

AVTCORE
Internet-Draft
Intended status: Standards Track
Expires: 13 November 2022

J. Ott
M. Engelbart
Technical University Munich
12 May 2022

RTP over QUIC
draft-engelbart-rtp-over-quic-03

Abstract

This document specifies a minimal mapping for encapsulating RTP and RTCP packets within QUIC. It also discusses how to leverage state from the QUIC implementation in the endpoints to reduce the exchange of RTCP packets.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at
<https://github.com/mengelbart/rtp-over-quic-draft>.

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 13 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document.

Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Terminology and Notation	3
3. Protocol Overview	4
4. Encapsulation	5
4.1. QUIC Streams	6
4.2. QUIC Datagrams	6
5. RTCP	6
5.1. Transport Layer Feedback	7
5.2. Application Layer Repair and other Control Messages	9
6. Congestion Control	10
6.1. Congestion Control at the QUIC layer	11
6.2. Congestion Control at the Application Layer	11
7. SDP Signalling	12
8. Discussion	14
8.1. Flow Identifier	14
8.2. Impact of Connection Migration	15
9. Security Considerations	15
10. IANA Considerations	15
11. References	15
11.1. Normative References	15
11.2. Informative References	18
Acknowledgments	18
Authors' Addresses	18

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is generally used to carry real-time media for conversational media sessions, such as video conferences, across the Internet. Since RTP requires real-time delivery and is tolerant to packet losses, the default underlying transport protocol has been UDP, recently with DTLS on top to secure the media exchange and occasionally TCP (and possibly TLS) as a fallback. With the advent of QUIC [RFC9000] and, most notably, its unreliable DATAGRAM extension [RFC9221], another secure transport protocol becomes available. QUIC and its DATAGRAMs combine desirable properties for real-time traffic (e.g., no unnecessary retransmissions, avoiding head-of-line blocking) with a secure end-to-end transport that is also expected to work well through NATs and firewalls.

Moreover, with QUIC's multiplexing capabilities, reliable and unreliable transport connections as, e.g., needed for WebRTC, can be established with only a single port used at either end of the connection. This document defines a mapping of how to carry RTP over QUIC. The focus is on RTP and RTCP packet mapping and on reducing the amount of RTCP traffic by leveraging state information readily available within a QUIC endpoint. This document also briefly touches upon how to signal media over QUIC using the Session Description Protocol (SDP) [RFC8866].

The scope of this document is limited to unicast RTP/RTCP.

Note that this draft is similar in spirit to but differs in numerous ways from [I-D.draft-hurst-quic-rtp-tunnelling].

2. Terminology and Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used:

Congestion Controller: QUIC specifies a congestion controller in Section 7 of [RFC9002], but the specific requirements for interactive real-time media lead to the development of dedicated congestion control algorithms. In this document, the term congestion controller refers to these algorithms dedicated to real-time applications.

Datagram: Datagrams exist in UDP as well as in QUICs unreliable datagram extension. If not explicitly noted differently, the term datagram in this document refers to a QUIC Datagram as defined in [RFC9221].

Endpoint: A QUIC server or client that participates in an RTP over QUIC session.

Frame: A QUIC frame as defined in [RFC9000].

Media Encoder: An entity that is used by an application to produce a stream of encoded media, which can be packetized in RTP packets to be transmitted over QUIC.

Receiver: An endpoint that receives media in RTP packets and may send or receive RTCP packets.

Sender: An endpoint that sends media in RTP packets and may send or receive RTCP packets.

Packet diagrams in this document use the format defined in Section 1.3 of [RFC9000] to illustrate the order and size of fields.

3. Protocol Overview

This document introduces a mapping of the Real-time Transport Protocol (RTP) to the QUIC transport protocol. RTP over QUIC allows the use of QUIC streams and unreliable QUIC datagrams to transport real-time data, and thus, the QUIC implementation MUST support QUICs unreliable datagram extension, if RTP packets should be sent over QUIC datagrams. Since datagram frames cannot be fragmented, the QUIC implementation MUST also provide a way to query the maximum datagram size so that an application can create RTP packets that always fit into a QUIC datagram frame.

[RFC3550] specifies that RTP sessions need to be transmitted on different transport addresses to allow multiplexing between them. RTP over QUIC uses a different approach to leverage the advantages of QUIC connections without managing a separate QUIC connection per RTP session. QUIC does not provide demultiplexing between different flows on datagrams but suggests that an application implement a demultiplexing mechanism if required. An example of such a mechanism are flow identifiers prepended to each datagram frame as described in Section 2.1 of [I-D.draft-ietf-masque-h3-datagram]. RTP over QUIC uses a flow identifier to replace the network address and port number to multiplex many RTP sessions over the same QUIC connection.

A congestion controller can be plugged in to adapt the media bitrate to the available bandwidth. This document does not mandate any congestion control algorithm. Some examples include Network-Assisted Dynamic Adaptation (NADA) [RFC8698] and Self-Clocked Rate Adaptation for Multimedia (SCReAM) [RFC8298]. These congestion control algorithms require some feedback about the network's performance to calculate target bitrates. Traditionally this feedback is generated at the receiver and sent back to the sender via RTCP. Since QUIC also collects some metrics about the network's performance, these metrics can be used to generate the required feedback at the sender-side and provide it to the congestion controller to avoid the additional overhead of the RTCP stream.

4. Encapsulation

QUIC supports two transport methods: reliable streams [RFC9000] and unreliable datagrams [RFC9221]. This document specifies a mapping of RTP to both of the transport modes. The encapsulation format for RTP over QUIC is described in Figure 1.

Section 4.1 and Section 4.2 explain the specifics of mapping of RTP to QUIC streams and QUIC datagrams respectively.

```
Payload {  
    Flow Identifier (i),  
    RTP/RTCP Packet (...)  
}
```

Figure 1: RTP over QUIC Payload Format

Flow Identifier: Flow identifier to demultiplex different data flows on the same QUIC connection.

RTP/RTCP Packet: The RTP/RTCP packet to transmit.

For multiplexing different RTP and other data streams on the same QUIC connection, each RTP/RTCP packet is prefixed with a flow identifier. A flow identifier is a QUIC variable-length integer which must be unique per stream.

RTP and RTCP packets of a single RTP session MAY be sent using the same flow identifier (following the procedures defined in [RFC5761]), or they MAY be sent using different flow identifiers. The respective mode of operation MUST be indicated using the appropriate signaling, e.g., when using SDP as discussed in Section 7.

RTP and RTCP packets of different RTP sessions MUST be sent using different flow identifiers.

Differentiating RTP/RTCP packets of different RTP sessions from non-RTP/RTCP datagrams is the responsibility of the application by means of appropriate use of flow identifiers and the corresponding signaling.

4.1. QUIC Streams

An application **MUST** open a new QUIC stream for each Application Data Unit (ADU). Each ADU **MUST** be encapsulated in a single RTP packet and the application **MUST** not send more than one RTP packet per stream. Opening a new stream for each packet adds implicit framing to RTP packets, allows to receive packets without strict ordering and gives an application the possibility to cancel certain packets.

Large RTP packets sent on a stream will be fragmented in smaller QUIC frames, that are transmitted reliably and in order, such that a receiving application can read a complete packet from the stream. No retransmission has to be implemented by the application, since QUIC frames that are lost in transit are retransmitted by the QUIC connection. If it is known to either the sender or the receiver, that a packet, which was not yet successfully and completely transmitted, is no longer needed, either side can close the stream.

**Editor's Note:* We considered adding a framing like the one described in [RFC4571] to send multiple RTP packets on one stream, but we don't think it is worth the additional overhead only to reduce the number of streams. Moreover, putting multiple ADUs into a single stream would also require defining policies when to use the same (and which) stream and when to open a new one.

4.2. QUIC Datagrams

RTP packets can be sent in QUIC datagrams. QUIC datagrams are an extension to QUIC described in [RFC9221]. QUIC datagrams preserve frame boundaries, thus a single RTP packet can be mapped to a single QUIC datagram, without the need for an additional framing. Senders **SHOULD** consider the header overhead associated with QUIC datagrams and ensure that the RTP/RTCP packets, including their payloads, QUIC, and IP headers, will fit into path MTU.

If an application wishes to retransmit lost RTP packets, the retransmission has to be implemented by the application by sending a new datagram for the RTP packet, because QUIC datagrams are not retransmitted on loss (see also Section 5.1 for loss signaling).

5. RTCP

The RTP Control Protocol (RTCP) allows RTP senders and receivers to exchange control information to monitor connection statistics and to identify and synchronize streams. Many of the statistics contained in RTCP packets overlap with the connection statistics collected by a QUIC connection. To avoid using up bandwidth for duplicated control information, the information **SHOULD** only be sent at one protocol

layer. QUIC relies on certain control frames to be sent.

In general, applications MAY send RTCP without any restrictions. This document specifies a baseline for replacing some of the RTCP packet types by mapping the contents to QUIC connection statistics. Future documents can extend this mapping for other RTCP format types. It is RECOMMENDED to expose relevant information from the QUIC layer to the application instead of exchanging additional RTCP packets, where applicable.

This section discusses what information can be exposed from the QUIC connection layer to reduce the RTCP overhead and which type of RTCP messages cannot be replaced by similar feedback from the transport layer. The list of RTCP packets in this section is not exhaustive and similar considerations SHOULD be taken into account before exchanging any other type of RTCP control packets.

TODO: Define parameters for SDP to signal RTCP vs. QUIC feedback. Could use RTCP by default and add parameters for "can use QUIC statistics for X".

5.1. Transport Layer Feedback

This section explains how some of the RTCP packet types which are used to signal reception statistics can be replaced by equivalent statistics that are already collected by QUIC. The following list explains how this mapping can be achieved for the individual fields of different RTCP packet types.

QUIC Datagrams are ack-eliciting packets, which means, that an acknowledgment is triggered when a datagram frame is received. Thus, a sender can assume that an RTP packet arrived at the receiver or was lost in transit, using the QUIC acknowledgments of QUIC Datagram frames. In the following, an RTP packet is regarded as acknowledged, when the QUIC Datagram frame that carried the RTP packet, was acknowledged. For RTP packets that are sent over QUIC streams, an RTP packet can be considered acknowledged, when all frames which carried fragments of the RTP packet were acknowledged.

Some of the transport layer feedback that can be implemented in RTCP contains information that is not included in QUIC by default, but can be added via QUIC extensions. One important example are arrival timestamps, which are not part of QUIC's default acknowledgment frames, but can be added using [I-D.draft-smith-quic-receive-ts] or [I-D.draft-huitema-quic-ts]. Another extension, that can improve the precision of the feedback from QUIC is [I-D.draft-ietf-quic-ack-frequency], which allows a sender to control the delay of acknowledgments sent by the receiver.

- * _Receiver Reports_ (PT=201, Name=RR, [RFC3550])
 - _Fraction lost_: The fraction of lost packets can be directly inferred from QUIC's acknowledgments. The calculation SHOULD include all packets up to the acknowledged RTP packet with the highest RTP sequence number. Later packets SHOULD be ignored, since they may still be in flight, unless other QUIC packets that were sent after the datagram frame, were already acknowledged.
 - _Cumulative lost_: Similar to the fraction of lost packets, the cumulative loss can be inferred from QUIC's acknowledgments including all packets up to the latest acknowledged packet.
 - _Highest Sequence Number received_: The highest sequence number received is the sequence number of all RTP packets that were acknowledged.
 - Interarrival jitter: If QUIC acknowledgments carry timestamps as described in one of the extensions referenced above, senders can infer from QUIC acks the interarrival jitter from the arrival timestamps.
 - Last SR: Similar to RTP arrival times, the arrival time of RTCP Sender Reports can be inferred from QUIC acknowledgments, if they include timestamps.
 - Delay since last SR: This field is not required when the receiver reports are entirely replaced by QUIC feedback.
- * _Negative Acknowledgments_ (PT=205, FMT=1, Name=Generic NACK, [RFC4585])
 - The generic negative acknowledgment packet contains information about packets which the receiver considered lost. Section 6.2.1. of [RFC4585] recommends to use this feature only, if the underlying protocol cannot provide similar feedback. QUIC does not provide negative acknowledgments, but can detect lost packets through acknowledgments.
- * _ECN Feedback_ (PT=205, FMT=8, Name=RTCP-ECN-FB, [RFC6679])

- ECN feedback packets report the count of observed ECN-CE marks. [RFC6679] defines two RTCP reports, one packet type (with PT=205 and FMT=8) and a new report block for the extended reports which are listed below. QUIC supports ECN reporting through acknowledgments. If the connection supports ECN, the reporting of ECN counts SHOULD be done using QUIC acknowledgments.
- * _Congestion Control Feedback_ (PT=205, FMT=11, Name=CCFB, [RFC8888])
 - RTP Congestion Control Feedback contains acknowledgments, arrival timestamps and ECN notifications for each received packet. Acknowledgments and ECNs can be inferred from QUIC as described above. Arrival timestamps can be added through extended acknowledgment frames as described in [I-D.draft-smith-quic-receive-ts] or [I-D.draft-huitema-quic-ts].
- * _Extended Reports_ (PT=207, Name=XR, [RFC3611])
 - Extended Reports offer an extensible framework for a variety of different control messages. Some of the standard report blocks which can be implemented in extended reports such as loss RLE or ECNs can be implemented in QUIC, too. For other report blocks, it SHOULD be evaluated individually, if the contained information can be transmitted using QUIC instead.

5.2. Application Layer Repair and other Control Messages

While the previous section presented some RTCP packet that can be replaced by QUIC features, QUIC cannot replace all of the available RTCP packet types. This mostly affects RTCP packet types which carry control information that is to be interpreted by the application layer instead of the transport itself.

Sender Reports (PT=200, Name=SR, [RFC3550]) are similar to _Receiver Reports_. They are sent by media senders and additionally contain a NTP and a RTP timestamp and the number of packets and octets transmitted by the sender. The timestamps can be used by a receiver to synchronize streams. QUIC cannot provide a similar control information, since it does not know about RTP timestamps. A QUIC receiver can also not calculate the packet or octet counts, since it does not know about lost datagrams. Thus, sender reports are required in RTP over QUIC to synchronize streams at the receiver. The sender reports SHOULD not contain any receiver report blocks, as the information can be inferred from the QUIC transport as explained in the previous section.

Next to carrying transmission statistics, RTCP packets can contain application layer control information, that cannot directly be mapped to QUIC. This includes for example the `_Source Description_` (PT=202, Name=SDES), `_Bye_` (PT=203, Name=BYE) and `_Application_` (PT=204, Name=APP) packet types from [RFC3550] or many of the payload specific feedback messages (PT=206) defined in [RFC4585], which can for example be used to control the codec behavior of the sender. Since QUIC does not provide any kind of application layer control messaging, these RTCP packet types SHOULD be used in the same way as they would be used over any other transport protocol.

6. Congestion Control

Like any other application on the internet, RTP over QUIC needs to perform congestion control to avoid overloading the network.

QUIC is a congestion controlled transport protocol. Senders are required to employ some form of congestion control. The default congestion control specified for QUIC is an algorithm similar to TCP NewReno, but senders are free to choose any congestion control algorithm as long as they follow the guidelines specified in Section 3 of [RFC8085].

RTP does not specify a congestion controller, but provides feedback formats for congestion control (e.g. [RFC8888]) as well as different congestion control algorithms in separate RFCs (e.g. [RFC8298] and [RFC8698]). The congestion control algorithms for RTP are specifically tailored for real-time transmissions at low latencies. The available congestion control algorithms for RTP expose a `target_bitrate` that can be used to dynamically reconfigure media codecs to produce media at a rate that can be sent in real-time under the observed network conditions.

This section defines two architectures for congestion control and bandwidth estimation for RTP over QUIC, but it does not mandate any specific congestion control algorithm to use.

It is assumed that the congestion controller in use provides a pacing mechanism to determine when a packet can be sent to avoid bursts. The currently proposed congestion control algorithms for real-time communications provide such pacing mechanisms. The use of congestion controllers which don't provide a pacing mechanism is out of scope of this document.

TODO: Add considerations for bandwidth shares when a QUIC connection is shared between RTP and non-RTP streams?

6.1. Congestion Control at the QUIC layer

If congestion control is to be applied at the transport layer, it is RECOMMENDED to configure the QUIC Implementation to use a delay-based real-time congestion control algorithm instead of a loss-based algorithm. The currently available delay-based congestion control algorithms depend on detailed arrival time feedback to estimate the current one-way delay between sender and receiver. Since QUIC does not provide arrival timestamps in its acknowledgments, the QUIC implementations of the sender and receiver MUST use an extension to add this information to QUICs acknowledgment frames, e.g. [I-D.draft-smith-quic-receive-ts].

If congestion control is done by the QUIC implementation, the application needs a mechanism to query the currently available bandwidth to adapt media codec configurations. The employed congestion controller of the QUIC connection SHOULD expose such an API to the application. If a current bandwidth estimation is not available from the QUIC congestion controller, the sender can either implement an alternative bandwidth estimation at the application layer as described in Section 6.2 or a receiver can feedback the observed bandwidth through RTCP, e.g., using [I-D.draft-alvestrand-rmcat-remb].

Editor's note: An alternative to the hard requirement to use a timestamp extension could be to use RTCP, but that would mean, that an application has to negotiate RTCP congestion control feedback which would then have to be passed to the QUIC congestion controller.

Editor's note: How can a QUIC connection be shared with non-RTP streams, when SCReAM/NADA/GCC is used as congestion controller? Can these algorithms be adapted to allow different streams including non-real-time streams? Do they even have to be adapted or should this just work?

6.2. Congestion Control at the Application Layer

If an application cannot access a bandwidth estimation from the QUIC layer, or the QUIC implementation does not support a delay-based, low-latency congestion control algorithm, it can alternatively implement a bandwidth estimation algorithm at the application layer. Calculating a bandwidth estimation at the application layer can be done using the same bandwidth estimation algorithms as described in Section 6.1 (NADA, SCReAM). The bandwidth estimation algorithm typically needs some feedback on the transmission performance. This feedback can be collected following the guidelines in Section 5.

If the application implements full congestion control rather than just a bandwidth estimation at the application layer using a congestion controller that satisfies the requirements of Section 7 of [RFC9002], and the connection is only used to send real-time media which is subject to the application layer congestion control, it is RECOMMENDED to disable any other congestion control that is possibly running at the QUIC layer. Disabling the additional congestion controllers helps to avoid any interference between the different congestion controllers.

7. SDP Signalling

Editor's note: See also
[I-D.draft-dawkins-avtcore-sdp-rtp-quic].

QUIC is a connection-based protocol that supports connectionless transmissions of DATAGRAM frames within an established connection. As noted above, demultiplexing DATAGRAMS intended for different purposes is up to the application using QUIC.

There are several necessary steps to carry out jointly between the communicating peers to enable RTP over QUIC:

1. The protocol identifier for the m= lines MUST be "QUIC/RTP", combined as per [RFC8866] with the respective audiovisual profile: for example, "QUIC/RTP/AVP".
2. The peers need to decide whether to establish a new QUIC connection or whether to re-use an existing one. In case of establishing a new connection, the initiator and the responder (client and server) need to be determined. Signaling for this step MUST follow [RFC8122] on SDP attributes for connection-oriented media for the a=setup, a=connection, and a=fingerprint attributes. They MUST use the appropriate protocol identification as per 1.
3. The peers must provide a means for identifying RTP sessions carried in QUIC DATAGRAMS. To enable using a common transport connection for one, two, or more media sessions in the first place, the BUNDLE grouping framework MUST be used [RFC8843]. All media sections belonging to a bundle group, except the first one, MUST set the port in the m= line to zero and MUST include the a=bundle-only attribute.

For disambiguating different RTP session, a reference needs to be provided for each m= line to allow associating this specific media session with a flow identifier. This could be achieved following different approaches:

- * Simply reusing the a=extmap attribute [RFC8285] and relying on RTP header extensions for demultiplexing different media packets carried in QUIC DATAGRAM frames.
- * Defining a variant or different flavor of the a=extmap attribute [RFC8285] that binds media sessions to flow identifiers used in QUIC DATAGRAMS.

Editor's note: It is likely preferable to use multiplexing using QUIC DATAGRAM flow identifiers because this multiplexing mechanisms will also work across RTP and non-RTP media streams.

In either case, the corresponding identifiers MUST be treated independently for each direction of transmission, so that an endpoint MAY choose its own identifies and only uses SDP to inform its peer which RTP sessions use which identifiers.

To this end, SDP MUST be used to indicate the respective flow identifiers for RTP and RTCP of the different RTP sessions (for which we borrow inspiration from [RFC3605]).

4. The peers MUST agree, for each RTP session, whether or not to apply RTP/RTCP multiplexing. If multiplexing RTP and RTCP shall take place on the same flow identifier, this MUST be indicated using the attribute a=rtcp-mux.

A sample session setup offer (liberally borrowed and extended from [RFC8843] and [RFC8122] could look as follows:

```
v=0
o=alice 2890844526 2890844526 IN IP6 2001:db8::3
s=
c=IN IP6 2001:db8::3
t=0 0
a=group:BUNDLE abc xyz

m=audio 10000 QUIC/RTP/AVP 0 8 97
a=setup:actpass
a=connection:new
a=fingerprint:SHA-256 \
  12:DF:3E:5D:49:6B:19:E5:7C:AB:4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF: \
  3E:5D:49:6B:19:E5:7C:AB:4A:AD
b=AS:200
a=mid:abc
a=rtcp-mux
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap:1 urn:ietf:params:<tbid>

m=video 0 QUIC/RTP/AVP 31 32
b=AS:1000
a=bundle-only
a=mid:bar
a=rtcp-mux
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap:2 urn:ietf:params:<tbid>
```

Figure 2: SDP Offer

Signaling details to be worked out.

