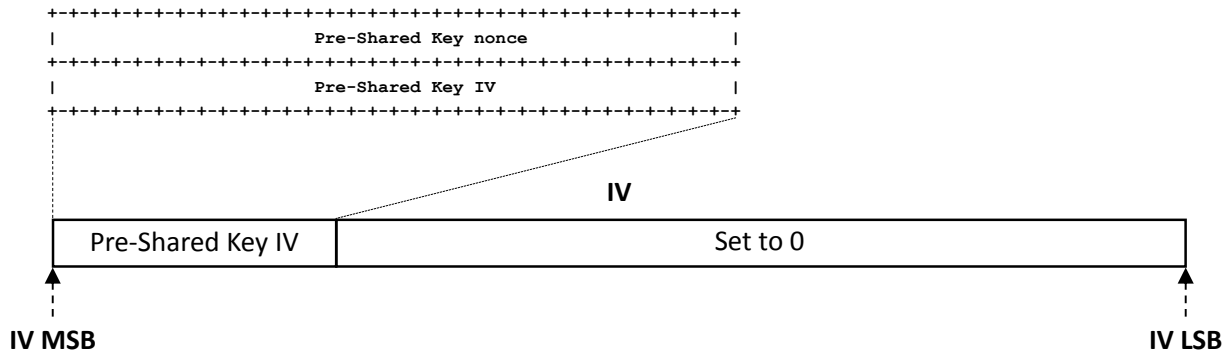


Figure 19: IV Generation

8.3 Encryption Key and Sequences

- The payload shall be encrypted by senders and decrypted by receivers using the encryption method selected by the PSK field (see section 5.2.3), using a 128-bit or 256-bit key derived through PBKDF2 as described in RFC 8018 section 5.2 and Appendix C.
- As per RFC 8018, a password is considered to be an octet string of arbitrary length whose interpretation as a text string is unspecified.
  Note: In the interest of interoperability, it is recommended that RIST devices follow some common text encoding rules for passwords. Examples are ASCII and UTF-8.
  Note: Octet strings are not required to have null terminators. If such terminators are desired, they need to be explicitly included in the passphrase.
- The default hashing algorithm for PBKDF2 shall be SHA-256 with 1,024 iterations.
- PSK implementations may offer other hashing algorithms from RFC 8018 with different numbers of iterations. If such options are offered, they shall be enabled by explicit out-of-

band configuration for all participants. The SHA-1 hashing function included in RFC 8018 is insecure and shall not be used.

- The key and IV used for both encryption and decryption shall be used as described in section 8.1 of this document. The PBKDF2 function shall use the Nonce as salt to generate the actual key. An example is provided in Appendix C.

Note: The resulting generated key is valid for up to 2^{32} packets. It is therefore safe to use the full 32-bit Pre-Shared Key IV for the AES operations on single packets. Since the Nonce (and therefore the AES key) changes when the IV wraps around, there is no risk of reusing the same IV+Nonce combination within the encrypted flow.

8.4 On-The-Fly Passphrase Change

In some one-to-many situations, it may become necessary to de-authorize a subset of the receivers. This section describes an optional mechanism for senders to change the passphrase on-the-fly with no service interruption for the receivers which remain authorized.

On-The-Fly Passphrase Change capability is optional.

The process is as follows:

- A new passphrase is distributed through out-of-band means to the receivers that are to remain authorized.
- This passphrase is loaded in all the relevant receivers, but remains inactive.
- Once all the relevant receivers are configured with the new passphrase, senders switch to a key generated by this new passphrase. This change is signaled in the GRE packets.

If On-The-Fly Passphrase Change capability is implemented, the Pre-Shared Key nonce field shall contain one bit, denoted by **B**, which identifies the passphrase to be used. The passphrase selected by **B**=0 shall be denoted as the "even passphrase", and the passphrase selected by **B**=1 shall be denoted as the "odd passphrase". Bit **B** shall be the MSB of the Nonce field, as indicated in Figure 20.