8. Discussion

8.1. Flow Identifier

[RFC9221] suggests to use flow identifiers to multiplex different streams on QUIC Datagrams, which is implemented in Section 4, but it is unclear how applications can combine RTP over QUIC with other data streams using the same QUIC connections. If the non-RTP data streams use the same flow identifiers, too and the application can make sure, that flow identifiers are unique, there should be no problem. Flow identifiers could be problematic, if different specifications for RTP and non-RTP data streams over QUIC mandate different incompatible flow identifiers.

8.2. Impact of Connection Migration

9. Security Considerations

TBD

10. IANA Considerations

This document has no IANA actions.

11. References

11.1. Normative References

[I-D.draft-alvestrand-rmcat-remb]

Alvestrand, H., "RTCP message for Receiver Estimated Maximum Bitrate", Work in Progress, Internet-Draft, draft-alvestrand-rmcat-remb-03, 21 October 2013, <<https://datatracker.ietf.org/doc/html/draft-alvestrand-rmcat-remb-03>>.

[I-D.draft-huitema-quic-ts]

Huitema, C., "Quic Timestamps For Measuring One-Way Delays", Work in Progress, Internet-Draft, draft-huitema-quic-ts-07, 6 March 2022, <<https://datatracker.ietf.org/doc/html/draft-huitema-quic-ts-07>>.

[I-D.draft-ietf-masque-h3-datagram]

Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", Work in Progress, Internet-Draft, draft-ietf-masque-h3-datagram-09, 11 April 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-masque-h3-datagram-09>>.

[I-D.draft-ietf-quic-ack-frequency]

Iyengar, J. and I. Swett, "QUIC Acknowledgement Frequency", Work in Progress, Internet-Draft, draft-ietf-quic-ack-frequency-01, 25 October 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-01>>.

[I-D.draft-smith-quic-receive-ts]

Smith, C. and I. Swett, "QUIC Extension for Reporting Packet Receive Timestamps", Work in Progress, Internet-Draft, draft-smith-quic-receive-ts-00, 25 October 2021, <<https://datatracker.ietf.org/doc/html/draft-smith-quic-receive-ts-00>>.

- [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>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/rfc/rfc3550>>.
- [RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, DOI 10.17487/RFC3605, October 2003, <<https://www.rfc-editor.org/rfc/rfc3605>>.
- [RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <<https://www.rfc-editor.org/rfc/rfc3611>>.
- [RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, DOI 10.17487/RFC4571, July 2006, <<https://www.rfc-editor.org/rfc/rfc4571>>.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <<https://www.rfc-editor.org/rfc/rfc4585>>.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, DOI 10.17487/RFC5761, April 2010, <<https://www.rfc-editor.org/rfc/rfc5761>>.
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <<https://www.rfc-editor.org/rfc/rfc6679>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/rfc/rfc8085>>.

- [RFC8122] Lennox, J. and C. Holmberg, "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 8122, DOI 10.17487/RFC8122, March 2017, <<https://www.rfc-editor.org/rfc/rfc8122>>.
- [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>>.
- [RFC8285] Singer, D., Desineni, H., and R. Even, Ed., "A General Mechanism for RTP Header Extensions", RFC 8285, DOI 10.17487/RFC8285, October 2017, <<https://www.rfc-editor.org/rfc/rfc8285>>.
- [RFC8298] Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", RFC 8298, DOI 10.17487/RFC8298, December 2017, <<https://www.rfc-editor.org/rfc/rfc8298>>.
- [RFC8698] Zhu, X., Pan, R., Ramalho, M., and S. Mena, "Network-Assisted Dynamic Adaptation (NADA): A Unified Congestion Control Scheme for Real-Time Media", RFC 8698, DOI 10.17487/RFC8698, February 2020, <<https://www.rfc-editor.org/rfc/rfc8698>>.
- [RFC8843] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 8843, DOI 10.17487/RFC8843, January 2021, <<https://www.rfc-editor.org/rfc/rfc8843>>.
- [RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <<https://www.rfc-editor.org/rfc/rfc8866>>.
- [RFC8888] Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", RFC 8888, DOI 10.17487/RFC8888, January 2021, <<https://www.rfc-editor.org/rfc/rfc8888>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/rfc/rfc9000>>.

- [RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <<https://www.rfc-editor.org/rfc/rfc9002>>.
- [RFC9221] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", RFC 9221, DOI 10.17487/RFC9221, March 2022, <<https://www.rfc-editor.org/rfc/rfc9221>>.

11.2. Informative References

- [I-D.draft-dawkins-avtcore-sdp-rtp-quic]
Dawkins, S., "SDP Offer/Answer for RTP using QUIC as Transport", Work in Progress, Internet-Draft, draft-dawkins-avtcore-sdp-rtp-quic-00, 28 January 2022, <<https://datatracker.ietf.org/doc/html/draft-dawkins-avtcore-sdp-rtp-quic-00>>.
- [I-D.draft-hurst-quic-rtp-tunnelling]
Hurst, S., "QRT: QUIC RTP Tunnelling", Work in Progress, Internet-Draft, draft-hurst-quic-rtp-tunnelling-01, 28 January 2021, <<https://datatracker.ietf.org/doc/html/draft-hurst-quic-rtp-tunnelling-01>>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Jörg Ott
Technical University Munich
Email: ott@in.tum.de

Mathis Engelbart
Technical University Munich
Email: mathis.engelbart@gmail.com

AVTCORE
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

J. Uberti
Clubhouse
C. Jennings
Cisco
S. Garcia Murillo
CoSMo
7 March 2022

Completely Encrypting RTP Header Extensions and Contributing Sources
draft-ietf-avtcore-cryptex-05

Abstract

While the Secure Real-time Transport Protocol (SRTP) provides confidentiality for the contents of a media packet, a significant amount of metadata is left unprotected, including RTP header extensions and contributing sources (CSRCs). However, this data can be moderately sensitive in many applications. While there have been previous attempts to protect this data, they have had limited deployment, due to complexity as well as technical limitations.

This document defines Cryptex as a new mechanism that completely encrypts header extensions and CSRCs and uses simpler signaling with the goal of facilitating deployment.

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Problem Statement	3
1.2. Previous Solutions	3
1.3. Goals	4
2. Terminology	5
3. Design	5
4. Signaling	5
5. RTP Header Processing	6
5.1. Sending	6
5.2. Receiving	7
6. Encryption and Decryption	7
6.1. Packet Structure	7
6.2. Encryption Procedure	8
6.3. Decryption Procedure	9
7. Backwards Compatibility	9
8. Security Considerations	9
9. IANA Considerations	10
9.1. cryptex SDP Attribute	10
10. Acknowledgements	11
11. References	11
11.1. Normative References	11
11.2. Informative References	12
Appendix A. Test Vectors	12
A.1. AES-CTR	12
A.1.1. RTP Packet with 1-byte header extension	12
A.1.2. RTP Packet with 2-byte header extension	13
A.1.3. RTP Packet with 1-byte header extension and CSRC fields	14
A.1.4. RTP Packet with 2-byte header extension and CSRC fields	15
A.1.5. RTP Packet with empty 1-byte header extension and CSRC fields	15
A.1.6. RTP Packet with empty 2-byte header extension and CSRC fields	16
A.2. AES-GCM	17
A.2.1. RTP Packet with 1-byte header extension	17
A.2.2. RTP Packet with 2-byte header extension	17

A.2.3.	RTP Packet with 1-byte header extension and CSRC fields	18
A.2.4.	RTP Packet with 2-byte header extension and CSRC fields	19
A.2.5.	RTP Packet with empty 1-byte header extension and CSRC fields	20
A.2.6.	RTP Packet with empty 2-byte header extension and CSRC fields	21
Authors' Addresses	21

1. Introduction

1.1. Problem Statement

The Secure Real-time Transport Protocol [RFC3711] mechanism provides message authentication for the entire RTP packet, but only encrypts the RTP payload. This has not historically been a problem, as much of the information carried in the header has minimal sensitivity (e.g., RTP timestamp); in addition, certain fields need to remain as cleartext because they are used for key scheduling (e.g., RTP SSRC and sequence number).

However, as noted in [RFC6904], the security requirements can be different for information carried in RTP header extensions, including the per-packet sound levels defined in [RFC6464] and [RFC6465], which are specifically noted as being sensitive in the Security Considerations section of those RFCs.

In addition to the contents of the header extensions, there are now enough header extensions in active use that the header extension identifiers themselves can provide meaningful information in terms of determining the identity of the endpoint and/or application. Accordingly, these identifiers can be considered a fingerprinting issue.

Finally, the CSRCs included in RTP packets can also be sensitive, potentially allowing a network eavesdropper to determine who was speaking and when during an otherwise secure conference call.

1.2. Previous Solutions

[RFC6904] was proposed in 2013 as a solution to the problem of unprotected header extension values. However, it has not seen significant adoption, and has a few technical shortcomings.

First, the mechanism is complicated. Since it allows encryption to be negotiated on a per-extension basis, a fair amount of signaling logic is required. And in the SRTP layer, a somewhat complex

transform is required to allow only the selected header extension values to be encrypted. One of the most popular SRTP implementations had a significant bug in this area that was not detected for five years.

Second, it only protects the header extension values, and not their ids or lengths. It also does not protect the CSRCs. As noted above, this leaves a fair amount of potentially sensitive information exposed.

Third, it bloats the header extension space. Because each extension must be offered in both unencrypted and encrypted forms, twice as many header extensions must be offered, which will in many cases push implementations past the 14-extension limit for the use of one-byte extension headers defined in [RFC8285]. Accordingly, implementations will need to use two-byte headers in many cases, which are not supported well by some existing implementations.

Finally, the header extension bloat combined with the need for backwards compatibility results in additional wire overhead. Because two-byte extension headers may not be handled well by existing implementations, one-byte extension identifiers will need to be used for the unencrypted (backwards compatible) forms, and two-byte for the encrypted forms. Thus, deployment of [RFC6904] encryption for header extensions will typically result in multiple extra bytes in each RTP packet, compared to the present situation.

1.3. Goals

From this analysis we can state the desired properties of a solution:

- * Build on existing [RFC3711] SRTP framework (simple to understand)
- * Build on existing [RFC8285] header extension framework (simple to implement)
- * Protection of header extension ids, lengths, and values
- * Protection of CSRCs when present
- * Simple signaling
- * Simple crypto transform and SRTP interactions
- * Backward compatible with unencrypted endpoints, if desired
- * Backward compatible with existing RTP tooling

The last point deserves further discussion. While we considered possible solutions that would have encrypted more of the RTP header (e.g., the number of CSRCs), we felt the inability to parse the resultant packets with current tools, as well as additional complexity incurred, outweighed the slight improvement in confidentiality.

2. Terminology

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.

3. Design

This specification proposes a mechanism to negotiate encryption of all RTP header extensions (ids, lengths, and values) as well as CSRC values. It reuses the existing SRTP framework, is accordingly simple to implement, and is backward compatible with existing RTP packet parsing code, even when support for the mechanism has been negotiated.

4. Signaling

In order to determine whether the mechanism defined in this specification is supported, this document defines a new "a=cryptex" Session Description Protocol (SDP) [RFC4566] attribute to indicate support.

This attribute takes no value, and can be used at the session level or media level.

The presence of this attribute in the SDP (either in an offer or answer) indicates that the endpoint is capable of receiving RTP packets encrypted with Cryptex, as defined below.

Once each peer has verified that the other party supports receiving RTP packets encrypted with Cryptex, senders can unilaterally decide whether to use the Cryptex mechanism or not.

If BUNDLE is in use and the a=cryptex attribute is present for a media line, it MUST be present for all media lines belonging to the same bundle group. This ensures that the encrypted MID header extensions used to demux BUNDLE can be processed correctly. When used with BUNDLE, this attribute is assigned to the TRANSPORT category [RFC8859].

It is possible to signal and negotiate both Encryption of Header Extensions as defined in [RFC6904] and cryptex in the SDP O/A, however if a packet is encrypted with cryptex, it MUST NOT use the [RFC6904] header extension encryption mechanisms.

5. RTP Header Processing

[RFC8285] defines two values for the "defined by profile" field for carrying one-byte and two-byte header extensions. In order to allow a receiver to determine if an incoming RTP packet is using the encryption scheme in this specification, two new values are defined:

- * 0xC0DE for the encrypted version of the one-byte header extensions (instead of 0xBEDE).
- * 0xC2DE for the encrypted versions of the two-byte header extensions (instead of 0x100).

In the case of using two-byte header extensions, the extension id with value 256 MUST NOT be negotiated, as the value of this id is meant to be contained in the "appbits" of the "defined by profile" field, which are not available when using the values above.

If the "a=extmap-allow-mixed" attribute defined in [RFC8285] is negotiated, either one-byte or two-byte header ids can be used (with the values above), as in [RFC8285].

5.1. Sending

When the mechanism defined by this specification has been negotiated, sending a RTP packet that has any CSRCs or contains any {RFC8285} header extensions follows the steps below. This mechanism MUST NOT be used with header extensions other than the [RFC8285] variety.

If the packet contains solely one-byte extension ids, the 16-bit RTP header extension tag MUST be set to 0xC0DE to indicate that the encryption has been applied, and the one-byte framing is being used. If the packet contains only two-byte extension ids, the header extension tag MUST be set to 0xC2DE to indicate encryption has been applied, and the two-byte framing is being used.

If the packet contains CSRCs but no header extensions, an empty extension block consisting of the 0xC0DE tag and a 16-bit length field set to zero (explicitly permitted by [RFC3550]) MUST be appended, and the X bit MUST be set to 1 to indicate an extension block is present. This is necessary to provide the receiver an indication that the CSRCs in the packet are encrypted.

The RTP packet MUST then be encrypted as described in Encryption Procedure.

5.2. Receiving

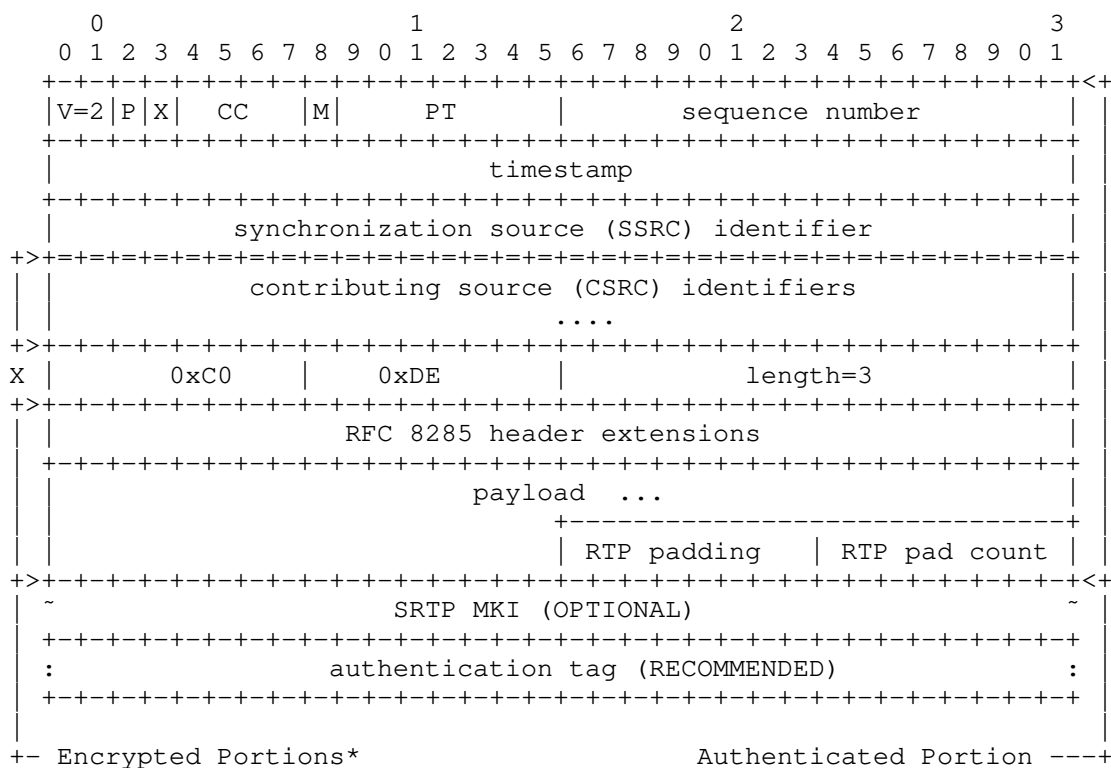
When receiving an RTP packet that contains header extensions, the "defined by profile" field MUST be checked to ensure the payload is formatted according to this specification. If the field does not match one of the values defined above, the implementation MUST instead handle it according to the specification that defines that value. The implementation MAY stop and report an error if it considers use of this specification mandatory for the RTP stream.

If the RTP packet passes this check, it is then decrypted according to Decryption Procedure, and passed to the the next layer to process the packet and its extensions. In the event that a zero-length extension block was added as indicated above, it can be left as-is and will be processed normally.

6. Encryption and Decryption

6.1. Packet Structure

When this mechanism is active, the SRTP packet is protected as follows:



* Note that the 4 bytes at the start of the extension block are not encrypted, as required by [RFC8285].

Specifically, the encrypted portion MUST include any CSRC identifiers, any RTP header extension (except for the first 4 bytes), and the RTP payload.

6.2. Encryption Procedure

The encryption procedure is identical to that of [RFC3711] except for the region to encrypt, which is as shown in the section above.

To minimize changes to surrounding code, the encryption mechanism can choose to replace a "defined by profile" field from [RFC8285] with its counterpart defined in RTP Header Processing above and encrypt at the same time.

For AEAD ciphers (e.g., GCM), the 12-byte fixed header and the four-byte header extension header (the "defined by profile" field and the length) are considered AAD, even though they are non-contiguous in the packet if CSRCs are present.

6.3. Decryption Procedure

The decryption procedure is identical to that of [RFC3711] except for the region to decrypt, which is as shown in the section above.

To minimize changes to surrounding code, the decryption mechanism can choose to replace the "defined by profile" field with its no-encryption counterpart from [RFC8285] and decrypt at the same time.

7. Backwards Compatibility

This specification attempts to encrypt as much as possible without interfering with backwards compatibility for systems that expect a certain structure from an RTPv2 packet, including systems that perform demultiplexing based on packet headers. Accordingly, the first two bytes of the RTP packet are not encrypted.

This specification also attempts to reuse the key scheduling from SRTP, which depends on the RTP packet sequence number and SSRC identifier. Accordingly these values are also not encrypted.

8. Security Considerations

This specification extends SRTP by expanding the portion of the packet that is encrypted, as shown in Packet Structure. It does not change how SRTP authentication works in any way. Given that more of the packet is being encrypted than before, this is necessarily an improvement.

The RTP fields that are left unencrypted (see rationale above) are as follows:

- * RTP version
- * padding bit
- * extension bit
- * number of CSRCs
- * marker bit
- * payload type
- * sequence number
- * timestamp

- * SSRC identifier
- * number of [RFC8285] header extensions

These values contain a fixed set (i.e., one that won't be changed by extensions) of information that, at present, is observed to have low sensitivity. In the event any of these values need to be encrypted, SRTP is likely the wrong protocol to use and a fully-encapsulating protocol such as DTLS is preferred (with its attendant per-packet overhead).

9. IANA Considerations

9.1. cryptex SDP Attribute

This document updates the "Session Description Protocol Parameters" registry as specified in Section 8.2.4 of [RFC8866]. Specifically, it adds the SDP 'cryptex' attribute to the table for SDP media-level attributes.

Contact name: TBD

Contact email address: TBD

Attribute name: cryptex

Attribute syntax: This attribute takes no values.

Attribute semantics: N/A

Attribute value: N/A

Usage level: media-level

Charset dependent: No

Purpose: The presence of this attribute in the SDP indicates that the endpoint is capable of receiving RTP packets encrypted with Cryptex as described in this document. O/A procedures: SDP O/A procedures are described in Section 4 of this document.

Mux Category: TRANSPORT

10. Acknowledgements

The authors wish to thank Lennart Grahl for pointing out many of the issues with the existing header encryption mechanism, as well as suggestions for this proposal. Thanks also to Jonathan Lennox, Inaki Castillo, and Bernard Aboba for their review and suggestions.

11. References

11.1. Normative References

- [BCP14] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.
- <<https://www.rfc-editor.org/info/bcp14>>
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <<https://www.rfc-editor.org/info/rfc3711>>.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <<https://www.rfc-editor.org/info/rfc4566>>.
- [RFC8285] Singer, D., Desineni, H., and R. Even, Ed., "A General Mechanism for RTP Header Extensions", RFC 8285, DOI 10.17487/RFC8285, October 2017, <<https://www.rfc-editor.org/info/rfc8285>>.
- [RFC8859] Nandakumar, S., "A Framework for Session Description Protocol (SDP) Attributes When Multiplexing", RFC 8859, DOI 10.17487/RFC8859, January 2021, <<https://www.rfc-editor.org/info/rfc8859>>.
- [RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <<https://www.rfc-editor.org/info/rfc8866>>.

11.2. Informative References

- [RFC6464] Lennox, J., Ed., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, DOI 10.17487/RFC6464, December 2011, <<https://www.rfc-editor.org/info/rfc6464>>.
- [RFC6465] Ivov, E., Ed., Marocco, E., Ed., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, DOI 10.17487/RFC6465, December 2011, <<https://www.rfc-editor.org/info/rfc6465>>.
- [RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, DOI 10.17487/RFC6904, April 2013, <<https://www.rfc-editor.org/info/rfc6904>>.

Appendix A. Test Vectors

All values are in hexadecimal and represented in network order (big endian).