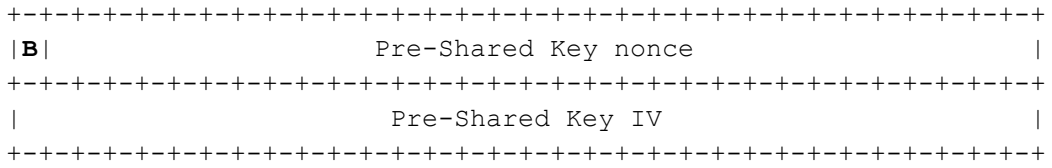


Figure 20: Pre-Shared Key nonce field with Odd/Even Bit B

Operation shall be as follows:

- All receivers shall use the full 32-bit Nonce field as the Nonce value for key generation.

- Receivers implementing support for odd/even passphrases shall initialize both passphrases to the same value. This ensures compatibility with senders that do not support different odd/even passphrases.
- Senders implementing support for odd/even passphrases shall initialize both passphrases to the same value, and may use any value for the Nonce.
- Passphrase changes shall occur as follows:
    - Senders and receivers who should remain authorized are configured with the new passphrase.
    - The new passphrase shall be associated with the value of **B** that is not currently in use. For example, if at configuration time, **B**=1, then the new passphrase will be associated with **B**=0.
    - Senders switch to the new passphrase by manual user intervention or other out-of-band means. At this time, senders shall generate a new Nonce with the inverted value of **B**. In the example above, the new Nonce will be set to **B**=0.
    - The Nonce change shall trigger a key recalculation at the receivers. Receivers supporting odd/even passphrases shall use the new passphrase.
    - From that point on, if senders decide to rotate the key, the new Nonce values shall have the same value of **B**.
    - This process can be repeated at a later point in time if a new passphrase change is required.

8.5 PSK Authentication Using EAP-SHA256-SRP-6

The PSK authentication method shall follow the protocol described in VSF TR-06-2:2022 Annex D. The EAP handshake packets shall be transmitted over the tunnel using control index 0x8010, as indicated in Table 2 and Figure 14. PSK authentication messages shall not be encrypted.

Once the peer has been authenticated, this authentication state should be cached for the duration of the session. The details of this process are left to the discretion of the implementer.

9 Interoperability with RIST Main Profile Devices

RIST Advanced Profile devices may interoperate with RIST Main Profile devices using the method described in this section. Support for this method is optional, but devices desiring compatibility with Main Profile shall do so using the method described in this section.

The method operates as follows:

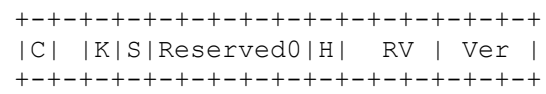
- The Advanced Profile device shall start in Main Profile mode, as described in VSF TR-06-2.
- The Advanced Profile device shall implement support for the Main Profile Keep-Alive message, as per section 5.5.3 of VSF TR-06-2.

- The Advanced Profile device shall set the **I** bit defined in section 5.3.6 of this Specification to "1" in the Keep-Alive messages it transmits. This bit indicates that the device is capable of Advanced Profile operation. The device may also set the **C** bit if it supports compression and the **F** bit if it supports fragmentation.
- If the Advanced Profile device receives a Main Profile Keep-Alive message with the **I** bit set to "1", it knows that the remote end is capable of Advanced Profile operation. If this bit is not set, the Advanced Profile device shall remain in Main Profile mode.
- The Advanced Profile device may switch to Advanced Profile operation at any time after it receives a Main Profile Keep-Alive message with the **I** bit set to "1".

A receiving Advanced Profile device can differentiate between a Main Profile packet (which starts with a GRE header) and an Advanced Profile packet (which starts with an RTP header) by inspecting the first 16 bits of the packet. The process is illustrated in Figure 21. The fields line up as follows:

  - The first two bits of an RTP packet are always 10 (version=2). TR-06-2 indicates that the C bit should be set to zero (no Checksum), so in most cases this is sufficient. Therefore, if the first bit of the packet is a zero, that packet is a Main Profile GRE packet.
  - If the first two bits of the packet are 10, this could be a Main Profile GRE packet with a checksum field or an RTP packet. The next step is to check the PT field. In a Main Profile GRE packet, this corresponds to the H, RV and Ver fields. The Ver field is always zero, and the RV field is either 000 or 001. The legal values for this field are 0 (TR-06-2:2020), 8 (TR-06-2:2021 with 128-bit AES key), or 72 (TR-06-2:2021 with 256-bit AES key). Therefore, if the bits corresponding to the PT field match what is expected for Advanced Profile (either 127 or a value signaled through SDP, which is 96 or higher), then the packet is an Advanced Profile Packet.

First 16 bits of a Main Profile GRE Packet



First 16 bits of a Main Profile RTP Packet

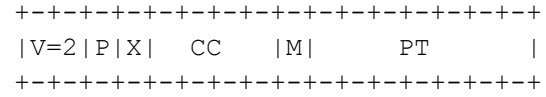
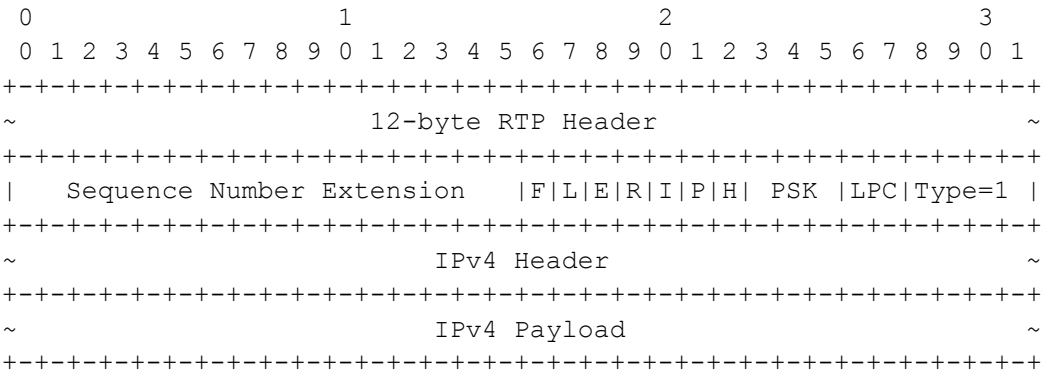


Figure 21: Detecting GRE or RTP on-the-fly

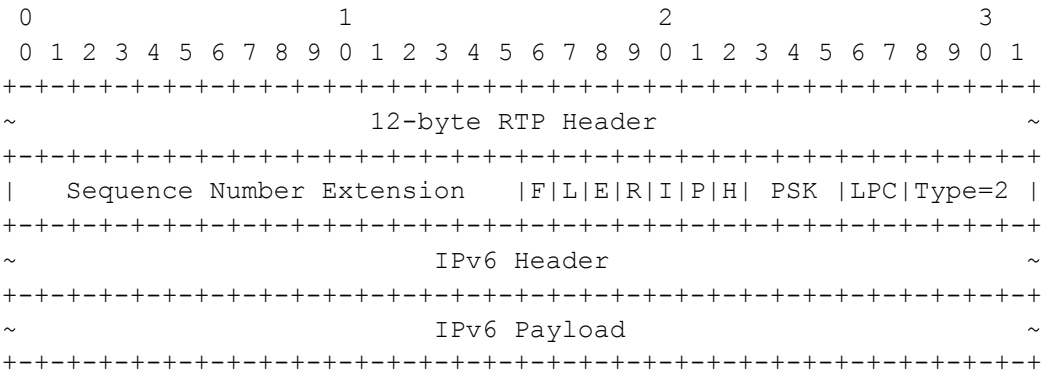
Appendix A Tunnel Packet Examples (Informative)

This section illustrates the formation of the tunnel packets for the various types defined in Section 5.2.3. In each example, only the required fields are shown. Optional fields can be present if signaled by the flags.