A.1. AES-CTR

Common values are organized as follows:

Rollover Counter:	00000000
Master Key:	e1f97a0d3e018be0d64fa32c06de4139
Master Salt:	0ec675ad498afeebb6960b3aabe6
Crypto Suite:	AES_CM_128_HMAC_SHA1_80
Session Key:	c61e7a93744f39ee10734afe3ff7a087
Session Salt:	30cbbbc08863d8c85d49db34a9ae1
Authentication Key:	cebe321f6ff7716b6fd4ab49af256a156d38baa4

A.1.1. RTP Packet with 1-byte header extension

RTP Packet:

900f1235
decafbad
cafebabe
bede0001
51000200
abababab
abababab
abababab
abababab

Encrypted RTP Packet:

900f1235
decafbad
cafebabe
c0de0001
eb923652
51c3e036
f8de27e9
c27ee3e0
b4651d9f
bc4218a7
0244522f
34a5

A.1.2. RTP Packet with 2-byte header extension

RTP Packet:

900f1236
decafbad
cafebabe
10000001
05020002
abababab
abababab
abababab
abababab

Encrypted RTP Packet:

```
900f1236
decafbad
cafebabe
c2de0001
4ed9cc4e
6a712b30
96c5ca77
339d4204
ce0d7739
6cab6958
5fbce381
94a5
```

A.1.3. RTP Packet with 1-byte header extension and CSRC fields

RTP Packet:

```
920f1238
decafbad
cafebabe
0001e240
0000b26e
bede0001
51000200
abababab
abababab
abababab
abababab
```

Encrypted RTP Packet:

```
920f1238
decafbad
cafebabe
8bb6e12b
5cff16dd
c0de0001
92838c8c
09e58393
e1de3a9a
74734d67
45671338
c3acf11d
a2df8423
bee0
```


A.1.4. RTP Packet with 2-byte header extension and CSRC fields

RTP Packet:

```
920f1239
decafbad
cafebabe
0001e240
0000b26e
10000001
05020002
abababab
abababab
abababab
abababab
```

Encrypted RTP Packet:

```
920f1239
decafbad
cafebabe
f70e513e
b90b9b25
c2de0001
bbed4848
faa64466
5f3d7f34
125914e9
f4d0ae92
3c6f479b
95a0f7b5
3133
```

A.1.5. RTP Packet with empty 1-byte header extension and CSRC fields

RTP Packet:

```
920f123a
decafbad
cafebabe
0001e240
0000b26e
bede0000
abababab
abababab
abababab
abababab
```

Encrypted RTP Packet:

```
920f123a
decafbad
cafebabe
7130b6ab
fe2ab0e3
c0de0000
e3d9f64b
25c9e74c
b4cf8e43
fb92e378
1c2c0cea
b6b3a499
a14c
```

A.1.6. RTP Packet with empty 2-byte header extension and CSRC fields

RTP Packet:

```
920f123b
decafbad
cafebabe
0001e240
0000b26e
10000000
abababab
abababab
abababab
abababab
```

Encrypted RTP Packet:

```
920f123b
decafbad
cafebabe
cbf24c12
4330e1c8
c2de0000
599dd45b
c9d687b6
03e8b59d
771fd38e
88b170e0
cd31e125
eabe
```

A.2. AES-GCM

Common values are organized as follows:

Rollover Counter:	00000000
Master Key:	000102030405060708090a0b0c0d0e0f
Master Salt:	a0a1a2a3a4a5a6a7a8a9aaab
Crypto Suite:	AEAD_AES_128_GCM
Session Key:	077c6143cb221bc355ff23d5f984a16e
Session Salt:	9af3e95364ebac9c99c5a7c4

A.2.1. RTP Packet with 1-byte header extension

RTP Packet:

900f1235
decafbad
cafebabe
bede0001
51000200
abababab
abababab
abababab
abababab

Encrypted RTP Packet:

900f1235
decafbad
cafebabe
c0de0001
39972dc9
572c4d99
e8fc355d
e743fb2e
94f9d8ff
54e72f41
93bbc5c7
4ffab0fa
9fa0fbeb

A.2.2. RTP Packet with 2-byte header extension

RTP Packet:

900f1236
decafbad
cafebabe
10000001
05020002
abababab
abababab
abababab
abababab

Encrypted RTP Packet:

900f1236
decafbad
cafebabe
c2de0001
bb75a4c5
45cd1f41
3bdb7daa
2b1e3263
de313667
c9632490
81b35a65
f5cb6c88
b394235f

A.2.3. RTP Packet with 1-byte header extension and CSRC fields

RTP Packet:

920f1238
decafbad
cafebabe
0001e240
0000b26e
bede0001
51000200
abababab
abababab
abababab
abababab

Encrypted RTP Packet:

920f1238
decafbad
cafebabe
63bbccc4
a7f695c4
c0de0001
8ad7c71f
ac70a80c
92866b4c
6ba98546
ef913586
e95ffaaf
fe956885
bb0647a8
bc094ac8

A.2.4. RTP Packet with 2-byte header extension and CSRC fields

RTP Packet:

920f1239
decafbad
cafebabe
0001e240
0000b26e
10000001
05020002
abababab
abababab
abababab
abababab

Encrypted RTP Packet:

```
920f1239
decafbad
cafebabe
3680524f
8d312b00
c2de0001
c78d1200
38422bc1
11a7187a
18246f98
0c059cc6
bc9df8b6
26394eca
344e4b05
d80fea83
```

A.2.5. RTP Packet with empty 1-byte header extension and CSRC fields

RTP Packet:

```
920f123a
decafbad
cafebabe
0001e240
0000b26e
bede0000
abababab
abababab
abababab
abababab
```

Encrypted RTP Packet:

```
920f123a
decafbad
cafebabe
15b6bb43
37906fff
c0de0000
b7b96453
7a2b03ab
7ba5389c
e9331712
6b5d974d
f30c6884
dcb651c5
e120c1da
```

A.2.6. RTP Packet with empty 2-byte header extension and CSRC fields

RTP Packet:

```
920f123b
decafbad
cafebabe
0001e240
0000b26e
10000000
abababab
abababab
abababab
abababab
```

Encrypted RTP Packet:

```
920f123b
decafbad
cafebabe
dcb38c9e
48bf95f4
c2de0000
61ee432c
f9203170
76613258
d3ce4236
c06ac429
681ad084
13512dc9
8b5207d8
```

Authors' Addresses

Justin Uberti
Clubhouse
Email: justin@uberti.name

Cullen Jennings
Cisco
Email: fluffy@iii.ca

Sergio Garcia Murillo
CoSMo
Email: sergio.garcia.murillo@cosmosoftware.io

avtcore
Internet-Draft
Intended status: Standards Track
Expires: 6 November 2022

S. Zhao
S. Wenger
Tencent
Y. Sanchez
Fraunhofer HHI
Y. Wang
Bytedance Inc.
M. M Hannuksela
Nokia Technologies
5 May 2022

RTP Payload Format for Versatile Video Coding (VVC)
draft-ietf-avtcore-rtp-vvc-16

Abstract

This memo describes an RTP payload format for the video coding standard ITU-T Recommendation H.266 and ISO/IEC International Standard 23090-3, both also known as Versatile Video Coding (VVC) and developed by the Joint Video Experts Team (JVET). The RTP payload format allows for packetization of one or more Network Abstraction Layer (NAL) units in each RTP packet payload as well as fragmentation of a NAL unit into multiple RTP packets. The payload format has wide applicability in videoconferencing, Internet video streaming, and high-bitrate entertainment-quality video, among other applications.

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 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Overview of the VVC Codec	3
1.1.1. Coding-Tool Features (informative)	3
1.1.2. Systems and Transport Interfaces (informative)	6
1.1.3. High-Level Picture Partitioning (informative)	11
1.1.4. NAL Unit Header	13
1.2. Overview of the Payload Format	14
2. Conventions	15
3. Definitions and Abbreviations	15
3.1. Definitions	15
3.1.1. Definitions from the VVC Specification	15
3.1.2. Definitions Specific to This Memo	18
3.2. Abbreviations	19
4. RTP Payload Format	20
4.1. RTP Header Usage	20
4.2. Payload Header Usage	22
4.3. Payload Structures	22
4.3.1. Single NAL Unit Packets	23
4.3.2. Aggregation Packets (APs)	23
4.3.3. Fragmentation Units	27
4.4. Decoding Order Number	30
5. Packetization Rules	31
6. De-packetization Process	32
7. Payload Format Parameters	34
7.1. Media Type Registration	34
7.2. Optional Parameters Definition	35
7.3. SDP Parameters	45
7.3.1. Mapping of Payload Type Parameters to SDP	46
7.3.2. Usage with SDP Offer/Answer Model	48
7.3.3. Usage in Declarative Session Descriptions	57
7.3.4. Considerations for Parameter Sets	59
8. Use with Feedback Messages	59
8.1. Picture Loss Indication (PLI)	59
8.2. Full Intra Request (FIR)	59
9. Security Considerations	60
10. Congestion Control	61
11. IANA Considerations	62

12. Acknowledgements	62
13. References	62
13.1. Normative References	62
13.2. Informative References	64
Appendix A. Change History	66
Authors' Addresses	66

1. Introduction

The Versatile Video Coding specification was formally published as both ITU-T Recommendation H.266 [VVC] and ISO/IEC International Standard 23090-3 [ISO23090-3]. VVC is reported to provide significant coding efficiency gains over High Efficiency Video Coding [HEVC], also known as H.265, and other earlier video codecs.

This memo specifies an RTP payload format for VVC. It shares its basic design with the NAL (Network Abstraction Layer) unit based RTP payload formats of AVC Video Coding [RFC6184], Scalable Video Coding (SVC) [RFC6190], High Efficiency Video Coding (HEVC) [RFC7798] and their respective predecessors. With respect to design philosophy, security, congestion control, and overall implementation complexity, it has similar properties to those earlier payload format specifications. This is a conscious choice, as at least RFC 6184 is widely deployed and generally known in the relevant implementer communities. Certain scalability-related mechanisms known from [RFC6190] were incorporated into this document, as VVC version 1 supports temporal, spatial, and signal-to-noise ratio (SNR) scalability.

1.1. Overview of the VVC Codec

VVC and HEVC share a similar hybrid video codec design. In this memo, we provide a very brief overview of those features of VVC that are, in some form, addressed by the payload format specified herein. Implementers have to read, understand, and apply the ITU-T/ISO/IEC specifications pertaining to VVC to arrive at interoperable, well-performing implementations.

Conceptually, both VVC and HEVC include a Video Coding Layer (VCL), which is often used to refer to the coding-tool features, and a NAL, which is often used to refer to the systems and transport interface aspects of the codecs.

1.1.1. Coding-Tool Features (informative)

Coding tool features are described below with occasional reference to the coding tool set of HEVC, which is well known in the community.

Similar to earlier hybrid-video-coding-based standards, including HEVC, the following basic video coding design is employed by VVC. A prediction signal is first formed by either intra- or motion-compensated prediction, and the residual (the difference between the original and the prediction) is then coded. The gains in coding efficiency are achieved by redesigning and improving almost all parts of the codec over earlier designs. In addition, VVC includes several tools to make the implementation on parallel architectures easier.

Finally, VVC includes temporal, spatial, and SNR scalability as well as multiview coding support.

Coding blocks and transform structure

Among major coding-tool differences between HEVC and VVC, one of the important improvements is the more flexible coding tree structure in VVC, i.e., multi-type tree. In addition to quadtree, binary and ternary trees are also supported, which contributes significant improvement in coding efficiency. Moreover, the maximum size of a coding tree unit (CTU) is increased from 64x64 to 128x128. To improve the coding efficiency of chroma signal, luma chroma separated trees at CTU level may be employed for intra-slices. The square transforms in HEVC are extended to non-square transforms for rectangular blocks resulting from binary and ternary tree splits. Besides, VVC supports multiple transform sets (MTS), including DCT-2, DST-7, and DCT-8 as well as the non-separable secondary transform. The transforms used in VVC can have different sizes with support for larger transform sizes. For DCT-2, the transform sizes range from 2x2 to 64x64, and for DST-7 and DCT-8, the transform sizes range from 4x4 to 32x32. In addition, VVC also support sub-block transform for both intra and inter coded blocks. For intra coded blocks, intra sub-partitioning (ISP) may be used to allow sub-block based intra prediction and transform. For inter blocks, sub-block transform may be used assuming that only a part of an inter-block has non-zero transform coefficients.

Entropy coding

Similar to HEVC, VVC uses a single entropy-coding engine, which is based on context adaptive binary arithmetic coding [CABAC], but with the support of multi-window sizes. The window sizes can be initialized differently for different context models. Due to such a design, it has more efficient adaptation speed and better coding efficiency. A joint chroma residual coding scheme is applied to further exploit the correlation between the residuals of two color components. In VVC, different residual coding schemes are applied for regular transform coefficients and residual samples generated using transform-skip mode.

In-loop filtering

VVC has more feature support in loop filters than HEVC. The deblocking filter in VVC is similar to HEVC but operates at a smaller grid. After deblocking and sample adaptive offset (SAO), an adaptive loop filter (ALF) may be used. As a Wiener filter, ALF reduces distortion of decoded pictures. Besides, VVC introduces a new module called luma mapping with chroma scaling to fully utilize the dynamic range of signal so that rate-distortion performance of both Standard Dynamic Range (SDR) and High Dynamic Range (HDR) content is improved.

Motion prediction and coding

Compared to HEVC, VVC introduces several improvements in this area. First, there is the adaptive motion vector resolution (AMVR), which can save bit cost for motion vectors by adaptively signaling motion vector resolution. Then the affine motion compensation is included to capture complicated motion like zooming and rotation. Meanwhile, prediction refinement with the optical flow with affine mode (PROF) is further deployed to mimic affine motion at the pixel level. Thirdly the decoder side motion vector refinement (DMVR) is a method to derive MV vector at decoder side based on block matching so that fewer bits may be spent on motion vectors. Bi-directional optical flow (BDOF) is a similar method to PROF. BDOF adds a sample wise offset at 4x4 sub-block level that is derived with equations based on gradients of the prediction samples and a motion difference relative to CU motion vectors. Furthermore, merge with motion vector difference (MMVD) is a special mode, which further signals a limited set of motion vector differences on top of merge mode. In addition to MMVD, there are another three types of special merge modes, i.e., sub-block merge, triangle, and combined intra-/inter-prediction (CIIP). Sub-block merge list includes one candidate of sub-block temporal motion vector prediction (SbTMVP) and up to four candidates of affine motion vectors. Triangle is based on triangular block motion compensation. CIIP combines intra- and inter- predictions with weighting. Adaptive weighting may be employed with a block-level tool called bi-prediction with CU based weighting (BCW) which provides more flexibility than in HEVC.

Intra prediction and intra-coding

To capture the diversified local image texture directions with finer granularity, VVC supports 65 angular directions instead of 33 directions in HEVC. The intra mode coding is based on a 6-most-probable-mode scheme, and the 6 most probable modes are derived using the neighboring intra prediction directions. In addition, to deal with the different distributions of intra prediction angles for different block aspect ratios, a wide-angle intra prediction (WAIP)

scheme is applied in VVC by including intra prediction angles beyond those present in HEVC. Unlike HEVC which only allows using the most adjacent line of reference samples for intra prediction, VVC also allows using two further reference lines, as known as multi-reference-line (MRL) intra prediction. The additional reference lines can be only used for the 6 most probable intra prediction modes. To capture the strong correlation between different colour components, in VVC, a cross-component linear mode (CCLM) is utilized which assumes a linear relationship between the luma sample values and their associated chroma samples. For intra prediction, VVC also applies a position-dependent prediction combination (PDPC) for refining the prediction samples closer to the intra prediction block boundary. Matrix-based intra prediction (MIP) modes are also used in VVC which generates an up to 8x8 intra prediction block using a weighted sum of downsampled neighboring reference samples, and the weights are hardcoded constants.

Other coding-tool features

VVC introduces dependent quantization (DQ) to reduce quantization error by state-based switching between two quantizers.

1.1.2. Systems and Transport Interfaces (informative)

VVC inherits the basic systems and transport interfaces designs from HEVC and AVC. These include the NAL-unit-based syntax structure, the hierarchical syntax and data unit structure, the supplemental enhancement information (SEI) message mechanism, and the video buffering model based on the hypothetical reference decoder (HRD). The scalability features of VVC are conceptually similar to the scalable variant of HEVC known as SHVC. The hierarchical syntax and data unit structure consists of parameter sets at various levels (decoder, sequence (pertaining to all), sequence (pertaining to a single), picture), picture-level header parameters, slice-level header parameters, and lower-level parameters.

A number of key components that influenced the network abstraction layer design of VVC as well as this memo are described below

Decoding capability information

The decoding capability information includes parameters that stay constant for the lifetime of a VVC bitstream, which in IETF terms can translate to a session. Such information includes profile, level, and sub-profile information to determine a maximum capability interop point that is guaranteed to be never exceeded, even if splicing of video sequences occurs within a session. It further includes constraint fields (most of which are flags), which can optionally be

set to indicate that the video bitstream will be constrained in the use of certain features as indicated by the values of those fields. With this, a bitstream can be labeled as not using certain tools, which allows among other things for resource allocation in a decoder implementation.

Video parameter set

The video parameter set (VPS) pertains to one or more coded video sequences (CVSs) of multiple layers covering the same range of access units, and includes, among other information, decoding dependency expressed as information for reference picture list construction of enhancement layers. The VPS provides a "big picture" of a scalable sequence, including what types of operation points are provided, the profile, tier, and level of the operation points, and some other high-level properties of the bitstream that can be used as the basis for session negotiation and content selection, etc. One VPS may be referenced by one or more sequence parameter sets.

Sequence parameter set

The sequence parameter set (SPS) contains syntax elements pertaining to a coded layer video sequence (CLVS), which is a group of pictures belonging to the same layer, starting with a random access point, and followed by pictures that may depend on each other, until the next random access point picture. In MPEG-2, the equivalent of a CVS was a group of pictures (GOP), which normally started with an I frame and was followed by P and B frames. While more complex in its options of random access points, VVC retains this basic concept. One remarkable difference of VVC is that a CLVS may start with a Gradual Decoding Refresh (GDR) picture, without requiring presence of traditional random access points in the bitstream, such as instantaneous decoding refresh (IDR) or clean random access (CRA) pictures. In many TV-like applications, a CVS contains a few hundred milliseconds to a few seconds of video. In video conferencing (without switching MCUs involved), a CVS can be as long in duration as the whole session.

Picture and adaptation parameter set

The picture parameter set and the adaptation parameter set (PPS and APS, respectively) carry information pertaining to zero or more pictures and zero or more slices, respectively. The PPS contains information that is likely to stay constant from picture to picture, at least for pictures for a certain type-whereas the APS contains information, such as adaptive loop filter coefficients, that are likely to change from picture to picture or even within a picture. A single APS is referenced by all slices of the same picture if that

APS contains information about luma mapping with chroma scaling (LMCS) or scaling list. Different APSs containing ALF parameters can be referenced by slices of the same picture.

Picture header

A Picture Header contains information that is common to all slices that belong to the same picture. Being able to send that information as a separate NAL unit when pictures are split into several slices allows for saving bitrate, compared to repeating the same information in all slices. However, there might be scenarios where low-bitrate video is transmitted using a single slice per picture. Having a separate NAL unit to convey that information incurs in an overhead for such scenarios. For such scenarios, the picture header syntax structure is directly included in the slice header, instead of its own NAL unit. The mode of the picture header syntax structure being included in its own NAL unit or not can only be switched on/off for an entire CLVS, and can only be switched off when in the entire CLVS each picture contains only one slice.

Profile, tier, and level

The profile, tier and level syntax structures in DCI, VPS and SPS contain profile, tier, level information for all layers that refer to the DCI, for layers associated with one or more output layer sets specified by the VPS, and for any layer that refers to the SPS, respectively.

Sub-profiles

Within the VVC specification, a sub-profile is a 32-bit number, coded according to ITU-T Rec. T.35, that does not carry a semantics. It is carried in the `profile_tier_level` structure and hence (potentially) present in the DCI, VPS, and SPS. External registration bodies can register a T.35 codepoint with ITU-T registration authorities and associate with their registration a description of bitstream restrictions beyond the profiles defined by ITU-T and ISO/IEC. This would allow encoder manufacturers to label the bitstreams generated by their encoder as complying with such sub-profile. It is expected that upstream standardization organizations (such as: DVB and ATSC), as well as walled-garden video services will take advantage of this labeled system. In contrast to "normal" profiles, it is expected that sub-profiles may indicate encoder choices traditionally left open in the (decoder-centric) video coding specs, such as GOP structures, minimum/maximum QP values, and the mandatory use of certain tools or SEI messages.

General constraint fields

The `profile_tier_level` structure carries a considerable number of constraint fields (most of which are flags), which an encoder can use to indicate to a decoder that it will not use a certain tool or technology. They were included in reaction to a perceived market need for labeled a bitstream as not exercising a certain tool that has become commercially unviable.

Temporal scalability support

VVC includes support of temporal scalability, by inclusion of the signaling of `TemporalId` in the NAL unit header, the restriction that pictures of a particular temporal sublayer cannot be used for inter prediction reference by pictures of a lower temporal sublayer, the sub-bitstream extraction process, and the requirement that each sub-bitstream extraction output be a conforming bitstream. Media-Aware Network Elements (MANEs) can utilize the `TemporalId` in the NAL unit header for stream adaptation purposes based on temporal scalability.