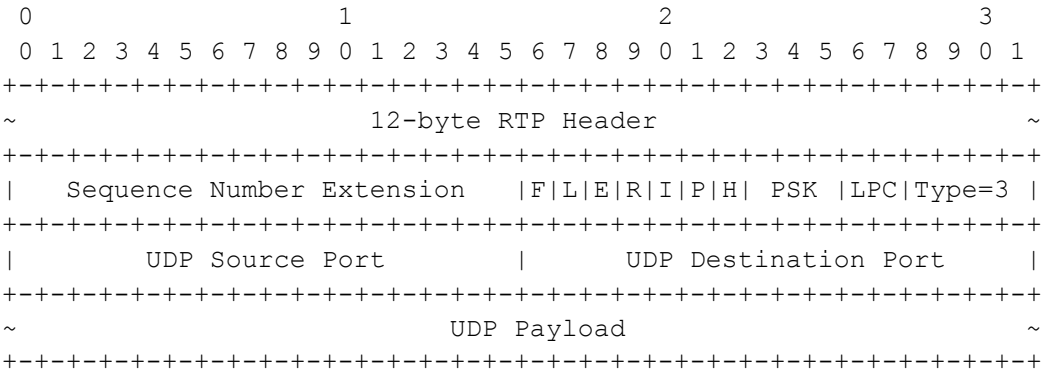
A.1 Type 1: IPv4 Packets



A.2 Type 2: IPv6 Packets

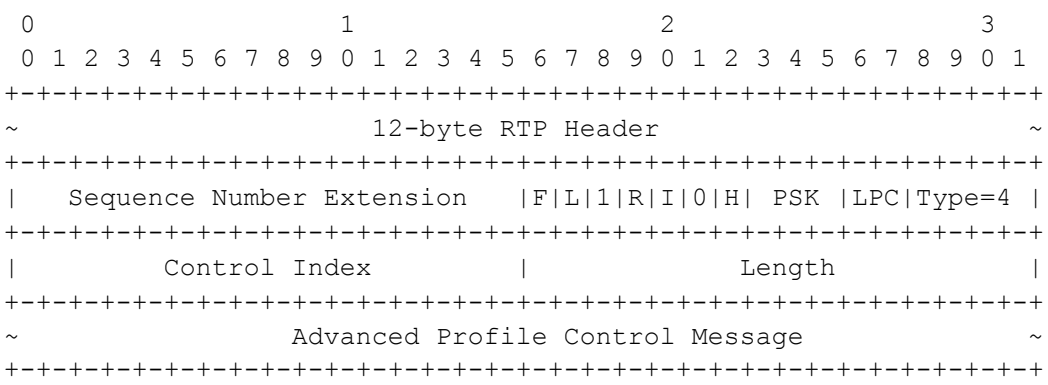


A.3 Type 3: Reduced Header UDP Packets from TR-06-2



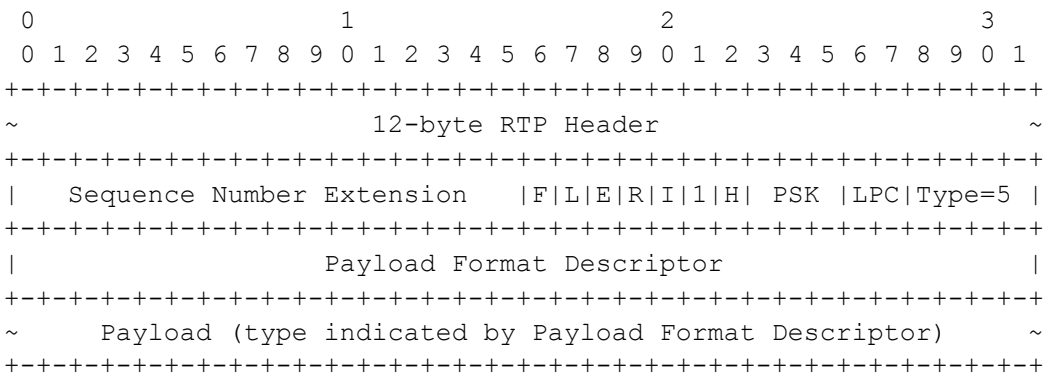
A.4 Type 4: Control Packets

Control packets have E=1 (Expedite bit set) and P=0 (no Payload Format Descriptor).



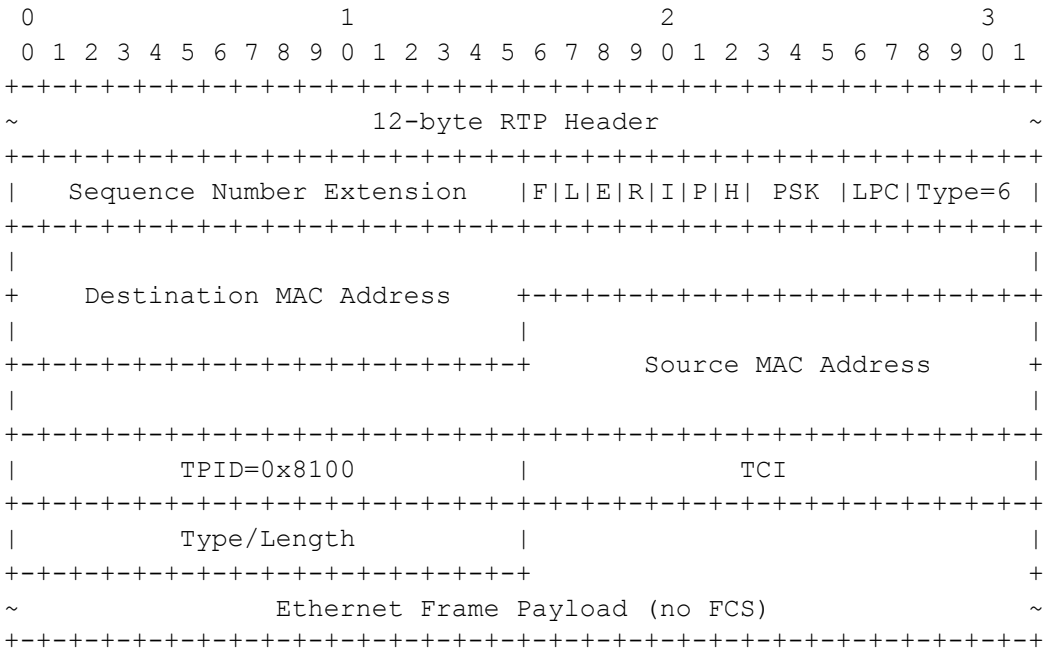
A.5 Type 5: Direct Payload Packets

Direct Payload packets have P=1 (include Payload Format Descriptor).



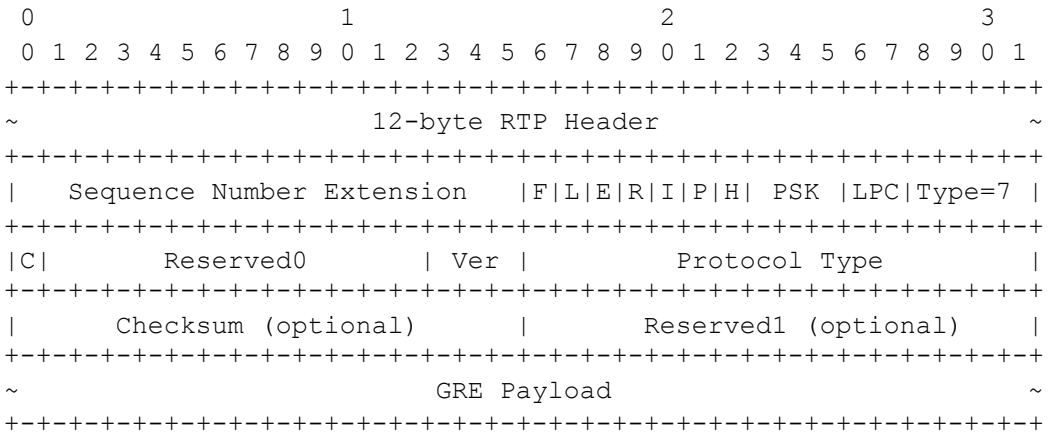
A.6 Type 6: Layer 2 Ethernet Frame

With Type=6, any Ethernet frame can be transported. The example below illustrates the transport of a VLAN-tagged Ethernet frame.