Reference picture resampling (RPR)

In AVC and HEVC, the spatial resolution of pictures cannot change unless a new sequence using a new SPS starts, with an Intra random access point (IRAP) picture. VVC enables picture resolution change within a sequence at a position without encoding an IRAP picture, which is always intra-coded. This feature is sometimes referred to as reference picture resampling (RPR), as the feature needs resampling of a reference picture used for inter prediction when that reference picture has a different resolution than the current picture being decoded. RPR allows resolution change without the need of coding an IRAP picture and hence avoids a momentary bit rate spike caused by an IRAP picture in streaming or video conferencing scenarios, e.g., to cope with network condition changes. RPR can also be used in application scenarios wherein zooming of the entire video region or some region of interest is needed.

Spatial, SNR, and multiview scalability

VVC includes support for spatial, SNR, and multiview scalability. Scalable video coding is widely considered to have technical benefits and enrich services for various video applications. Until recently, however, the functionality has not been included in the first version of specifications of the video codecs. In VVC, however, all those forms of scalability are supported in the first version of VVC natively through the signaling of the `nuh_layer_id` in the NAL unit header, the VPS which associates layers with given `nuh_layer_id` to

each other, reference picture selection, reference picture resampling for spatial scalability, and a number of other mechanisms not relevant for this memo.

Spatial scalability

With the existence of Reference Picture Resampling (RPR), the additional burden for scalability support is just a modification of the high-level syntax (HLS). The inter-layer prediction is employed in a scalable system to improve the coding efficiency of the enhancement layers. In addition to the spatial and temporal motion-compensated predictions that are available in a single-layer codec, the inter-layer prediction in VVC uses the possibly resampled video data of the reconstructed reference picture from a reference layer to predict the current enhancement layer. The resampling process for inter-layer prediction, when used, is performed at the block-level, reusing the existing interpolation process for motion compensation in single-layer coding. It means that no additional resampling process is needed to support spatial scalability.

SNR scalability

SNR scalability is similar to spatial scalability except that the resampling factors are 1:1. In other words, there is no change in resolution, but there is inter-layer prediction.

Multiview scalability

The first version of VVC also supports multiview scalability, wherein a multi-layer bitstream carries layers representing multiple views, and one or more of the represented views can be output at the same time.

SEI messages

Supplemental enhancement information (SEI) messages are information in the bitstream that do not influence the decoding process as specified in the VVC spec, but address issues of representation/rendering of the decoded bitstream, label the bitstream for certain applications, among other, similar tasks. The overall concept of SEI messages and many of the messages themselves has been inherited from the AVC and HEVC specs. Except for the SEI messages that affect the specification of the hypothetical reference decoder (HRD), other SEI messages for use in the VVC environment, which are generally useful also in other video coding technologies, are not included in the main VVC specification but in a companion specification [VSEI].

1.1.1.3. High-Level Picture Partitioning (informative)

VVC inherited the concept of tiles and wavefront parallel processing (WPP) from HEVC, with some minor to moderate differences. The basic concept of slices was kept in VVC but designed in an essentially different form. VVC is the first video coding standard that includes subpictures as a feature, which provides the same functionality as HEVC motion-constrained tile sets (MCTSs) but designed differently to have better coding efficiency and to be friendlier for usage in application systems. More details of these differences are described below.

Tiles and WPP

Same as in HEVC, a picture can be split into tile rows and tile columns in VVC, in-picture prediction across tile boundaries is disallowed, etc. However, the syntax for signaling of tile partitioning has been simplified, by using a unified syntax design for both the uniform and the non-uniform mode. In addition, signaling of entry point offsets for tiles in the slice header is optional in VVC while it is mandatory in HEVC. The WPP design in VVC has two differences compared to HEVC: i) The CTU row delay is reduced from two CTUs to one CTU; ii) signaling of entry point offsets for WPP in the slice header is optional in VVC while it is mandatory in HEVC.

Slices

In VVC, the conventional slices based on CTUs (as in HEVC) or macroblocks (as in AVC) have been removed. The main reasoning behind this architectural change is as follows. The advances in video coding since 2003 (the publication year of AVC v1) have been such that slice-based error concealment has become practically impossible, due to the ever-increasing number and efficiency of in-picture and inter-picture prediction mechanisms. An error-concealed picture is the decoding result of a transmitted coded picture for which there is some data loss (e.g., loss of some slices) of the coded picture or a reference picture for at least some part of the coded picture is not error-free (e.g., that reference picture was an error-concealed picture). For example, when one of the multiple slices of a picture is lost, it may be error-concealed using an interpolation of the neighboring slices. While advanced video coding prediction mechanisms provide significantly higher coding efficiency, they also make it harder for machines to estimate the quality of an error-concealed picture, which was already a hard problem with the use of simpler prediction mechanisms. Advanced in-picture prediction mechanisms also cause the coding efficiency loss due to splitting a picture into multiple slices to be more significant. Furthermore,

network conditions become significantly better while at the same time techniques for dealing with packet losses have become significantly improved. As a result, very few implementations have recently used slices for maximum transmission unit size matching. Instead, substantially all applications where low-delay error resilience is required (e.g., video telephony and video conferencing) rely on system/transport-level error resilience (e.g., retransmission, forward error correction) and/or picture-based error resilience tools (feedback-based error resilience, insertion of IRAPs, scalability with higher protection level of the base layer, and so on). Considering all the above, nowadays it is very rare that a picture that cannot be correctly decoded is passed to the decoder, and when such a rare case occurs, the system can afford to wait for an error-free picture to be decoded and available for display without resulting in frequent and long periods of picture freezing seen by end users.

Slices in VVC have two modes: rectangular slices and raster-scan slices. The rectangular slice, as indicated by its name, covers a rectangular region of the picture. Typically, a rectangular slice consists of several complete tiles. However, it is also possible that a rectangular slice is a subset of a tile and consists of one or more consecutive, complete CTU rows within a tile. A raster-scan slice consists of one or more complete tiles in a tile raster scan order, hence the region covered by a raster-scan slices need not but could have a non-rectangular shape, but it may also happen to have the shape of a rectangle. The concept of slices in VVC is therefore strongly linked to or based on tiles instead of CTUs (as in HEVC) or macroblocks (as in AVC).

Subpictures

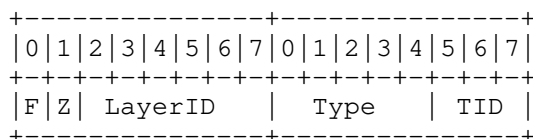
VVC is the first video coding standard that includes the support of subpictures as a feature. Each subpicture consists of one or more complete rectangular slices that collectively cover a rectangular region of the picture. A subpicture may be either specified to be extractable (i.e., coded independently of other subpictures of the same picture and of earlier pictures in decoding order) or not extractable. Regardless of whether a subpicture is extractable or not, the encoder can control whether in-loop filtering (including deblocking, SAO, and ALF) is applied across the subpicture boundaries individually for each subpicture.

Functionally, subpictures are similar to the motion-constrained tile sets (MCTSs) in HEVC. They both allow independent coding and extraction of a rectangular subset of a sequence of coded pictures, for use cases like viewport-dependent 360o video streaming optimization and region of interest (ROI) applications.

There are several important design differences between subpictures and MCTSs. First, the subpictures feature in VVC allows motion vectors of a coding block pointing outside of the subpicture even when the subpicture is extractable by applying sample padding at subpicture boundaries in this case, similarly as at picture boundaries. Second, additional changes were introduced for the selection and derivation of motion vectors in the merge mode and in the decoder side motion vector refinement process of VVC. This allows higher coding efficiency compared to the non-normative motion constraints applied at the encoder-side for MCTSs. Third, rewriting of SHs (and PH NAL units, when present) is not needed when extracting one or more extractable subpictures from a sequence of pictures to create a sub-bitstream that is a conforming bitstream. In sub-bitstream extractions based on HEVC MCTSs, rewriting of SHs is needed. Note that in both HEVC MCTSs extraction and VVC subpictures extraction, rewriting of SPSSs and PPSSs is needed. However, typically there are only a few parameter sets in a bitstream, while each picture has at least one slice, therefore rewriting of SHs can be a significant burden for application systems. Fourth, slices of different subpictures within a picture are allowed to have different NAL unit types. Fifth, VVC specifies HRD and level definitions for subpicture sequences, thus the conformance of the sub-bitstream of each extractable subpicture sequence can be ensured by encoders.

1.1.4. NAL Unit Header

VVC maintains the NAL unit concept of HEVC with modifications. VVC uses a two-byte NAL unit header, as shown in Figure 1. The payload of a NAL unit refers to the NAL unit excluding the NAL unit header.



The Structure of the VVC NAL Unit Header.

Figure 1

The semantics of the fields in the NAL unit header are as specified in VVC and described briefly below for convenience. In addition to the name and size of each field, the corresponding syntax element name in VVC is also provided.

F: 1 bit

`forbidden_zero_bit`. Required to be zero in VVC. Note that the inclusion of this bit in the NAL unit header was to enable transport of VVC video over MPEG-2 transport systems (avoidance of start code emulations) [MPEG2S]. In the context of this memo the value 1 may be used to indicate a syntax violation, e.g., for a NAL unit resulted from aggregating a number of fragmented units of a NAL unit but missing the last fragment, as described in the last sentence of section 4.3.3.

Z: 1 bit

`nuh_reserved_zero_bit`. Required to be zero in VVC, and reserved for future extensions by ITU-T and ISO/IEC. This memo does not overload the "Z" bit for local extensions, as a) overloading the "F" bit is sufficient and b) to preserve the usefulness of this memo to possible future versions of [VVC].

LayerId: 6 bits

`nuh_layer_id`. Identifies the layer a NAL unit belongs to, wherein a layer may be, e.g., a spatial scalable layer, a quality scalable layer, a layer containing a different view, etc.

Type: 5 bits

`nal_unit_type`. This field specifies the NAL unit type as defined in Table 5 of [VVC]. For a reference of all currently defined NAL unit types and their semantics, please refer to Section 7.4.2.2 in [VVC].

TID: 3 bits

`nuh_temporal_id_plus1`. This field specifies the temporal identifier of the NAL unit plus 1. The value of TemporalId is equal to TID minus 1. A TID value of 0 is illegal to ensure that there is at least one bit in the NAL unit header equal to 1, so to enable the consideration of start code emulations in the NAL unit payload data independent of the NAL unit header.

1.2. Overview of the Payload Format

This payload format defines the following processes required for transport of VVC coded data over RTP [RFC3550]:

- * Usage of RTP header with this payload format

- * Packetization of VVC coded NAL units into RTP packets using three types of payload structures: a single NAL unit packet, aggregation packet, and fragment unit
- * Transmission of VVC NAL units of the same bitstream within a single RTP stream
- * Media type parameters to be used with the Session Description Protocol (SDP) [RFC8866]
- * Usage of RTCP feedback messages

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Definitions and Abbreviations

3.1. Definitions

This document uses the terms and definitions of VVC. Section 3.1.1 lists relevant definitions from [VVC] for convenience. Section 3.1.2 provides definitions specific to this memo. All the used terms and definitions in this memo are verbatim copies of [VVC] specification.

3.1.1. Definitions from the VVC Specification

Access unit (AU): A set of PUs that belong to different layers and contain coded pictures associated with the same time for output from the DPB.

Adaptation parameter set (APS): A syntax structure containing syntax elements that apply to zero or more slices as determined by zero or more syntax elements found in slice headers.

Bitstream: A sequence of bits, in the form of a NAL unit stream or a byte stream, that forms the representation of a sequence of AUs forming one or more coded video sequences (CVSs).

Coded picture: A coded representation of a picture comprising VCL NAL units with a particular value of `nuh_layer_id` within an AU and containing all CTUs of the picture.

Clean random access (CRA) PU: A PU in which the coded picture is a CRA picture.

Clean random access (CRA) picture: An IRAP picture for which each VCL NAL unit has nal_unit_type equal to CRA_NUT.

Coded video sequence (CVS): A sequence of AUs that consists, in decoding order, of a CVSS AU, followed by zero or more AUs that are not CVSS AUs, including all subsequent AUs up to but not including any subsequent AU that is a CVSS AU.

Coded video sequence start (CVSS) AU: An AU in which there is a PU for each layer in the CVS and the coded picture in each PU is a CLVSS picture.

Coded layer video sequence (CLVS): A sequence of PUs with the same value of nuh_layer_id that consists, in decoding order, of a CLVSS PU, followed by zero or more PUs that are not CLVSS PUs, including all subsequent PUs up to but not including any subsequent PU that is a CLVSS PU.

Coded layer video sequence start (CLVSS) PU: A PU in which the coded picture is a CLVSS picture.

Coded layer video sequence start (CLVSS) picture: A coded picture that is an IRAP picture with NoOutputBeforeRecoveryFlag equal to 1 or a GDR picture with NoOutputBeforeRecoveryFlag equal to 1.

Coding tree unit (CTU): A CTB of luma samples, two corresponding CTBs of chroma samples of a picture that has three sample arrays, or a CTB of samples of a monochrome picture or a picture that is coded using three separate colour planes and syntax structures used to code the samples.

Decoding Capability Information (DCI): A syntax structure containing syntax elements that apply to the entire bitstream.

Decoded picture buffer (DPB): A buffer holding decoded pictures for reference, output reordering, or output delay specified for the hypothetical reference decoder.

Gradual decoding refresh (GDR) picture: A picture for which each VCL NAL unit has nal_unit_type equal to GDR_NUT.

Instantaneous decoding refresh (IDR) PU: A PU in which the coded picture is an IDR picture.

Instantaneous decoding refresh (IDR) picture: An IRAP picture for which each VCL NAL unit has `nal_unit_type` equal to `IDR_W_RADL` or `IDR_N_LP`.

Intra random access point (IRAP) AU: An AU in which there is a PU for each layer in the CVS and the coded picture in each PU is an IRAP picture.

Intra random access point (IRAP) PU: A PU in which the coded picture is an IRAP picture.

Intra random access point (IRAP) picture: A coded picture for which all VCL NAL units have the same value of `nal_unit_type` in the range of `IDR_W_RADL` to `CRA_NUT`, inclusive.

Layer: A set of VCL NAL units that all have a particular value of `nuh_layer_id` and the associated non-VCL NAL units.

Network abstraction layer (NAL) unit: A syntax structure containing an indication of the type of data to follow and bytes containing that data in the form of an RBSP interspersed as necessary with emulation prevention bytes.

Network abstraction layer (NAL) unit stream: A sequence of NAL units.

Output Layer Set (OLS): A set of layers for which one or more layers are specified as the output layers.

Operation point (OP): A temporal subset of an OLS, identified by an OLS index and a highest value of `TemporalId`.

Picture parameter set (PPS): A syntax structure containing syntax elements that apply to zero or more entire coded pictures as determined by a syntax element found in each slice header.

Picture unit (PU): A set of NAL units that are associated with each other according to a specified classification rule, are consecutive in decoding order, and contain exactly one coded picture.

Random access: The act of starting the decoding process for a bitstream at a point other than the beginning of the stream.

Sequence parameter set (SPS): A syntax structure containing syntax elements that apply to zero or more entire CLVSs as determined by the content of a syntax element found in the PPS referred to by a syntax element found in each picture header.

Slice: An integer number of complete tiles or an integer number of consecutive complete CTU rows within a tile of a picture that are exclusively contained in a single NAL unit.

Slice header (SH): A part of a coded slice containing the data elements pertaining to all tiles or CTU rows within a tile represented in the slice.

Sublayer: A temporal scalable layer of a temporal scalable bitstream consisting of VCL NAL units with a particular value of the TemporalId variable, and the associated non-VCL NAL units.

Subpicture: A rectangular region of one or more slices within a picture.

Sublayer representation: A subset of the bitstream consisting of NAL units of a particular sublayer and the lower sublayers.

Tile: A rectangular region of CTUs within a particular tile column and a particular tile row in a picture.

Tile column: A rectangular region of CTUs having a height equal to the height of the picture and a width specified by syntax elements in the picture parameter set.

Tile row: A rectangular region of CTUs having a height specified by syntax elements in the picture parameter set and a width equal to the width of the picture.

Video coding layer (VCL) NAL unit: A collective term for coded slice NAL units and the subset of NAL units that have reserved values of nal_unit_type that are classified as VCL NAL units in this Specification.

3.1.2. Definitions Specific to This Memo

Media-Aware Network Element (MANE): A network element, such as a middlebox, selective forwarding unit, or application-layer gateway that is capable of parsing certain aspects of the RTP payload headers or the RTP payload and reacting to their contents.

Informative note: The concept of a MANE goes beyond normal routers or gateways in that a MANE has to be aware of the signaling (e.g., to learn about the payload type mappings of the media streams), and in that it has to be trusted when working with Secure RTP (SRTP). The advantage of using MANEs is that they allow packets to be dropped according to the needs of the media coding. For example, if a MANE has to drop packets due to congestion on a

certain link, it can identify and remove those packets whose elimination produces the least adverse effect on the user experience. After dropping packets, MANEs must rewrite RTCP packets to match the changes to the RTP stream, as specified in Section 7 of [RFC3550].

NAL unit decoding order: A NAL unit order that conforms to the constraints on NAL unit order given in Section 7.4.2.4 in [VVC], follow the Order of NAL units in the bitstream.

RTP stream (See [RFC7656]): Within the scope of this memo, one RTP stream is utilized to transport a VVC bitstream, which may contain one or more layers, and each layer may contain one or more temporal sublayers.

Transmission order: The order of packets in ascending RTP sequence number order (in modulo arithmetic). Within an aggregation packet, the NAL unit transmission order is the same as the order of appearance of NAL units in the packet.

3.2. Abbreviations

AU	Access Unit
AP	Aggregation Packet
APS	Adaptation Parameter Set
CTU	Coding Tree Unit
CVS	Coded Video Sequence
DPB	Decoded Picture Buffer
DCI	Decoding Capability Information
DON	Decoding Order Number
FIR	Full Intra Request
FU	Fragmentation Unit
GDR	Gradual Decoding Refresh
HRD	Hypothetical Reference Decoder
IDR	Instantaneous Decoding Refresh

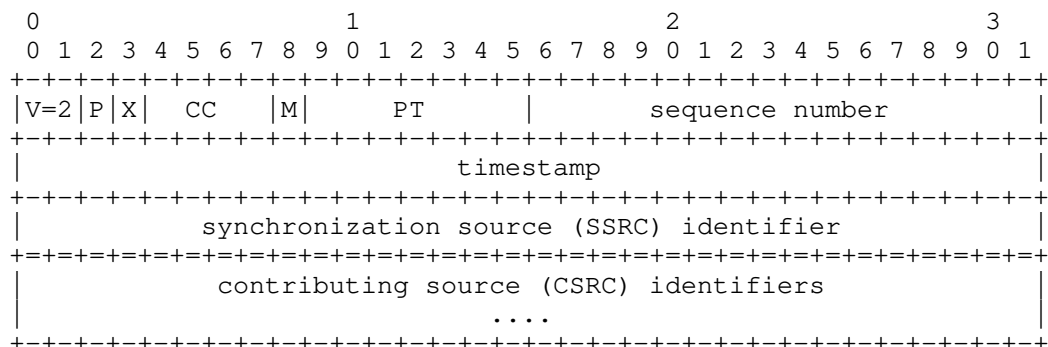
IRAP	Intra Random Access Point
MANE	Media-Aware Network Element
MTU	Maximum Transfer Unit
NAL	Network Abstraction Layer
NALU	Network Abstraction Layer Unit
OLS	Output Layer Set
PLI	Picture Loss Indication
PPS	Picture Parameter Set
RPSI	Reference Picture Selection Indication
SEI	Supplemental Enhancement Information
SLI	Slice Loss Indication
SPS	Sequence Parameter Set
VCL	Video Coding Layer
VPS	Video Parameter Set

4. RTP Payload Format

4.1. RTP Header Usage

The format of the RTP header is specified in [RFC3550] (reprinted as Figure 2 for convenience). This payload format uses the fields of the header in a manner consistent with that specification.

The RTP payload (and the settings for some RTP header bits) for aggregation packets and fragmentation units are specified in Section 4.3.2 and Section 4.3.3, respectively.



RTP Header According to [RFC3550]

Figure 2

The RTP header information to be set according to this RTP payload format is set as follows:

Marker bit (M): 1 bit

Set for the last packet, in transmission order, among each set of packets that contain NAL units of one access unit. This is in line with the normal use of the M bit in video formats to allow an efficient playout buffer handling.

Payload Type (PT): 7 bits

The assignment of an RTP payload type for this new packet format is outside the scope of this document and will not be specified here. The assignment of a payload type has to be performed either through the profile used or in a dynamic way.

Sequence Number (SN): 16 bits

Set and used in accordance with [RFC3550].

Timestamp: 32 bits

The RTP timestamp is set to the sampling timestamp of the content. A 90 kHz clock rate MUST be used. If the NAL unit has no timing properties of its own (e.g., parameter set and SEI NAL units), the RTP timestamp MUST be set to the RTP timestamp of the coded pictures of the access unit in which the NAL unit (according to Section 7.4.2.4 of [VVC]) is included. Receivers MUST use the RTP timestamp for the display process, even when the bitstream contains picture timing SEI messages or decoding unit information SEI messages as specified in [VVC].

Informative note: When picture timing SEI messages are present, the RTP sender is responsible to ensure that the RTP timestamps are consistent with the timing information carried in the picture timing SEI messages.

Synchronization source (SSRC): 32 bits

Used to identify the source of the RTP packets. A single SSRC is used for all parts of a single bitstream.

4.2. Payload Header Usage

The first two bytes of the payload of an RTP packet are referred to as the payload header. The payload header consists of the same fields (F, Z, LayerId, Type, and TID) as the NAL unit header as shown in Section 1.1.4, irrespective of the type of the payload structure.

The TID value indicates (among other things) the relative importance of an RTP packet, for example, because NAL units belonging to higher temporal sublayers are not used for the decoding of lower temporal sublayers. A lower value of TID indicates a higher importance. More-important NAL units MAY be better protected against transmission losses than less-important NAL units.

4.3. Payload Structures

Three different types of RTP packet payload structures are specified. A receiver can identify the type of an RTP packet payload through the Type field in the payload header.

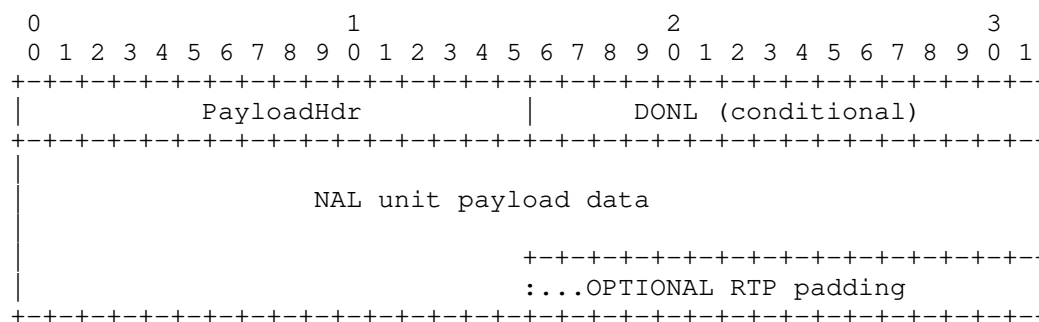
The three different payload structures are as follows:

- * Single NAL unit packet: Contains a single NAL unit in the payload, and the NAL unit header of the NAL unit also serves as the payload header. This payload structure is specified in Section 4.4.1.

- * Aggregation Packet (AP): Contains more than one NAL unit within one access unit. This payload structure is specified in Section 4.3.2.
- * Fragmentation Unit (FU): Contains a subset of a single NAL unit. This payload structure is specified in Section 4.3.3.

4.3.1. Single NAL Unit Packets

A single NAL unit packet contains exactly one NAL unit, and consists of a payload header (denoted as PayloadHdr), a conditional 16-bit DONL field (in network byte order), and the NAL unit payload data (the NAL unit excluding its NAL unit header) of the contained NAL unit, as shown in Figure 3.



The Structure of a Single NAL Unit Packet

Figure 3

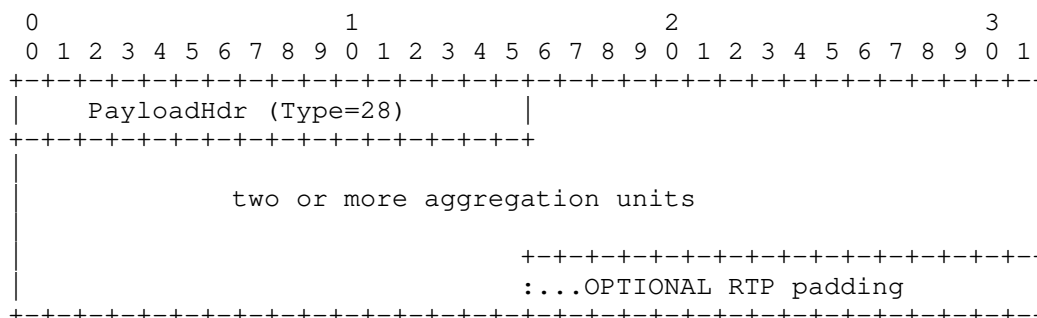
The DONL field, when present, specifies the value of the 16 least significant bits of the decoding order number of the contained NAL unit. If sprop-max-don-diff is greater than 0, the DONL field **MUST** be present, and the variable DON for the contained NAL unit is derived as equal to the value of the DONL field. Otherwise (sprop-max-don-diff is equal to 0), the DONL field **MUST NOT** be present.

4.3.2. Aggregation Packets (APs)

Aggregation Packets (APs) can reduce packetization overhead for small NAL units, such as most of the non-VCL NAL units, which are often only a few octets in size.

An AP aggregates NAL units of one access unit and it MUST NOT contain NAL units from more than one AU. Each NAL unit to be carried in an AP is encapsulated in an aggregation unit. NAL units aggregated in one AP are included in NAL unit decoding order.

An AP consists of a payload header (denoted as PayloadHdr) followed by two or more aggregation units, as shown in Figure 4.



The Structure of an Aggregation Packet

Figure 4

The fields in the payload header of an AP are set as follows. The F bit MUST be equal to 0 if the F bit of each aggregated NAL unit is equal to zero; otherwise, it MUST be equal to 1. The Type field MUST be equal to 28.

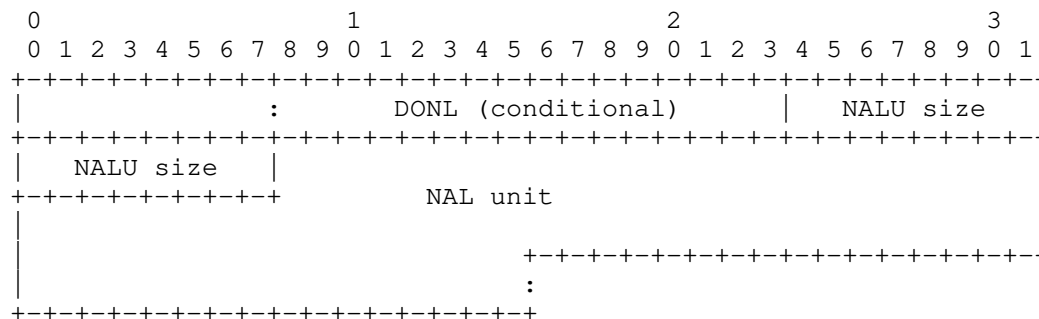
The value of LayerId MUST be equal to the lowest value of LayerId of all the aggregated NAL units. The value of TID MUST be the lowest value of TID of all the aggregated NAL units.

Informative note: All VCL NAL units in an AP have the same TID value since they belong to the same access unit. However, an AP may contain non-VCL NAL units for which the TID value in the NAL unit header may be different than the TID value of the VCL NAL units in the same AP.

Informative Note: If a system envisions sub-picture level or picture level modifications, for example by removing sub-pictures or pictures of a particular layer, a good design choice on the sender's side would be to aggregate NAL units belonging to only the same sub-picture or picture of a particular layer.

An AP MUST carry at least two aggregation units and can carry as many aggregation units as necessary; however, the total amount of data in an AP obviously MUST fit into an IP packet, and the size SHOULD be chosen so that the resulting IP packet is smaller than the MTU size so to avoid IP layer fragmentation. An AP MUST NOT contain FUs specified in Section 4.3.3. APs MUST NOT be nested; i.e., an AP can not contain another AP.

The first aggregation unit in an AP consists of a conditional 16-bit DONL field (in network byte order) followed by a 16-bit unsigned size information (in network byte order) that indicates the size of the NAL unit in bytes (excluding these two octets, but including the NAL unit header), followed by the NAL unit itself, including its NAL unit header, as shown in Figure 5.



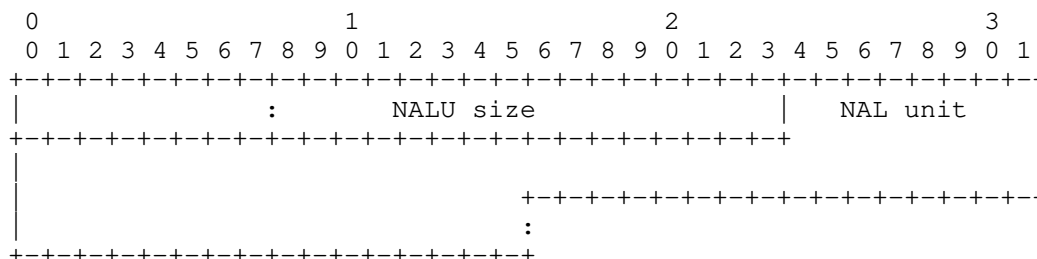
The Structure of the First Aggregation Unit in an AP

Figure 5

The DONL field, when present, specifies the value of the 16 least significant bits of the decoding order number of the aggregated NAL unit.

If `sprop-max-don-diff` is greater than 0, the DONL field MUST be present in an aggregation unit that is the first aggregation unit in an AP, and the variable `DON` for the aggregated NAL unit is derived as equal to the value of the DONL field, and the variable `DON` for an aggregation unit that is not the first aggregation unit in an AP aggregated NAL unit is derived as equal to the `DON` of the preceding aggregated NAL unit in the same AP plus 1 modulo 65536. Otherwise (`sprop-max-don-diff` is equal to 0), the DONL field MUST NOT be present in an aggregation unit that is the first aggregation unit in an AP.

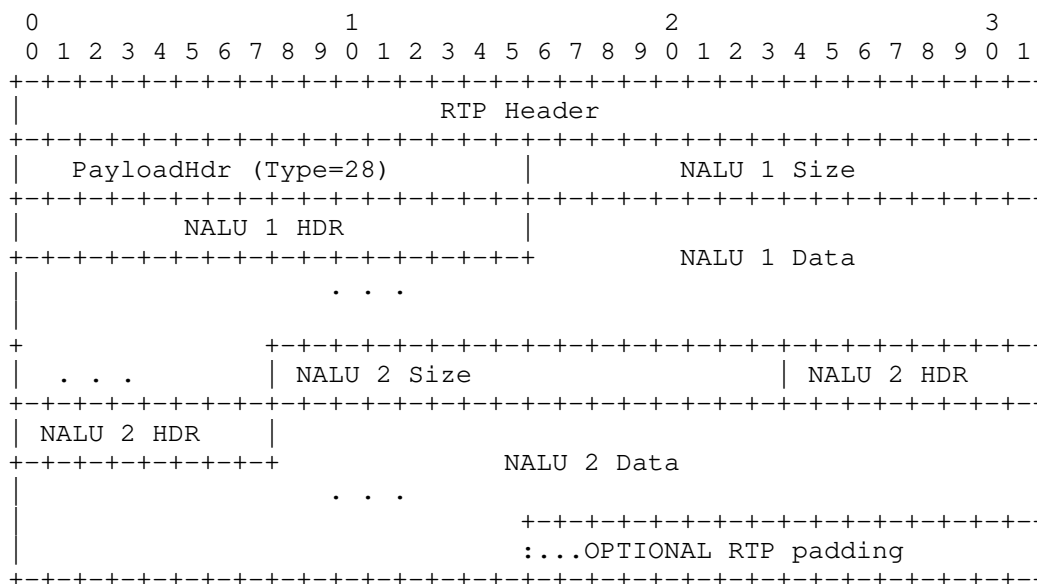
An aggregation unit that is not the first aggregation unit in an AP will be followed immediately by a 16-bit unsigned size information (in network byte order) that indicates the size of the NAL unit in bytes (excluding these two octets, but including the NAL unit header), followed by the NAL unit itself, including its NAL unit header, as shown in Figure 6.



The Structure of an Aggregation Unit That Is Not the First Aggregation Unit in an AP

Figure 6

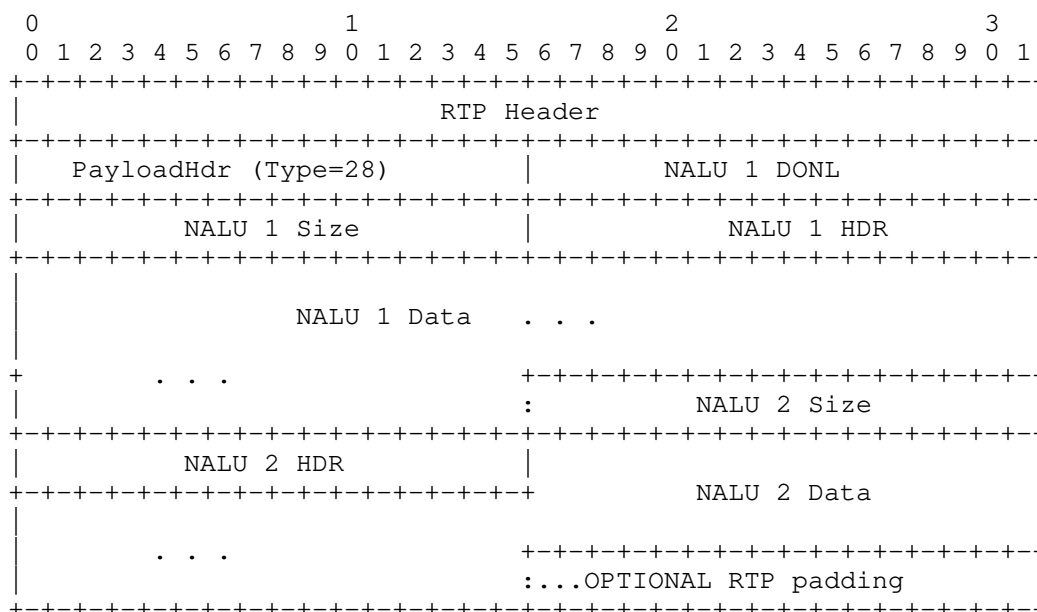
Figure 7 presents an example of an AP that contains two aggregation units, labeled as 1 and 2 in the figure, without the DONL field being present.



An Example of an AP Packet Containing Two Aggregation Units without the DONL Field

Figure 7

Figure 8 presents an example of an AP that contains two aggregation units, labeled as 1 and 2 in the figure, with the DONL field being present.



An Example of an AP Containing Two Aggregation Units with the DONL Field

Figure 8

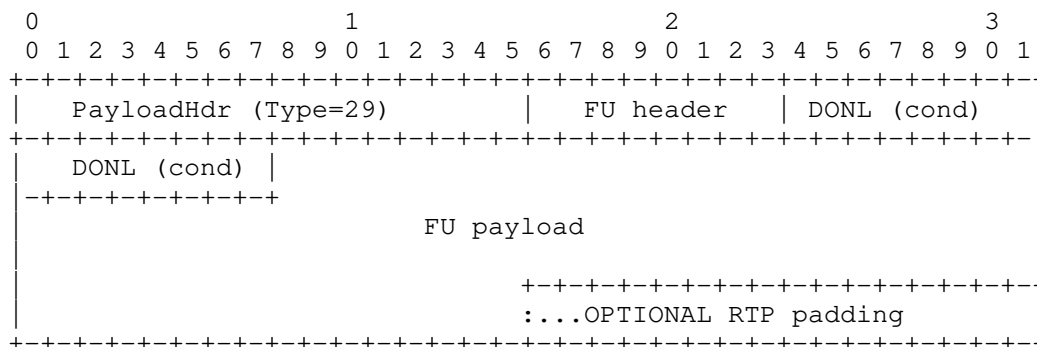
4.3.3. Fragmentation Units

Fragmentation Units (FUs) are introduced to enable fragmenting a single NAL unit into multiple RTP packets, possibly without cooperation or knowledge of the [VVC] encoder. A fragment of a NAL unit consists of an integer number of consecutive octets of that NAL unit. Fragments of the same NAL unit **MUST** be sent in consecutive order with ascending RTP sequence numbers (with no other RTP packets within the same RTP stream being sent between the first and last fragment).

When a NAL unit is fragmented and conveyed within FUs, it is referred to as a fragmented NAL unit. APs MUST NOT be fragmented. FUs MUST NOT be nested; i.e., an FU can not contain a subset of another FU.

The RTP timestamp of an RTP packet carrying an FU is set to the NALU-time of the fragmented NAL unit.

An FU consists of a payload header (denoted as PayloadHdr), an FU header of one octet, a conditional 16-bit DONL field (in network byte order), and an FU payload, as shown in Figure 9.

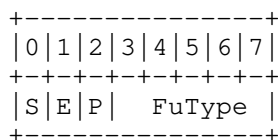


The Structure of an FU

Figure 9

The fields in the payload header are set as follows. The Type field MUST be equal to 29. The fields F, LayerId, and TID MUST be equal to the fields F, LayerId, and TID, respectively, of the fragmented NAL unit.

The FU header consists of an S bit, an E bit, an R bit and a 5-bit FuType field, as shown in Figure 10.



The Structure of FU Header

Figure 10

The semantics of the FU header fields are as follows:

S: 1 bit

When set to 1, the S bit indicates the start of a fragmented NAL unit, i.e., the first byte of the FU payload is also the first byte of the payload of the fragmented NAL unit. When the FU payload is not the start of the fragmented NAL unit payload, the S bit MUST be set to 0.

E: 1 bit

When set to 1, the E bit indicates the end of a fragmented NAL unit, i.e., the last byte of the payload is also the last byte of the fragmented NAL unit. When the FU payload is not the last fragment of a fragmented NAL unit, the E bit MUST be set to 0.

P: 1 bit

When set to 1, the P bit indicates the last FU of the last VCL NAL unit of a coded picture, i.e., the last byte of the FU payload is also the last byte of the last VCL NAL unit of the coded picture. When the FU payload is not the last fragment of the last VCL NAL unit of a coded picture, the P bit MUST be set to 0.

FuType: 5 bits

The field FuType MUST be equal to the field Type of the fragmented NAL unit.

The DONL field, when present, specifies the value of the 16 least significant bits of the decoding order number of the fragmented NAL unit.

If sprop-max-don-diff is greater than 0, and the S bit is equal to 1, the DONL field MUST be present in the FU, and the variable DON for the fragmented NAL unit is derived as equal to the value of the DONL field. Otherwise (sprop-max-don-diff is equal to 0, or the S bit is equal to 0), the DONL field MUST NOT be present in the FU.

A non-fragmented NAL unit MUST NOT be transmitted in one FU; i.e., the Start bit and End bit must not both be set to 1 in the same FU header.

The FU payload consists of fragments of the payload of the fragmented NAL unit so that if the FU payloads of consecutive FUs, starting with an FU with the S bit equal to 1 and ending with an FU with the E bit equal to 1, are sequentially concatenated, the payload of the fragmented NAL unit can be reconstructed. The NAL unit header of the fragmented NAL unit is not included as such in the FU payload, but rather the information of the NAL unit header of the fragmented NAL unit is conveyed in F, LayerId, and TID fields of the FU payload headers of the FUs and the FuType field of the FU header of the FUs. An FU payload MUST NOT be empty.

If an FU is lost, the receiver SHOULD discard all following fragmentation units in transmission order corresponding to the same fragmented NAL unit, unless the decoder in the receiver is known to be prepared to gracefully handle incomplete NAL units.

A receiver in an endpoint or in a MANE MAY aggregate the first $n-1$ fragments of a NAL unit to an (incomplete) NAL unit, even if fragment n of that NAL unit is not received. In this case, the `forbidden_zero_bit` of the NAL unit MUST be set to 1 to indicate a syntax violation.

4.4. Decoding Order Number

For each NAL unit, the variable `AbsDon` is derived, representing the decoding order number that is indicative of the NAL unit decoding order.

Let NAL unit n be the n -th NAL unit in transmission order within an RTP stream.

If `sprop-max-don-diff` is equal to 0, `AbsDon[n]`, the value of `AbsDon` for NAL unit n , is derived as equal to n .

Otherwise (`sprop-max-don-diff` is greater than 0), `AbsDon[n]` is derived as follows, where `DON[n]` is the value of the variable `DON` for NAL unit n :

- * If n is equal to 0 (i.e., NAL unit n is the very first NAL unit in transmission order), `AbsDon[0]` is set equal to `DON[0]`.

- * Otherwise (n is greater than 0), the following applies for derivation of `AbsDon[n]`:

- If `DON[n] == DON[n-1]`,
`AbsDon[n] = AbsDon[n-1]`

- If (`DON[n] > DON[n-1]` and `DON[n] - DON[n-1] < 32768`),
`AbsDon[n] = AbsDon[n-1] + DON[n] - DON[n-1]`

- If (`DON[n] < DON[n-1]` and `DON[n-1] - DON[n] >= 32768`),
`AbsDon[n] = AbsDon[n-1] + 65536 - DON[n-1] + DON[n]`

- If (`DON[n] > DON[n-1]` and `DON[n] - DON[n-1] >= 32768`),
`AbsDon[n] = AbsDon[n-1] - (DON[n-1] + 65536 - DON[n])`

- If (`DON[n] < DON[n-1]` and `DON[n-1] - DON[n] < 32768`),
`AbsDon[n] = AbsDon[n-1] - (DON[n-1] - DON[n])`

For any two NAL units m and n , the following applies:

- * $\text{AbsDon}[n]$ greater than $\text{AbsDon}[m]$ indicates that NAL unit n follows NAL unit m in NAL unit decoding order.
- * When $\text{AbsDon}[n]$ is equal to $\text{AbsDon}[m]$, the NAL unit decoding order of the two NAL units can be in either order.
- * $\text{AbsDon}[n]$ less than $\text{AbsDon}[m]$ indicates that NAL unit n precedes NAL unit m in decoding order.

Informative note: When two consecutive NAL units in the NAL unit decoding order have different values of AbsDon , the absolute difference between the two AbsDon values may be greater than or equal to 1.

Informative note: There are multiple reasons to allow for the absolute difference of the values of AbsDon for two consecutive NAL units in the NAL unit decoding order to be greater than one. An increment by one is not required, as at the time of associating values of AbsDon to NAL units, it may not be known whether all NAL units are to be delivered to the receiver. For example, a gateway might not forward VCL NAL units of higher sublayers or some SEI NAL units when there is congestion in the network. In another example, the first intra-coded picture of a pre-encoded clip is transmitted in advance to ensure that it is readily available in the receiver, and when transmitting the first intra-coded picture, the originator does not exactly know how many NAL units will be encoded before the first intra-coded picture of the pre-encoded clip follows in decoding order. Thus, the values of AbsDon for the NAL units of the first intra-coded picture of the pre-encoded clip have to be estimated when they are transmitted, and gaps in values of AbsDon may occur.