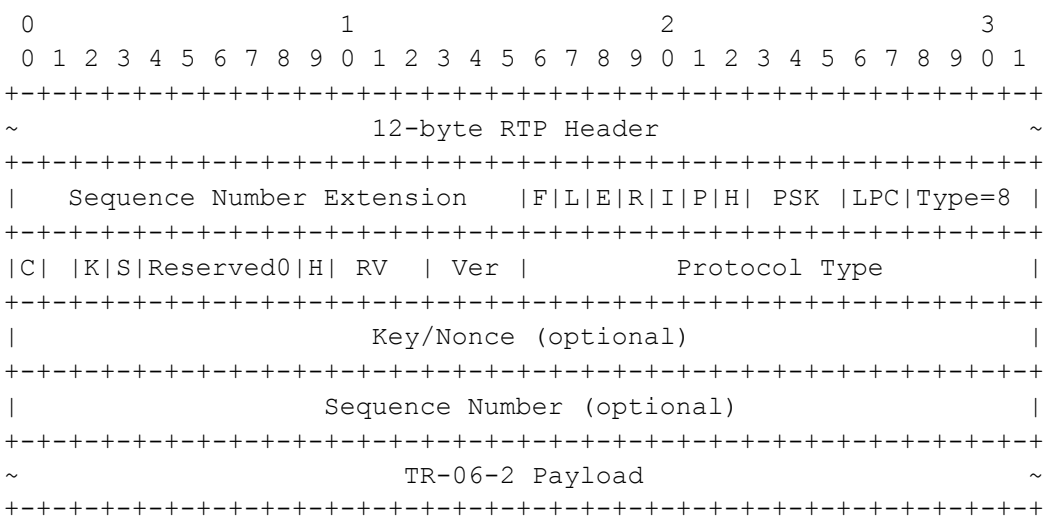
A.7 Type 7: RFC-2784 GRE Packets

The example below includes the fields marked optional in RFC-2784.



A.8 Type 8: TR-06-2 RIST Main Profile GRE Packets

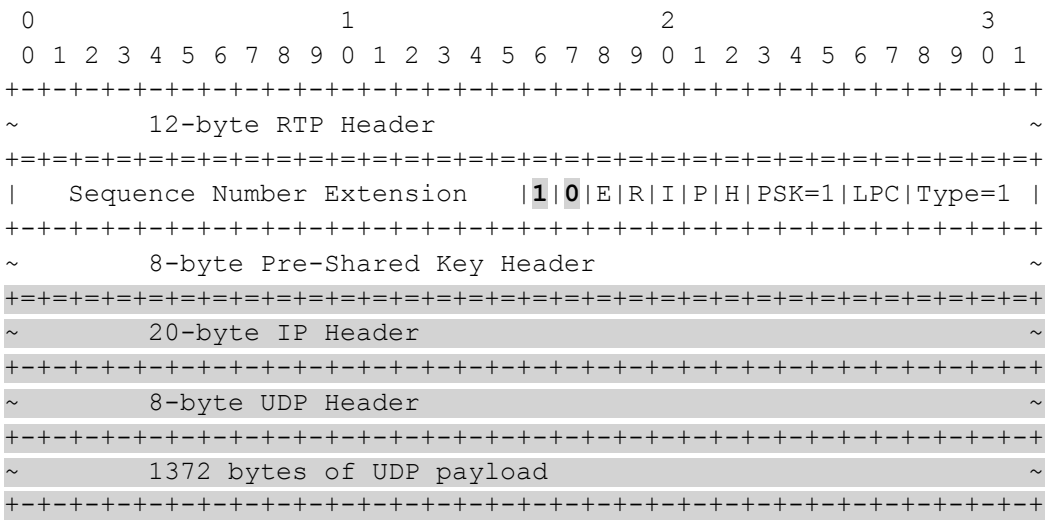
The example below includes the fields marked as optional in TR-06-2



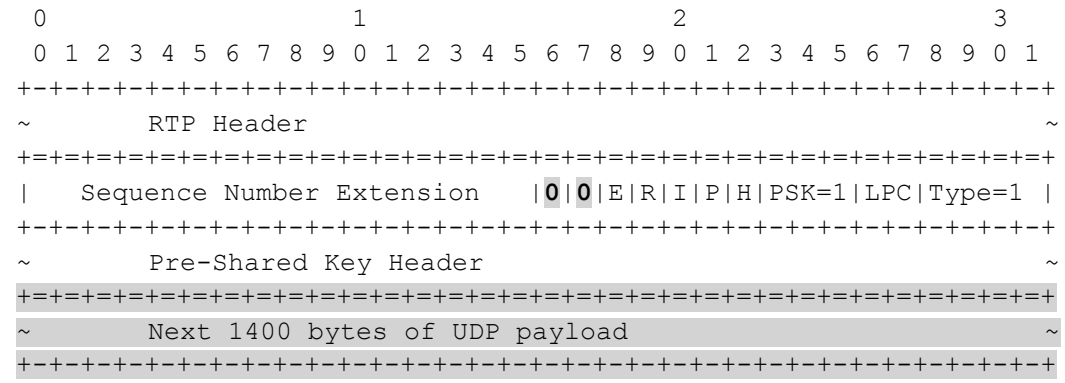
Appendix B Fragmentation Example (Informative)

The example assumes the transport through the RIST Advanced Profile tunnel of an IPv4 jumbo frame with a UDP payload of 4000 bytes, fragment size is 1400 bytes, and Type set to 1 to signal an IPv4 payload. The packet is encrypted with PSK AES-CTR mode, therefore the Pre-Shared Key PSK flag bits are set to 1 on all fragments. The total tunnel payload bytes to be fragmented are 4028: the IP Header (20 bytes) + UDP header (8 bytes) + UDP Payload (4000 bytes).