5. Packetization Rules

The following packetization rules apply:

- * If $\text{sprop-max-don-diff}$ is greater than 0, the transmission order of NAL units carried in the RTP stream MAY be different than the NAL unit decoding order. Otherwise ($\text{sprop-max-don-diff}$ is equal to 0), the transmission order of NAL units carried in the RTP stream MUST be the same as the NAL unit decoding order.
- * A NAL unit of a small size SHOULD be encapsulated in an aggregation packet together with one or more other NAL units in order to avoid the unnecessary packetization overhead for small

NAL units. For example, non-VCL NAL units such as access unit delimiters, parameter sets, or SEI NAL units are typically small and can often be aggregated with VCL NAL units without violating MTU size constraints.

- * Each non-VCL NAL unit SHOULD, when possible from an MTU size match viewpoint, be encapsulated in an aggregation packet together with its associated VCL NAL unit, as typically a non-VCL NAL unit would be meaningless without the associated VCL NAL unit being available.
- * For carrying exactly one NAL unit in an RTP packet, a single NAL unit packet MUST be used.

6. De-packetization Process

The general concept behind de-packetization is to get the NAL units out of the RTP packets in an RTP stream and pass them to the decoder in the NAL unit decoding order.

The de-packetization process is implementation dependent. Therefore, the following description should be seen as an example of a suitable implementation. Other schemes may be used as well, as long as the output for the same input is the same as the process described below. The output is the same when the set of output NAL units and their order are both identical. Optimizations relative to the described algorithms are possible.

All normal RTP mechanisms related to buffer management apply. In particular, duplicated or outdated RTP packets (as indicated by the RTP sequence number and the RTP timestamp) are removed. To determine the exact time for decoding, factors such as a possible intentional delay to allow for proper inter-stream synchronization MUST be factored in.

NAL units with NAL unit type values in the range of 0 to 27, inclusive, may be passed to the decoder. NAL-unit-like structures with NAL unit type values in the range of 28 to 31, inclusive, MUST NOT be passed to the decoder.

The receiver includes a receiver buffer, which is used to compensate for transmission delay jitter within individual RTP stream, and to reorder NAL units from transmission order to the NAL unit decoding order. In this section, the receiver operation is described under the assumption that there is no transmission delay jitter within an RTP stream. To make a difference from a practical receiver buffer that is also used for compensation of transmission delay jitter, the receiver buffer is hereafter called the de-packetization buffer in

this section. Receivers should also prepare for transmission delay jitter; that is, either reserve separate buffers for transmission delay jitter buffering and de-packetization buffering or use a receiver buffer for both transmission delay jitter and de-packetization. Moreover, receivers should take transmission delay jitter into account in the buffering operation, e.g., by additional initial buffering before starting of decoding and playback.

The de-packetization process extracts the NAL units from the RTP packets in an RTP stream as follows. When an RTP packet carries a single NAL unit packet, the payload of the RTP packet is extracted as a single NAL unit, excluding the DONL field, i.e., third and fourth bytes, when `sprop-max-don-diff` is greater than 0. When an RTP packet carries an Aggregation Packet, several NAL units are extracted from the payload of the RTP packet. In this case, each NAL unit corresponds to the part of the payload of each aggregation unit that follows the NALU size field as described in Section 4.3.2. When an RTP packet carries a Fragmentation Unit (FU), all RTP packets from the first FU (with the S field equal to 1) of the fragmented NAL unit up to the last FU (with the E field equal to 1) of the fragmented NAL unit are collected. The NAL unit is extracted from these RTP packets by concatenating all FU payloads in the same order as the corresponding RTP packets and appending the NAL unit header with the fields F, LayerId, and TID, set to equal to the values of the fields F, LayerId, and TID in the payload header of the FUs respectively, and with the NAL unit type set equal to the value of the field FuType in the FU header of the FUs, as described in Section 4.3.3.

When `sprop-max-don-diff` is equal to 0, the de-packetization buffer size is zero bytes, and the NAL units carried in the single RTP stream are directly passed to the decoder in their transmission order, which is identical to their decoding order.

When `sprop-max-don-diff` is greater than 0, the process described in the remainder of this section applies.

There are two buffering states in the receiver: initial buffering and buffering while playing. Initial buffering starts when the reception is initialized. After initial buffering, decoding and playback are started, and the buffering-while-playing mode is used.

Regardless of the buffering state, the receiver stores incoming NAL units in reception order into the de-packetization buffer. NAL units carried in RTP packets are stored in the de-packetization buffer individually, and the value of `AbsDon` is calculated and stored for each NAL unit.

Initial buffering lasts until the difference between the greatest and smallest AbsDon values of the NAL units in the de-packetization buffer is greater than or equal to the value of sprop-max-don-diff.

After initial buffering, whenever the difference between the greatest and smallest AbsDon values of the NAL units in the de-packetization buffer is greater than or equal to the value of sprop-max-don-diff, the following operation is repeatedly applied until this difference is smaller than sprop-max-don-diff:

- * The NAL unit in the de-packetization buffer with the smallest value of AbsDon is removed from the de-packetization buffer and passed to the decoder.

When no more NAL units are flowing into the de-packetization buffer, all NAL units remaining in the de-packetization buffer are removed from the buffer and passed to the decoder in the order of increasing AbsDon values.

7. Payload Format Parameters

This section specifies the optional parameters. A mapping of the parameters with Session Description Protocol (SDP) [RFC4556] is also provided for applications that use SDP.

7.1. Media Type Registration

The receiver MUST ignore any parameter unspecified in this memo.

Type name: video

Subtype name: H266

Required parameters: N/A

Optional parameters:

profile-id, tier-flag, sub-profile-id, interop-constraints, level-id, sprop-sublayer-id, sprop-ols-id, recv-sublayer-id, recv-ols-id, max-recv-level-id, sprop-dci, sprop-vps, sprop-sps, sprop-pps, sprop-sei, max-lsr, max-fps, sprop-max-don-diff, sprop-depack-buf-bytes, depack-buf-cap (Refer to Section 7.2 for definitions).

Encoding considerations:

This type is only defined for transfer via RTP (RFC 3550).

Security considerations:

See Section 9 of RFC XXXX.

Interoperability considerations: N/A

Published specification:

Please refer to RFC XXXX and its Section 13.

Applications that use this media type:

Any application that relies on VVC-based video services over RTP

Fragment identifier considerations: N/A

Additional information: N/A

Person & email address to contact for further information:

Stephan Wenger (stewe@stewe.org)

Intended usage: COMMON

Restrictions on usage: N/A

Author: See Authors' Addresses section of RFC XXXX.

Change controller:

IETF Audio/Video Transport Core Maintenance Working Group
delegated from the IESG.

7.2. Optional Parameters Definition

profile-id, tier-flag, sub-profile-id, interop-constraints, and
level-id:

These parameters indicate the profile, tier, default level, sub-profile, and some constraints of the bitstream carried by the RTP stream, or a specific set of the profile, tier, default level, sub-profile and some constraints the receiver supports.

The subset of coding tools that may have been used to generate the bitstream or that the receiver supports, as well as some additional constraints are indicated collectively by profile-id, sub-profile-id, and interop-constraints.

Informative note: There are 128 values of profile-id. The subset of coding tools identified by the profile-id can be further constrained with up to 255 instances of sub-profile-id. In addition, 68 bits included in interop-constraints, which can be extended up to 324 bits provide means to further restrict tools from existing profiles. To be able to support this fine-granular signaling of coding tool subsets with profile-id, sub-profile-id and interop-constraints, it would be safe to require symmetric use of these parameters in SDP offer/answer unless recv-ols-id is included in the SDP answer for choosing one of the layers offered.

The tier is indicated by tier-flag. The default level is indicated by level-id. The tier and the default level specify the limits on values of syntax elements or arithmetic combinations of values of syntax elements that are followed when generating the bitstream or that the receiver supports.

In SDP offer/answer, when the SDP answer does not include the recv-ols-id parameter that is less than the sprop-ols-id parameter in the SDP offer, the following applies:

- The tier-flag, profile-id, sub-profile-id, and interop-constraints parameters MUST be used symmetrically, i.e., the value of each of these parameters in the offer MUST be the same as that in the answer, either explicitly signaled or implicitly inferred.
- The level-id parameter is changeable as long as the highest level indicated by the answer is either equal to or lower than that in the offer. Note that a highest level higher than level-id in the offer for receiving can be included as max-recv-level-id.

In SDP offer/answer, when the SDP answer does include the recv-ols-id parameter that is less than the sprop-ols-id parameter in the SDP offer, the set of tier-flag, profile-id, sub-profile-id, interop-constraints, and level-id parameters included in the answer MUST be consistent with that for the chosen output layer set as indicated in the SDP offer, with the exception that the level-id parameter in the SDP answer is changeable as long as the highest level indicated by the answer is either lower than or equal to that in the offer.

More specifications of these parameters, including how they relate to syntax elements specified in [VVC] are provided below.

profile-id:

When profile-id is not present, a value of 1 (i.e., the Main 10 profile) MUST be inferred.

When used to indicate properties of a bitstream, profile-id is derived from the general_profile_idc syntax element that applies to the bitstream in an instance of the profile_tier_level() syntax structure.

VVC bitstreams transported over RTP using the technologies of this memo SHOULD contain only a single profile_tier_level() structure in the DCI, unless the sender can assure that a receiver can correctly decode the VVC bitstream regardless of which profile_tier_level() structure contained in the DCI was used for deriving profile-id and other parameters for the SDP O/A exchange.

As specified in [VVC], a profile_tier_level() syntax structure may be contained in an SPS NAL unit, and one or more profile_tier_level() syntax structures may be contained in a VPS NAL unit and in a DCI NAL unit. One of the following three cases applies to the container NAL unit of the profile_tier_level() syntax structure containing syntax elements used to derive the values of profile-id, tier-flag, level-id, sub-profile-id, or interop-constraints: 1) The container NAL unit is an SPS, the bitstream is a single-layer bitstream, and the profile_tier_level() syntax structures in all SPSS referenced by the CVSSs in the bitstream has the same values respectively for those profile_tier_level() syntax elements; 2) The container NAL unit is a VPS, the profile_tier_level() syntax structure is the one in the VPS that applies to the OLS corresponding to the bitstream, and the profile_tier_level() syntax structures applicable to the OLS corresponding to the bitstream in all VPSSs referenced by the CVSSs in the bitstream have the same values respectively for those profile_tier_level() syntax elements; 3) The container NAL unit is a DCI NAL unit and the profile_tier_level() syntax structures in all DCI NAL units in the bitstream has the same values respectively for those profile_tier_level() syntax elements.

[VVC] allows for multiple profile_tier_level() structures in a DCI NAL unit, which may contain different values for the syntax elements used to derive the values of profile-id, tier-flag, level-id, sub-profile-id, or interop-constraints in the different entries. However, herein defined is only a single profile-id, tier-flag, level-id, sub-profile-id, or interop-constraints. When signaling these parameters and a DCI NAL unit is present with multiple profile_tier_level() structures, these values SHOULD be the same as the first profile_tier_level structure in the DCI, unless the sender has ensured that the receiver can decode the bitstream when a different value is chosen.

tier-flag, level-id:

The value of tier-flag MUST be in the range of 0 to 1, inclusive.
The value of level-id MUST be in the range of 0 to 255, inclusive.

If the tier-flag and level-id parameters are used to indicate properties of a bitstream, they indicate the tier and the highest level the bitstream complies with.

If the tier-flag and level-id parameters are used for capability exchange, the following applies. If max-recv-level-id is not present, the default level defined by level-id indicates the highest level the codec wishes to support. Otherwise, max-recv-level-id indicates the highest level the codec supports for receiving. For either receiving or sending, all levels that are lower than the highest level supported MUST also be supported.

If no tier-flag is present, a value of 0 MUST be inferred; if no level-id is present, a value of 51 (i.e., level 3.1) MUST be inferred.

Informative note: The level values currently defined in the VVC specification are in the form of "majorNum.minorNum", and the value of the level-id for each of the levels is equal to $\text{majorNum} * 16 + \text{minorNum} * 3$. It is expected that if any levels are defined in the future, the same convention will be used, but this cannot be guaranteed.

When used to indicate properties of a bitstream, the tier-flag and level-id parameters are derived respectively from the syntax element `general_tier_flag`, and the syntax element `general_level_idc` or `sub_layer_level_idc[j]`, that apply to the bitstream, in an instance of the `profile_tier_level()` syntax structure.

If the tier-flag and level-id are derived from the `profile_tier_level()` syntax structure in a DCI NAL unit, the following applies:

- tier-flag = `general_tier_flag`
- level-id = `general_level_idc`

Otherwise, if the tier-flag and level-id are derived from the `profile_tier_level()` syntax structure in an SPS or VPS NAL unit, and the bitstream contains the highest sublayer representation in the OLS corresponding to the bitstream, the following applies:

- tier-flag = general_tier_flag
- level-id = general_level_idc

Otherwise, if the tier-flag and level-id are derived from the profile_tier_level() syntax structure in an SPS or VPS NAL unit, and the bitstream does not contain the highest sublayer representation in the OLS corresponding to the bitstream, the following applies, with j being the value of the sprop-sublayer-id parameter:

- tier-flag = general_tier_flag
- level-id = sub_layer_level_idc[j]

sub-profile-id:

The value of the parameter is a comma-separated (',') list of data using base64 [RFC4648] representation.

When used to indicate properties of a bitstream, sub-profile-id is derived from each of the ptl_num_sub_profiles general_sub_profile_idc[i] syntax elements that apply to the bitstream in a profile_tier_level() syntax structure.

interop-constraints:

A base64 [RFC4648] representation of the data that includes the syntax elements ptl_frame_only_constraint_flag and ptl_multilayer_enabled_flag and the general_constraints_info() syntax structure that apply to the bitstream in an instance of the profile_tier_level() syntax structure.

If the interop-constraints parameter is not present, the following MUST be inferred:

- ptl_frame_only_constraint_flag = 1
- ptl_multilayer_enabled_flag = 0
- gci_present_flag in the general_constraints_info() syntax structure = 0

Using interop-constraints for capability exchange results in a requirement on any bitstream to be compliant with the interop-constraints.

sprop-sublayer-id:

This parameter MAY be used to indicate the highest allowed value of TID in the bitstream. When not present, the value of sprop-sublayer-id is inferred to be equal to 6.

The value of sprop-sublayer-id MUST be in the range of 0 to 6, inclusive.

sprop-ols-id:

This parameter MAY be used to indicate the OLS that the bitstream applies to. When not present, the value of sprop-ols-id is inferred to be equal to TargetOlsIdx as specified in 8.1.1 in [VVC]. If this optional parameter is present, sprop-vps MUST also be present or its content MUST be known a priori at the receiver.

The value of sprop-ols-id MUST be in the range of 0 to 256, inclusive.

Informative note: VVC allows having up to 257 output layer sets indicated in the VPS as the number of output layer sets minus 2 is indicated with a field of 8 bits.

recv-sublayer-id:

This parameter MAY be used to signal a receiver's choice of the offered or declared sublayer representations in the sprop-vps and sprop-sps. The value of recv-sublayer-id indicates the TID of the highest sublayer that a receiver supports. When not present, the value of recv-sublayer-id is inferred to be equal to the value of the sprop-sublayer-id parameter in the SDP offer.

The value of recv-sublayer-id MUST be in the range of 0 to 6, inclusive.

recv-ols-id:

This parameter MAY be used to signal a receiver's choice of the offered or declared output layer sets in the sprop-vps. The value of recv-ols-id indicates the OLS index of the bitstream that a receiver supports. When not present, the value of recv-ols-id is inferred to be equal to value of the sprop-ols-id parameter inferred from or indicated in the SDP offer. When present, the value of recv-ols-id must be included only when sprop-ols-id was received and must refer to an output layer set in the VPS that includes no layers other than all or a subset of the layers of the OLS referred to by sprop-ols-id. If this optional parameter is present, sprop-vps must have been received or its content must be known a priori at the receiver.

The value of `recv-ols-id` MUST be in the range of 0 to 256, inclusive.

`max-recv-level-id`:

This parameter MAY be used to indicate the highest level a receiver supports.

The value of `max-recv-level-id` MUST be in the range of 0 to 255, inclusive.

When `max-recv-level-id` is not present, the value is inferred to be equal to `level-id`.

`max-recv-level-id` MUST NOT be present when the highest level the receiver supports is not higher than the default level.

`sprop-dci`:

This parameter MAY be used to convey a decoding capability information NAL unit of the bitstream for out-of-band transmission. The parameter MAY also be used for capability exchange. The value of the parameter is a base64 [RFC4648] representations of the decoding capability information NAL unit as specified in Section 7.3.2.1 of [VVC].

`sprop-vps`:

This parameter MAY be used to convey any video parameter set NAL unit of the bitstream for out-of-band transmission of video parameter sets. The parameter MAY also be used for capability exchange and to indicate sub-stream characteristics (i.e., properties of output layer sets and sublayer representations as defined in [VVC]). The value of the parameter is a comma-separated ('(',')') list of base64 [RFC4648] representations of the video parameter set NAL units as specified in Section 7.3.2.3 of [VVC].

The `sprop-vps` parameter MAY contain one or more than one video parameter set NAL units. However, all other video parameter sets contained in the `sprop-vps` parameter MUST be consistent with the first video parameter set in the `sprop-vps` parameter. A video parameter set `vpsB` is said to be consistent with another video parameter set `vpsA` if the number of OLSs in `vpsA` and `vpsB` is the same and any decoder that conforms to the profile, tier, level, and constraints indicated by the data starting from the syntax element `general_profile_idc` to the syntax structure `general_constraints_info()`, inclusive, in the `profile_tier_level()`

) syntax structure corresponding to any OLS with index `olsIdx` in `vpsA` can decode any CVS(s) referencing `vpsB` when `TargetOlsIdx` is equal to `olsIdx` that conforms to the profile, tier, level, and constraints indicated by the data starting from the syntax element `general_profile_idc` to the syntax structure `general_constraints_info()`, inclusive, in the `profile_tier_level()` syntax structure corresponding to the OLS with index `TargetOlsIdx` in `vpsB`.

`sprop-sps:`

This parameter MAY be used to convey sequence parameter set NAL units of the bitstream for out-of-band transmission of sequence parameter sets. The value of the parameter is a comma-separated ('(',')') list of base64 [RFC4648] representations of the sequence parameter set NAL units as specified in Section 7.3.2.4 of [VVC].

A sequence parameter set `spsB` is said to be consistent with another sequence parameter set `spsA` if any decoder that conforms to the profile, tier, level, and constraints indicated by the data starting from the syntax element `general_profile_idc` to the syntax structure `general_constraints_info()`, inclusive, in the `profile_tier_level()` syntax structure in `spsA` can decode any CLVS(s) referencing `spsB` that conforms to the profile, tier, level, and constraints indicated by the data starting from the syntax element `general_profile_idc` to the syntax structure `general_constraints_info()`, inclusive, in the `profile_tier_level()` syntax structure in `spsB`.

`sprop-pps:`

This parameter MAY be used to convey picture parameter set NAL units of the bitstream for out-of-band transmission of picture parameter sets. The value of the parameter is a comma-separated ('(',')') list of base64 [RFC4648] representations of the picture parameter set NAL units as specified in Section 7.3.2.5 of [VVC].

`sprop-sei:`

This parameter MAY be used to convey one or more SEI messages that describe bitstream characteristics. When present, a decoder can rely on the bitstream characteristics that are described in the SEI messages for the entire duration of the session, independently from the persistence scopes of the SEI messages as specified in [VSEI].

The value of the parameter is a comma-separated (',') list of base64 [RFC4648] representations of SEI NAL units as specified in [VSEI].

Informative note: Intentionally, no list of applicable or inapplicable SEI messages is specified here. Conveying certain SEI messages in sprop-sei may be sensible in some application scenarios and meaningless in others. However, a few examples are described below:

- 1) In an environment where the bitstream was created from film-based source material, and no splicing is going to occur during the lifetime of the session, the film grain characteristics SEI message is likely meaningful, and sending it in sprop-sei rather than in the bitstream at each entry point may help with saving bits and allows one to configure the renderer only once, avoiding unwanted artifacts.
- 2) Examples for SEI messages that would be meaningless to be conveyed in sprop-sei include the decoded picture hash SEI message (it is close to impossible that all decoded pictures have the same hashtag) or the filler payload SEI message (as there is no point in just having more bits in SDP).

max-lsr:

The max-lsr MAY be used to signal the capabilities of a receiver implementation and MUST NOT be used for any other purpose. The value of max-lsr is an integer indicating the maximum processing rate in units of luma samples per second. The max-lsr parameter signals that the receiver is capable of decoding video at a higher rate than is required by the highest level.

Informative note: When the OPTIONAL media type parameters are used to signal the properties of a bitstream, and max-lsr is not present, the values of tier-flag, profile-id, sub-profile-id interop-constraints, and level-id must always be such that the bitstream complies fully with the specified profile, tier, and level.

When max-lsr is signaled, the receiver MUST be able to decode bitstreams that conform to the highest level, with the exception that the MaxLumaSr value in Table 136 of [VVC] for the highest level is replaced with the value of max-lsr. Senders MAY use this knowledge to send pictures of a given size at a higher picture rate than is indicated in the highest level.

When not present, the value of max-lsr is inferred to be equal to the value of MaxLumaSr given in Table 136 of [VVC] for the highest level.

The value of max-lsr MUST be in the range of MaxLumaSr to $16 * \text{MaxLumaSr}$, inclusive, where MaxLumaSr is given in Table 136 of [VVC] for the highest level.

max-fps:

The value of max-fps is an integer indicating the maximum picture rate in units of pictures per 100 seconds that can be effectively processed by the receiver. The max-fps parameter MAY be used to signal that the receiver has a constraint in that it is not capable of processing video effectively at the full picture rate that is implied by the highest level and, when present, max-lsr.

The value of max-fps is not necessarily the picture rate at which the maximum picture size can be sent, it constitutes a constraint on maximum picture rate for all resolutions.

Informative note: The max-fps parameter is semantically different from max-lsr in that max-fps is used to signal a constraint, lowering the maximum picture rate from what is implied by other parameters.

The encoder MUST use a picture rate equal to or less than this value. In cases where the max-fps parameter is absent, the encoder is free to choose any picture rate according to the highest level and any signaled optional parameters.

The value of max-fps MUST be smaller than or equal to the full picture rate that is implied by the highest level and, when present, max-lsr.

sprop-max-don-diff:

If there is no NAL unit naluA that is followed in transmission order by any NAL unit preceding naluA in decoding order (i.e., the transmission order of the NAL units is the same as the decoding order), the value of this parameter MUST be equal to 0.

Otherwise, this parameter specifies the maximum absolute difference between the decoding order number (i.e., AbsDon) values of any two NAL units naluA and naluB, where naluA follows naluB in decoding order and precedes naluB in transmission order.

The value of `sprop-max-don-diff` MUST be an integer in the range of 0 to 32767, inclusive.

When not present, the value of `sprop-max-don-diff` is inferred to be equal to 0.

`sprop-depack-buf-bytes`:

This parameter signals the required size of the de-packetization buffer in units of bytes. The value of the parameter MUST be greater than or equal to the maximum buffer occupancy (in units of bytes) of the de-packetization buffer as specified in Section 6.

The value of `sprop-depack-buf-bytes` MUST be an integer in the range of 0 to 4294967295, inclusive.

When `sprop-max-don-diff` is present and greater than 0, this parameter MUST be present and the value MUST be greater than 0. When not present, the value of `sprop-depack-buf-bytes` is inferred to be equal to 0.

Informative note: The value of `sprop-depack-buf-bytes` indicates the required size of the de-packetization buffer only. When network jitter can occur, an appropriately sized jitter buffer has to be available as well.

`depack-buf-cap`:

This parameter signals the capabilities of a receiver implementation and indicates the amount of de-packetization buffer space in units of bytes that the receiver has available for reconstructing the NAL unit decoding order from NAL units carried in the RTP stream. A receiver is able to handle any RTP stream for which the value of the `sprop-depack-buf-bytes` parameter is smaller than or equal to this parameter.

When not present, the value of `depack-buf-cap` is inferred to be equal to 4294967295. The value of `depack-buf-cap` MUST be an integer in the range of 1 to 4294967295, inclusive.

Informative note: `depack-buf-cap` indicates the maximum possible size of the de-packetization buffer of the receiver only, without allowing for network jitter.

7.3. SDP Parameters

The receiver MUST ignore any parameter unspecified in this memo.

7.3.1. Mapping of Payload Type Parameters to SDP

The media type video/H266 string is mapped to fields in the Session Description Protocol (SDP) [RFC8866] as follows:

- * The media name in the "m=" line of SDP MUST be video.
- * The encoding name in the "a=rtpmap" line of SDP MUST be H266 (the media subtype).
- * The clock rate in the "a=rtpmap" line MUST be 90000.
- * The OPTIONAL parameters profile-id, tier-flag, sub-profile-id, interop-constraints, level-id, sprop-sublayer-id, sprop-ols-id, recv-sublayer-id, recv-ols-id, max-recv-level-id, max-lsr, max-fps, sprop-max-don-diff, sprop-depack-buf-bytes and depack-buf-cap, when present, MUST be included in the "a=fmtp" line of SDP. The fmtp line is expressed as a media type string, in the form of a semicolon-separated list of parameter=value pairs.
- * The OPTIONAL parameter sprop-vps, sprop-sps, sprop-pps, sprop-sei, and sprop-dci, when present, MUST be included in the "a=fmtp" line of SDP or conveyed using the "fmtp" source attribute as specified in Section 6.3 of [RFC5576]. For a particular media format (i.e., RTP payload type), sprop-vps, sprop-sps, sprop-pps, sprop-sei, or sprop-dci MUST NOT be both included in the "a=fmtp" line of SDP and conveyed using the "fmtp" source attribute. When included in the "a=fmtp" line of SDP, those parameters are expressed as a media type string, in the form of a semicolon-separated list of parameter=value pairs. When conveyed in the "a=fmtp" line of SDP for a particular payload type, the parameters sprop-vps, sprop-sps, sprop-pps, sprop-sei, and sprop-dci MUST be applied to each SSRC with the payload type. When conveyed using the "fmtp" source attribute, these parameters are only associated with the given source and payload type as parts of the "fmtp" source attribute.

Informative note: Conveyance of sprop-vps, sprop-sps, and sprop-pps using the "fmtp" source attribute allows for out-of-band transport of parameter sets in topologies like Topo-Video-switch-MCU as specified in [RFC7667]

An general usage of media representation in SDP is as follows:

```
m=video 49170 RTP/AVP 98
a=rtpmap:98 H266/90000
a=fmtp:98 profile-id=1;
    sprop-vps=<video parameter sets data>;
    sprop-sps=<sequence parameter set data>;
    sprop-pps=<picture parameter set data>;
```

A SIP Offer/Answer exchange wherein both parties are expected to both send and receive could look like the following. Only the media codec-specific parts of the SDP are shown. Some lines are wrapped due to text constraints.

Offerer->Answerer:

```
m=video 49170 RTP/AVP 98
a=rtpmap:98 H266/90000
a=fmtp:98 profile-id=1; level_id=83;
```

The above represents an offer for symmetric video communication using [VVC] and it's payload specification, at the main profile and level 5.1 (and, as the levels are downgradable, all lower levels. Informally speaking, this offer tells the receiver of the offer that the sender is willing to receive up to 4Kp60 resolution at the maximum bitrates specified in [VVC]. At the same time, if this offer were accepted "as is", the offer can expect that the answerer would be able to receive and properly decode H.266 media up to and including level 5.1.

Answerer->Offerer:

```
m=video 49170 RTP/AVP 98
a=rtpmap:98 H266/90000
a=fmtp:98 profile-id=1; level_id=67
```

With this answer to the offer above, the system receiving the offer advises the offerer that it is incapable of handling H.266 at level 5.1 but is capable of decoding 1080p60. As H.266 video codecs must support decoding at all levels below the maximum level they implement, the resulting user experience would likely be that both systems send video at 1080p60. However, nothing prevents an encoder from further downgrading its sending to, for example 720p30 if it were short of cycles, bandwidth, or for other reasons.

7.3.2. Usage with SDP Offer/Answer Model

This section describes the negotiation of unicast messages using the offer-answer model as described in [RFC3264] and its updates. The section is split into subsections, covering a) media format configurations not involving non-temporal scalability; b) scalable media format configurations; c) the description of the use of those parameters not involving the media configuration itself but rather the parameters of the payload format design; and d) multicast.

7.3.2.1. Non-scalable media format configuration

A non-scalable VVC media configuration is such a configuration where no non-temporal scalability mechanisms are allowed. In [VVC] version 1, that implies that `general_profile_idc` indicates one of the following profiles: Main10, Main10 Still Picture, Main 10 4:4:4, Main10 4:4:4 Still Picture, with `general_profile_idc` values of 1, 65, 33, and 97, respectively. Note that non-scalable media configurations includes temporal scalability, inline with VVC's design philosophy and profile structure.

The following limitations and rules pertaining to the media configuration apply:

- * The parameters identifying a media format configuration for VVC are `profile-id`, `tier-flag`, `sub-profile-id`, `level-id`, and `interop-constraints`. These media configuration parameters, except `level-id`, MUST be used symmetrically.

The answerer MUST structure its answer in according to one of the following three options:

1) maintain all configuration parameters with the values remaining the same as in the offer for the media format (payload type), with the exception that the value of `level-id` is changeable as long as the highest level indicated by the answer is not higher than that indicated by the offer;

2) include in the answer the `recv-sublayer-id` parameter, with a value less than the `sprop-sublayer-id` parameter in the offer, for the media format (payload type), and maintain all configuration parameters with the values remaining the same as in the offer for the media format (payload type), with the exception that the value of `level-id` is changeable as long as the highest level indicated by the answer is not higher than the level indicated by the `sprop-sps` or `sprop-vps` in offer for the chosen sublayer representation; or

3) remove the media format (payload type) completely (when one or more of the parameter values are not supported).

Informative note: The above requirement for symmetric use does not apply for level-id, and does not apply for the other bitstream or RTP stream properties and capability parameters as described in Section 7.3.2.3 below.

- * To simplify handling and matching of these configurations, the same RTP payload type number used in the offer SHOULD also be used in the answer, as specified in [RFC3264].
- * The same RTP payload type number used in the offer for the media subtype H266 MUST be used in the answer when the answer includes recv-sublayer-id. When the answer does not include recv-sublayer-id, the answer MUST NOT contain a payload type number used in the offer for the media subtype H266 unless the configuration is exactly the same as in the offer or the configuration in the answer only differs from that in the offer with a different value of level-id. The answer MAY contain the recv-sublayer-id parameter if an VVC bitstream contains multiple operation points (using temporal scalability and sublayers) and sprop-sps or sprop-vps is included in the offer where information of sublayers are present in the first sequence parameter set or video parameter set contained in sprop-sps or sprop-vps respectively. If the sprop-sps or sprop-vps is provided in an offer, an answerer MAY select a particular operation point indicated in the first sequence parameter set or video parameter set contained in sprop-sps or sprop-vps respectively. When the answer includes a recv-sublayer-id that is less than a sprop-sublayer-id in the offer, the following applies:
 - 1) When sprop-sps parameter is present, all sequence parameter sets contained in the sprop-sps parameter in the SDP answer and all sequence parameter sets sent in-band for either the offerer-to-answerer direction or the answerer-to-offerer direction MUST be consistent with the first sequence parameter set in the sprop-sps parameter of the offer (see the semantics of sprop-sps in Section 7.1 of this document on one sequence parameter set being consistent with another sequence parameter set).

2) When sprop-vps parameter is present, all video parameter sets contained in the sprop-vps parameter in the SDP answer and all video parameter sets sent in-band for either the offerer-to-answerer direction or the answerer-to-offerer direction MUST be consistent with the first video parameter set in the sprop-vps parameter of the offer (see the semantics of sprop-vps in Section 7.1 of this document on one video parameter set being consistent with another video parameter set).

3) The bitstream sent in either direction MUST conform to the profile, tier, level, and constraints of the chosen sublayer representation as indicated by the profile_tier_level() syntax structure in the first sequence parameter set in the sprop-sps parameter or by the first profile_tier_level() syntax structure in the first video parameter set in the sprop-vps parameter of the offer.

Informative note: When an offerer receives an answer that does not include recv-sublayer-id, it has to compare payload types not declared in the offer based on the media type (i.e., video/H266) and the above media configuration parameters with any payload types it has already declared. This will enable it to determine whether the configuration in question is new or if it is equivalent to configuration already offered, since a different payload type number may be used in the answer. The ability to perform operation point selection enables a receiver to utilize the temporal scalable nature of an VVC bitstream.

7.3.2.2. Scalable media format configuration

A scalable VVC media configuration is such a configuration where non-temporal scalability mechanisms are allowed. In [VVC] version 1, that implies that general_profile_idc indicates one of the following profiles: Multilayer Main 10, and Multilayer Main 10 4:4:4, with general_profile_idc values of 17 and 49, respectively.

The following limitations and rules pertaining to the media configuration apply. They are listed in an order that would be logical for an implementation to follow:

- * The parameters identifying a media format configuration for scalable VVC are profile-id, tier-flag, sub-profile-id, level-id, interop-constraints, and sprop-vps. These media configuration parameters, except level-id, MUST be used symmetrically, except as noted below.

- * The answerer MAY include a level-id that MUST be lower than or equal to the level-id indicated in the offer (either expressed by level-id in the offer, or implied by the default level as specific in Section 7.1).
- * When sprop-ols-id is present in an offer, sprop-vps MUST also be present in the same offer and including at least one valid VPS, so to allow the answerer to meaningfully interpret sprop-ols-id and select recv-ols-id (see below).
- * The answerer MUST NOT include recv-ols-id unless the offer includes sprop-ols-id. When present, recv-ols-id MUST indicate a supported output layer set in the VPS that includes no layers other than all or a subset of the layers of the OLS referred to by sprop-ols-id. If unable, the answerer MUST remove the media format.

Informative note: if an offerer wants to offer more than one output layer set, it can do so by offering multiple VVC media with different payload types.

- * The offerer MAY include sprop-sublayer-id which indicates the highest allowed value of TID in the bitstream. The answerer MAY include recv-sublayer-id which can be used to reduce the number of sublayers from the value of sprop-sublayer-id.
- * When the answerer includes recv-ols-id and configuration parameters profile-id, tier-flag, sub-profile-id, level-id, and interop-constraints, it MUST use the configuration parameter values as signaled in the sprop-vps for the operating point with the largest number of sublayers for the chosen output layer set, with the exception that the value of level-id is changeable as long as the highest level indicated by the answer is not higher than the level indicated by the sprop-vps in offer for the operating point with the largest number of sublayers for the chosen output layer set.

7.3.2.3. Payload format configuration

The following limitations and rules pertain to the configuration of the payload format buffer management mostly and apply to both scalable and non-scalable VVC.

- * The parameters sprop-max-don-diff, and sprop-depack-buf-bytes describe the properties of an RTP stream that the offerer or the answerer is sending for the media format configuration. This differs from the normal usage of the offer/answer parameters: normally such parameters declare the properties of the bitstream

or RTP stream that the offerer or the answerer is able to receive. When dealing with VVC, the offerer assumes that the answerer will be able to receive media encoded using the configuration being offered.

Informative note: The above parameters apply for any RTP stream, when present, sent by a declaring entity with the same configuration. In other words, the applicability of the above parameters to RTP streams depends on the source endpoint. Rather than being bound to the payload type, the values may have to be applied to another payload type when being sent, as they apply for the configuration.

- * The capability parameter max-lsr MAY be used to declare further capabilities of the offerer or answerer for receiving. It MUST NOT be present when the direction attribute is sendonly.
- * The capability parameter max-fps MAY be used to declare lower capabilities of the offerer or answerer for receiving. It MUST NOT be present when the direction attribute is sendonly.
- * When an offerer offers an interleaved stream, indicated by the presence of sprop-max-don-diff with a value larger than zero, the offerer MUST include the size of the de-packetization buffer sprop-depack-buf-bytes.
- * To enable the offerer and answerer to inform each other about their capabilities for de-packetization buffering in receiving RTP streams, both parties are RECOMMENDED to include depack-buf-cap.
- * The sprop-dci, sprop-vps, sprop-sps, or sprop-pps, when present (included in the "a=fmtp" line of SDP or conveyed using the "fmtp" source attribute as specified in Section 6.3 of [RFC5576]), are used for out-of-band transport of the parameter sets (DCI, VPS, SPS, or PPS, respectively).
- * The answerer MAY use either out-of-band or in-band transport of parameter sets for the bitstream it is sending, regardless of whether out-of-band parameter sets transport has been used in the offerer-to-answerer direction. Parameter sets included in an answer are independent of those parameter sets included in the offer, as they are used for decoding two different bitstreams, one from the answerer to the offerer and the other in the opposite direction. In case some RTP packets are sent before the SDP offer/answer settles down, in-band parameter sets MUST be used for those RTP stream parts sent before the SDP offer/answer.

- * The following rules apply to transport of parameter set in the offerer-to-answerer direction.
 - An offer MAY include sprop-dci, sprop-vps, sprop-sps, and/or sprop-pps. If none of these parameters is present in the offer, then only in-band transport of parameter sets is used.
 - If the level to use in the offerer-to-answerer direction is equal to the default level in the offer, the answerer MUST be prepared to use the parameter sets included in sprop-vps, sprop-sps, and sprop-pps (either included in the "a=fmtp" line of SDP or conveyed using the "fmtp" source attribute) for decoding the incoming bitstream, e.g., by passing these parameter set NAL units to the video decoder before passing any NAL units carried in the RTP streams. Otherwise, the answerer MUST ignore sprop-vps, sprop-sps, and sprop-pps (either included in the "a=fmtp" line of SDP or conveyed using the "fmtp" source attribute) and the offerer MUST transmit parameter sets in-band.
- * The following rules apply to transport of parameter set in the answerer-to-offerer direction.
 - An answer MAY include sprop-dci, sprop-vps, sprop-sps, and/or sprop-pps. If none of these parameters is present in the answer, then only in-band transport of parameter sets is used.
 - The offerer MUST be prepared to use the parameter sets included in sprop-vps, sprop-sps, and sprop-pps (either included in the "a=fmtp" line of SDP or conveyed using the "fmtp" source attribute) for decoding the incoming bitstream, e.g., by passing these parameter set NAL units to the video decoder before passing any NAL units carried in the RTP streams.
- * When sprop-dci, sprop-vps, sprop-sps, and/or sprop-pps are conveyed using the "fmtp" source attribute as specified in Section 6.3 of [RFC5576], the receiver of the parameters MUST store the parameter sets included in sprop-dci, sprop-vps, sprop-sps, and/or sprop-pps and associate them with the source given as part of the "fmtp" source attribute. Parameter sets associated with one source (given as part of the "fmtp" source attribute) MUST only be used to decode NAL units conveyed in RTP packets from the same source (given as part of the "fmtp" source attribute). When this mechanism is in use, SSRC collision detection and resolution MUST be performed as specified in [RFC5576].

Table 1 lists the interpretation of all the parameters that MAY be used for the various combinations of offer, answer, and direction attributes. Note that the two columns wherein the `recv-ols-id` parameter is used only apply to answers, whereas the other columns apply to both offers and answers.

	sendonly --+				
	answer: recvonly, recv-ols-id --+				
	recvonly w/o recv-ols-id --+				
	answer: sendrecv, recv-ols-id --+				
	sendrecv w/o recv-ols-id --+				
profile-id	C	D	C	D	P
tier-flag	C	D	C	D	P
level-id	D	D	D	D	P
sub-profile-id	C	D	C	D	P
interop-constraints	C	D	C	D	P
max-recv-level-id	R	R	R	R	-
sprop-max-don-diff	P	P	-	-	P
sprop-depack-buf-bytes	P	P	-	-	P
depack-buf-cap	R	R	R	R	-
max-lsr	R	R	R	R	-
max-fps	R	R	R	R	-
sprop-dci	P	P	-	-	P
sprop-sei	P	P	-	-	P
sprop-vps	P	P	-	-	P
sprop-sps	P	P	-	-	P
sprop-pps	P	P	-	-	P
sprop-sublayer-id	P	P	-	-	P
recv-sublayer-id	O	O	O	O	-
sprop-ols-id	P	P	-	-	P
recv-ols-id	X	O	X	O	-

Table 1. Interpretation of parameters for various combinations of offers, answers, direction attributes, with and without recv-ols-id. Columns that do not indicate offer or answer apply to both.

Legend:

- C: configuration for sending and receiving bitstreams
- D: changeable configuration, same as C except possible to answer with a different but consistent value (see the semantics of the six parameters related to profile, tier, and level on these parameters being consistent)
- P: properties of the bitstream to be sent
- R: receiver capabilities
- O: operation point selection
- X: MUST NOT be present
- : not usable, when present MUST be ignored

Parameters used for declaring receiver capabilities are, in general, downgradable; i.e., they express the upper limit for a sender's possible behavior. Thus, a sender MAY select to set its encoder using only lower/lesser or equal values of these parameters.

When the answer does not include a `recv-ols-id` that is less than the `sprop-ols-id` in the offer, parameters declaring a configuration point are not changeable, with the exception of the `level-id` parameter for unicast usage, and these parameters express values a receiver expects to be used and MUST be used verbatim in the answer as in the offer.

When a sender's capabilities are declared with the configuration parameters, these parameters express a configuration that is acceptable for the sender to receive bitstreams. In order to achieve high interoperability levels, it is often advisable to offer multiple alternative configurations. It is impossible to offer multiple configurations in a single payload type. Thus, when multiple configuration offers are made, each offer requires its own RTP payload type associated with the offer. However, it is possible to offer multiple operation points using one configuration in a single payload type by including `sprop-vps` in the offer and `recv-ols-id` in the answer.

An implementation SHOULD be able to understand all media type parameters (including all optional media type parameters), even if it doesn't support the functionality related to the parameter. This, in conjunction with proper application logic in the implementation allows the implementation, after having received an offer, to create an answer by potentially downgrading one or more of the optional parameters to the point where the implementation can cope, leading to higher chances of interoperability beyond the most basic interop points (for which, as described above, no optional parameters are necessary).

Informative note: in implementations of previous H.26x payload formats it was occasionally observed that implementations were incapable of parsing most (or all) of the optional parameters. As a result, the offer-answer exchange resulted in a baseline performance (using the default values for the optional parameters) with the resulting suboptimal user experience. However, there are valid reasons to forego the implementation complexity of implementing the parsing of some or all of the optional parameters, for example, when there is pre-determined knowledge, not negotiated by an SDP-based offer/answer process, of the capabilities of the involved systems (walled gardens, baseline requirements defined in application standards higher up in the stack, and similar).

An answerer MAY extend the offer with additional media format configurations. However, to enable their usage, in most cases a second offer is required from the offerer to provide the bitstream property parameters that the media sender will use. This also has the effect that the offerer has to be able to receive this media format configuration, not only to send it.