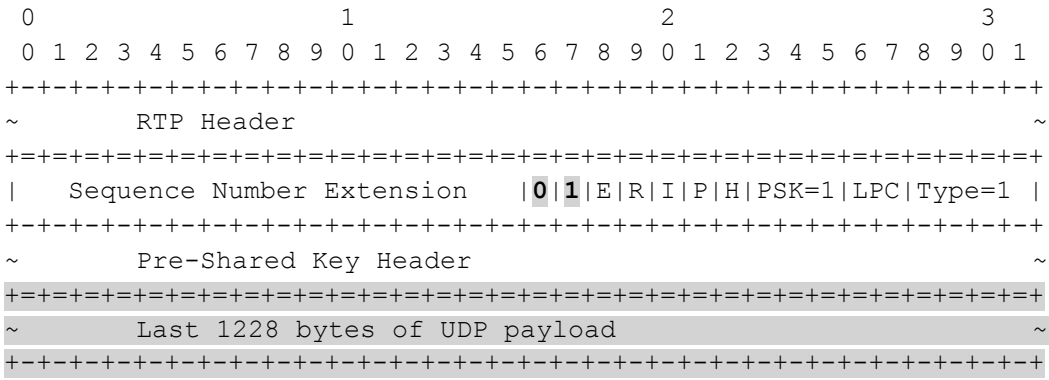
The first fragment contains 1400 bytes of tunnel payload: IP Header (20 bytes) + UDP header (8 bytes) + UDP Payload (1372 bytes). The F bit is set to 1 and the L bit is set to 0.



The second fragment contains the next 1400 bytes of the UDP payload. The F bit is set to 0 and the L bit is set to 0.



The last fragment contains the remainder of the UDP payload of 1228 bytes. The F bit is set to 0 and the L bit is set to 1.


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
RTP Header
Sequence Number Extension
0
1
E
R
I
P
H
PSK=1
LPC
Type=1
Pre-Shared Key Header
Last 1228 bytes of UDP payload

Appendix C PSK Key Generation Example (Informative)

This appendix provides one example of 128-bit and 256-bit AES keys generated from a known passphrase and nonce. It is provided to allow implementations to be checked against known values.

The inputs are:

- Passphrase: **`Reliable Internet Stream Transport`**
- Nonce: **`0x52495354`**

Figure 22 shows the packet received from the network with the above nonce.

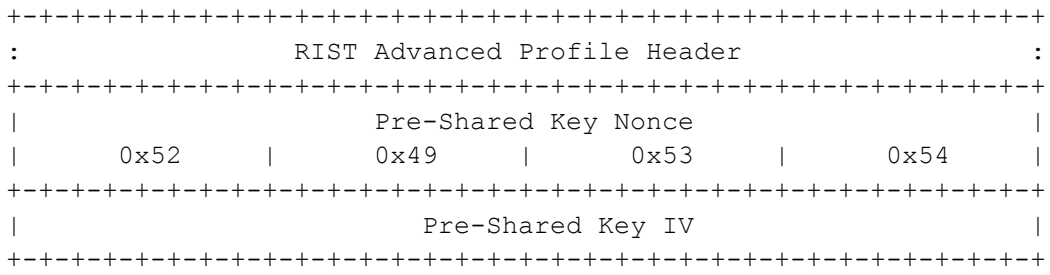


Figure 22: Sample Received Packet

Using the PBKDF2 hashing algorithm specified in section 8.3, the following keys are derived from this input:

- 128 bit key: `1c2b0cfc90ae2638fea78c7fb2977047`
- 256 bit key: `1c2b0cfc90ae2638fea78c7fb297704718bff7f4052743001a9b7ebb51cc9f1c`

The following Python 3 code can be used to generate the keys:

```
import hashlib

key = hashlib.pbkdf2_hmac("sha256", b'Reliable Internet Stream Transport', bytes.fromhex('52495354'), 1024,
16)
print("Derived 128 bit key:", key.hex())

key = hashlib.pbkdf2_hmac("sha256", b'Reliable Internet Stream Transport', bytes.fromhex('52495354'), 1024,
32)
print("Derived 256 bit key:", key.hex())
```