7.3.2.4. Multicast

For bitstreams being delivered over multicast, the following rules apply:

- * The media format configuration is identified by profile-id, tier-flag, sub-profile-id, level-id, and interop-constraints. These media format configuration parameters, including level-id, MUST be used symmetrically; that is, the answerer MUST either maintain all configuration parameters or remove the media format (payload type) completely. Note that this implies that the level-id for offer/answer in multicast is not changeable.
- * To simplify the handling and matching of these configurations, the same RTP payload type number used in the offer SHOULD also be used in the answer, as specified in [RFC3264]. An answer MUST NOT contain a payload type number used in the offer unless the configuration is the same as in the offer.
- * Parameter sets received MUST be associated with the originating source and MUST only be used in decoding the incoming bitstream from the same source.
- * The rules for other parameters are the same as above for unicast as long as the three above rules are obeyed.

7.3.3. Usage in Declarative Session Descriptions

When VVC over RTP is offered with SDP in a declarative style, as in Real Time Streaming Protocol (RTSP) [RFC7826] or Session Announcement Protocol (SAP) [RFC2974], the following considerations are necessary.

- * All parameters capable of indicating both bitstream properties and receiver capabilities are used to indicate only bitstream properties. For example, in this case, the parameter profile-id, tier-id, level-id declares the values used by the bitstream, not the capabilities for receiving bitstreams. As a result, the following interpretation of the parameters MUST be used:
 - Declaring actual configuration or bitstream properties:

- o profile-id
- o tier-flag
- o level-id
- o interop-constraints
- o sub-profile-id
- o sprop-dci
- o sprop-vps
- o sprop-sps
- o sprop-pps
- o sprop-max-don-diff
- o sprop-depack-buf-bytes
- o sprop-sublayer-id
- o sprop-ols-id
- o sprop-sei
- Not usable (when present, they MUST be ignored):
 - o max-lsr
 - o max-fps
 - o max-recv-level-id
 - o depack-buf-cap
 - o recv-sublayer-id
 - o recv-ols-id
- A receiver of the SDP is required to support all parameters and values of the parameters provided; otherwise, the receiver MUST reject (RTSP) or not participate in (SAP) the session. It falls on the creator of the session to use values that are expected to be supported by the receiving application.

7.3.4. Considerations for Parameter Sets

When out-of-band transport of parameter sets is used, parameter sets MAY still be additionally transported in-band unless explicitly disallowed by an application, and some of these additional parameter sets may update some of the out-of-band transported parameter sets. Update of a parameter set refers to the sending of a parameter set of the same type using the same parameter set ID but with different values for at least one other parameter of the parameter set.

8. Use with Feedback Messages

The following subsections define the use of the Picture Loss Indication (PLI) and Full Intra Request (FIR) feedback messages with [VVC]. The PLI is defined in [RFC4585], and the FIR message is defined in [RFC5104]. In accordance with this memo, unlike [HEVC], a sender MUST NOT send Slice Loss Indication (SLI) or Reference Picture Selection Indication (RPSI), and a receiver SHOULD ignore RPSI and treat a received SLI as a PLI.

8.1. Picture Loss Indication (PLI)

As specified in RFC 4585, Section 6.3.1, the reception of a PLI by a media sender indicates "the loss of an undefined amount of coded video data belonging to one or more pictures". Without having any specific knowledge of the setup of the bitstream (such as use and location of in-band parameter sets, non-IRAP decoder refresh points, picture structures, and so forth), a reaction to the reception of an PLI by a VVC sender SHOULD be to send an IRAP picture and relevant parameter sets; potentially with sufficient redundancy so to ensure correct reception. However, sometimes information about the bitstream structure is known. For example, state could have been established outside of the mechanisms defined in this document that parameter sets are conveyed out of band only, and stay static for the duration of the session. In that case, it is obviously unnecessary to send them in-band as a result of the reception of a PLI. Other examples could be devised based on a priori knowledge of different aspects of the bitstream structure. In all cases, the timing and congestion control mechanisms of RFC 4585 MUST be observed.

8.2. Full Intra Request (FIR)

The purpose of the FIR message is to force an encoder to send an independent decoder refresh point as soon as possible, while observing applicable congestion-control-related constraints, such as those set out in [RFC8082]).

Upon reception of a FIR, a sender MUST send an IDR picture. Parameter sets MUST also be sent, except when there is a priori knowledge that the parameter sets have been correctly established. A typical example for that is an understanding between sender and receiver, established by means outside this document, that parameter sets are exclusively sent out-of-band.

9. Security Considerations

The scope of this Security Considerations section is limited to the payload format itself and to one feature of [VVC] that may pose a particularly serious security risk if implemented naively. The payload format, in isolation, does not form a complete system. Implementers are advised to read and understand relevant security-related documents, especially those pertaining to RTP (see the Security Considerations section in [RFC3550]), and the security of the call-control stack chosen (that may make use of the media type registration of this memo). Implementers should also consider known security vulnerabilities of video coding and decoding implementations in general and avoid those.

Within this RTP payload format, and with the exception of the user data SEI message as described below, no security threats other than those common to RTP payload formats are known. In other words, neither the various media-plane-based mechanisms, nor the signaling part of this memo, seems to pose a security risk beyond those common to all RTP-based systems.

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550], and in any applicable RTP profile such as RTP/AVP [RFC3551], RTP/AVPF [RFC4585], RTP/SAVP [RFC3711], or RTP/SAVPF [RFC5124]. However, as "Securing the RTP Framework: Why RTP Does Not Mandate a Single Media Security Solution" [RFC7202] discusses, it is not an RTP payload format's responsibility to discuss or mandate what solutions are used to meet the basic security goals like confidentiality, integrity and source authenticity for RTP in general. This responsibility lays on anyone using RTP in an application. They can find guidance on available security mechanisms and important considerations in "Options for Securing RTP Sessions" [RFC7201]. The rest of this section discusses the security impacting properties of the payload format itself.

Because the data compression used with this payload format is applied end-to-end, any encryption needs to be performed after compression. A potential denial-of-service threat exists for data encodings using compression techniques that have non-uniform receiver-end computational load. The attacker can inject pathological datagrams

into the bitstream that are complex to decode and that cause the receiver to be overloaded. [VVC] is particularly vulnerable to such attacks, as it is extremely simple to generate datagrams containing NAL units that affect the decoding process of many future NAL units. Therefore, the usage of data origin authentication and data integrity protection of at least the RTP packet is RECOMMENDED, for example, with SRTP [RFC3711].

Like HEVC [RFC7798], [VVC] includes a user data Supplemental Enhancement Information (SEI) message. This SEI message allows inclusion of an arbitrary bitstring into the video bitstream. Such a bitstring could include JavaScript, machine code, and other active content. [VVC] leaves the handling of this SEI message to the receiving system. In order to avoid harmful side effects of the user data SEI message, decoder implementations cannot naively trust its content. For example, it would be a bad and insecure implementation practice to forward any JavaScript a decoder implementation detects to a web browser. The safest way to deal with user data SEI messages is to simply discard them, but that can have negative side effects on the quality of experience by the user.

End-to-end security with authentication, integrity, or confidentiality protection will prevent a MANE from performing media-aware operations other than discarding complete packets. In the case of confidentiality protection, it will even be prevented from discarding packets in a media-aware way. To be allowed to perform such operations, a MANE is required to be a trusted entity that is included in the security context establishment.

10. Congestion Control

Congestion control for RTP SHALL be used in accordance with RTP [RFC3550] and with any applicable RTP profile, e.g., AVP [RFC3551]. If best-effort service is being used, an additional requirement is that users of this payload format MUST monitor packet loss to ensure that the packet loss rate is within an acceptable range. Packet loss is considered acceptable if a TCP flow across the same network path, and experiencing the same network conditions, would achieve an average throughput, measured on a reasonable timescale, that is not less than all RTP streams combined are achieved. This condition can be satisfied by implementing congestion-control mechanisms to adapt the transmission rate, the number of layers subscribed for a layered multicast session, or by arranging for a receiver to leave the session if the loss rate is unacceptably high.

The bitrate adaptation necessary for obeying the congestion control principle is easily achievable when real-time encoding is used, for example, by adequately tuning the quantization parameter. However,

when pre-encoded content is being transmitted, bandwidth adaptation requires the pre-coded bitstream to be tailored for such adaptivity. The key mechanisms available in [VVC] are temporal scalability, and spatial/SNR scalability. A media sender can remove NAL units belonging to higher temporal sublayers (i.e., those NAL units with a high value of TID) or higher spatio-SNR layers until the sending bitrate drops to an acceptable range.

The mechanisms mentioned above generally work within a defined profile and level and, therefore, no renegotiation of the channel is required. Only when non-downgradable parameters (such as profile) are required to be changed does it become necessary to terminate and restart the RTP stream(s). This may be accomplished by using different RTP payload types.

MANEs MAY remove certain unusable packets from the RTP stream when that RTP stream was damaged due to previous packet losses. This can help reduce the network load in certain special cases. For example, MANEs can remove those FUs where the leading FUs belonging to the same NAL unit have been lost or those dependent slice segments when the leading slice segments belonging to the same slice have been lost, because the trailing FUs or dependent slice segments are meaningless to most decoders. MANE can also remove higher temporal scalable layers if the outbound transmission (from the MANE's viewpoint) experiences congestion.

11. IANA Considerations

A new media type, as specified in Section 7.1 of this memo, has been registered with IANA.

12. Acknowledgements

Dr. Byeongdoo Choi is thanked for the video codec related technical discussion and other aspects in this memo. Xin Zhao and Dr. Xiang Li are thanked for their contributions on [VVC] specification descriptive content. Spencer Dawkins is thanked for his valuable review comments that led to great improvements of this memo. Some parts of this specification share text with the RTP payload format for HEVC [RFC7798]. We thank the authors of that specification for their excellent work.

13. References

13.1. Normative References

- [ISO23090-3] ISO/IEC 23090-3, "Information technology - Coded representation of immersive media Part 3 Versatile Video Coding", 2021, <<https://www.iso.org/standard/73022.html>>.
- [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/info/rfc2119>>.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <<https://www.rfc-editor.org/info/rfc3264>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <<https://www.rfc-editor.org/info/rfc3551>>.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <<https://www.rfc-editor.org/info/rfc3711>>.
- [RFC4556] Zhu, L. and B. Tung, "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)", RFC 4556, DOI 10.17487/RFC4556, June 2006, <<https://www.rfc-editor.org/info/rfc4556>>.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <<https://www.rfc-editor.org/info/rfc4585>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.

- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <<https://www.rfc-editor.org/info/rfc5104>>.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <<https://www.rfc-editor.org/info/rfc5124>>.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, DOI 10.17487/RFC5576, June 2009, <<https://www.rfc-editor.org/info/rfc5576>>.
- [RFC8082] Wenger, S., Lennox, J., Burman, B., and M. Westerlund, "Using Codec Control Messages in the RTP Audio-Visual Profile with Feedback with Layered Codecs", RFC 8082, DOI 10.17487/RFC8082, March 2017, <<https://www.rfc-editor.org/info/rfc8082>>.
- [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/info/rfc8174>>.
- [RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <<https://www.rfc-editor.org/info/rfc8866>>.
- [VSEI] "Versatile supplemental enhancement information messages for coded video bitstreams", 2020, <<https://www.itu.int/rec/T-REC-H.274>>.
- [VVC] "Versatile Video Coding, ITU-T Recommendation H.266", 2020, <<http://www.itu.int/rec/T-REC-H.266>>.

13.2. Informative References

- [CABAC] and et al, "Transform coefficient coding in HEVC, IEEE Transactions on Circuits and Systems for Video Technology", DOI 10.1109/TCSVT.2012.2223055, December 2012, <<https://doi.org/10.1109/TCSVT.2012.2223055>>.
- [HEVC] "High efficiency video coding, ITU-T Recommendation H.265", 2019, <<https://www.itu.int/rec/T-REC-H.265>>.

- [MPEG2S] ISO/IEC, "Information technology - Generic coding of moving pictures and associated audio information - Part 1: Systems, ISO International Standard 13818-1", 2013.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974, October 2000, <<https://www.rfc-editor.org/info/rfc2974>>.
- [RFC6184] Wang, Y.-K., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, DOI 10.17487/RFC6184, May 2011, <<https://www.rfc-editor.org/info/rfc6184>>.
- [RFC6190] Wenger, S., Wang, Y.-K., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <<https://www.rfc-editor.org/info/rfc6190>>.
- [RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <<https://www.rfc-editor.org/info/rfc7201>>.
- [RFC7202] Perkins, C. and M. Westerlund, "Securing the RTP Framework: Why RTP Does Not Mandate a Single Media Security Solution", RFC 7202, DOI 10.17487/RFC7202, April 2014, <<https://www.rfc-editor.org/info/rfc7202>>.
- [RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <<https://www.rfc-editor.org/info/rfc7656>>.
- [RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <<https://www.rfc-editor.org/info/rfc7667>>.
- [RFC7798] Wang, Y.-K., Sanchez, Y., Schierl, T., Wenger, S., and M. M. Hannuksela, "RTP Payload Format for High Efficiency Video Coding (HEVC)", RFC 7798, DOI 10.17487/RFC7798, March 2016, <<https://www.rfc-editor.org/info/rfc7798>>.
- [RFC7826] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, Ed., "Real-Time Streaming Protocol Version 2.0", RFC 7826, DOI 10.17487/RFC7826, December 2016, <<https://www.rfc-editor.org/info/rfc7826>>.

Appendix A. Change History

To RFC Editor: PLEASE REMOVE THIS SECTION BEFORE PUBLICATION

draft-zhao-payload-rtp-vvc-00 initial version

draft-zhao-payload-rtp-vvc-01 editorial clarifications and
corrections

draft-ietf-payload-rtp-vvc-00 initial WG draft

draft-ietf-payload-rtp-vvc-01 VVC specification update

draft-ietf-payload-rtp-vvc-02 VVC specification update

draft-ietf-payload-rtp-vvc-03 VVC coding tool introduction
update

draft-ietf-payload-rtp-vvc-04 VVC coding tool introduction
update

draft-ietf-payload-rtp-vvc-05 reference update and adding
placement for open issues

draft-ietf-payload-rtp-vvc-06 address editor's note

draft-ietf-payload-rtp-vvc-07 address editor's notes

draft-ietf-payload-rtp-vvc-08 address editor's notes

draft-ietf-payload-rtp-vvc-09 address editor's notes

draft-ietf-payload-rtp-vvc-10 address editor's notes

draft-ietf-payload-rtp-vvc-11 address editor's notes

draft-ietf-payload-rtp-vvc-12 address editor's notes

draft-ietf-payload-rtp-vvc-13 address editor's notes

draft-ietf-payload-rtp-vvc-14 address 2nd WGLC comments

Authors' Addresses

Shuai Zhao
Tencent
2747 Park Blvd
Palo Alto, 94588
United States of America
Email: shuai.zhao@ieee.org

Stephan Wenger
Tencent
2747 Park Blvd
Palo Alto, 94588
United States of America
Email: stewe@stewe.org

Yago Sanchez
Fraunhofer HHI
Einsteinufer 37
10587 Berlin
Germany
Email: yago.sanchez@hhi.fraunhofer.de

Ye-Kui Wang
Bytedance Inc.
8910 University Center Lane
San Diego, 92122
United States of America
Email: yekui.wang@bytedance.com

Miska M. Hannuksela
Nokia Technologies
Hatanpään valtatie 30
FI-33100 Tampere
Finland
Email: miska.hannuksela@nokia.com

AVTCORE
Internet-Draft
Intended status: Standards Track
Expires: 12 January 2022

S. Garcia Murillo
CoSMo
Y. Fablet
Apple Inc.
A. Gouaillard
CoSMo
J. Uberti
Clubhouse
11 July 2021

Multi Codec RTP payload format
draft-murillo-avtcore-multi-codec-payload-format-01

Abstract

RTP Media Chains usually rely on piping encoder output directly to packetizers. Media packetization formats often support a specific codec format and optimize RTP packets generation accordingly. With the development of Selective Forward Unit (SFU) solutions, RTP Media Chains used in WebRTC solutions are increasingly relying on application-specific transforms that sit between encoder and packetizer on one end and between depacketizer and decoder on the other end. These transforms are typically encrypting media content so that the media content is not readable from the SFU, for instance using [SFrame] or [WebRTCInsertableStreams]. In that context, RTP packetizers can no longer expect to use packetization formats that mandate media content to be in a specific codec format. This document provides a solution to that problem by describing a RTP packetization format that can be used for many media content, and how to negotiate use of this format. This document also describes a solution to allow SFUs to continue performing packet routing on top of this RTP packetization format.

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 12 January 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Goals	6
3. RTP Packetization	6
4. Payload Multiplexing	7
5. SDP Negotiation	8
6. SFU Packet Selection	9
7. Sender Processing Rules	10
8. Redundancy Techniques Considerations	10
8.1. Retransmission Techniques	10
8.2. Forward Error Correction (FEC) Techniques	11
8.3. Redundant Audio Data Techniques	11
9. Alternatives	11
9.1. Generic Packetization With In-Payload APT	12
9.2. A Payload Type for Generic Packetization AND Media Format	12
9.3. A RTP Header To Choose Packetization	13
10. Security Considerations	14
11. IANA Considerations	14
11.1. Registration of audio/generic	14
12. Registration of video/generic	15
13. References	15
13.1. Normative References	15
13.2. Informative References	16
Authors' Addresses	17

1. Introduction

As per Figure 1 of [RFC7656], a Media Packetizer transforms a single Encoded Stream into one or several RTP packets. The Encoded Stream is coming straight from the Media Encoder and is expected to follow the format produced by the Media Encoder. A number of Media Packetizer formats have been designed to process a specific format produced by Media Encoder. For instance [RFC6184] is dedicated to the processing of content produced by H.264 Media Encoders, and generates packets following NALUs organization.

WebRTC applications are increasingly deploying end-to-end encryption solutions on top of RTP Media Chains. End-to-end encryption is implemented by inserting application-specific Media Transformers between Media Encoder and Media Packetizer on the sending side, and between Media Depacketizer and Media Decoder on the receiving side, as described in Figure 1 and Figure 2. To support end-to-end encryption, Media Transformers can use the [SFrame] format. In browsers, Media Transformers are implemented using [WebRTCInsertableStreams], for instance by injecting JavaScript code provided by web pages.

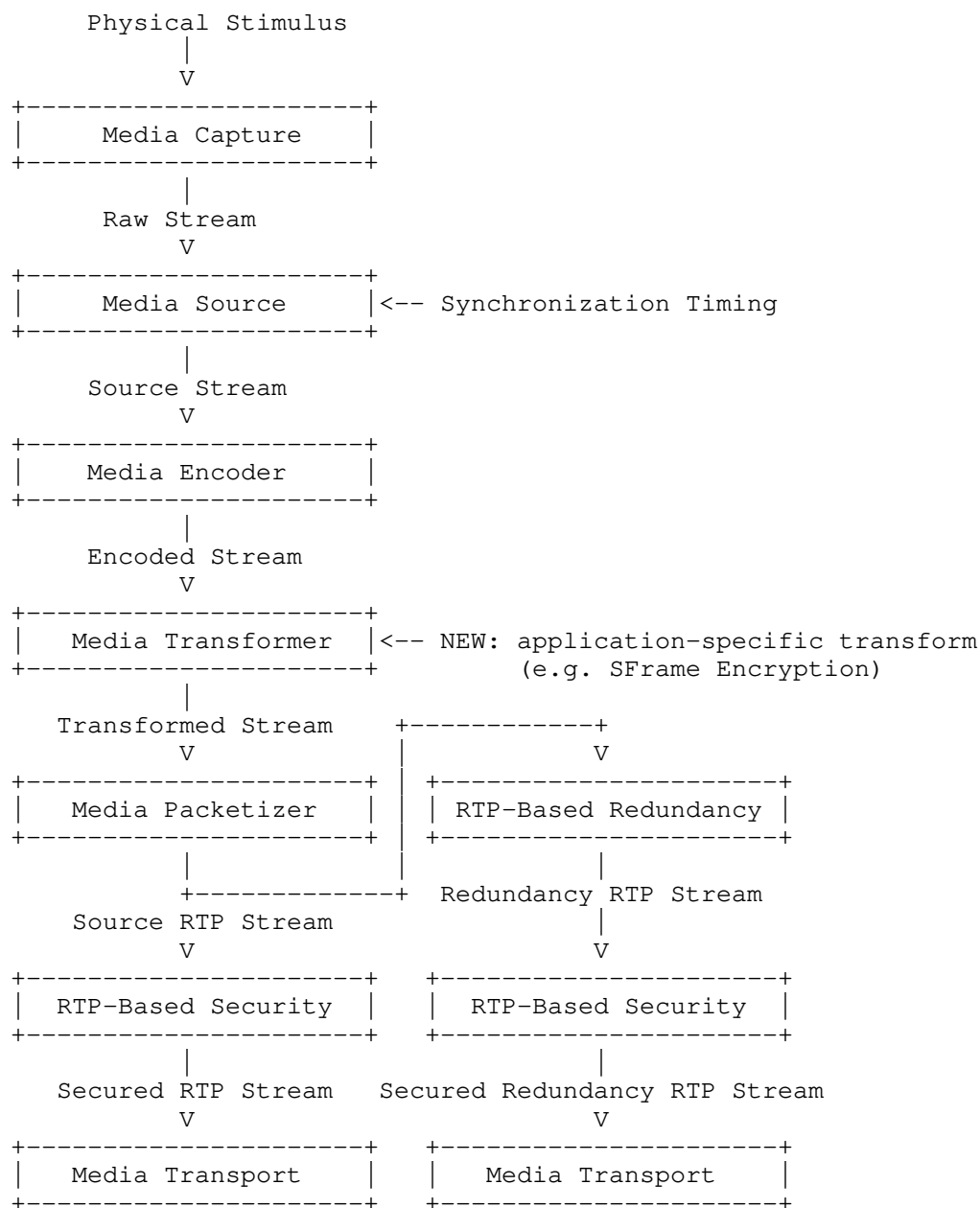
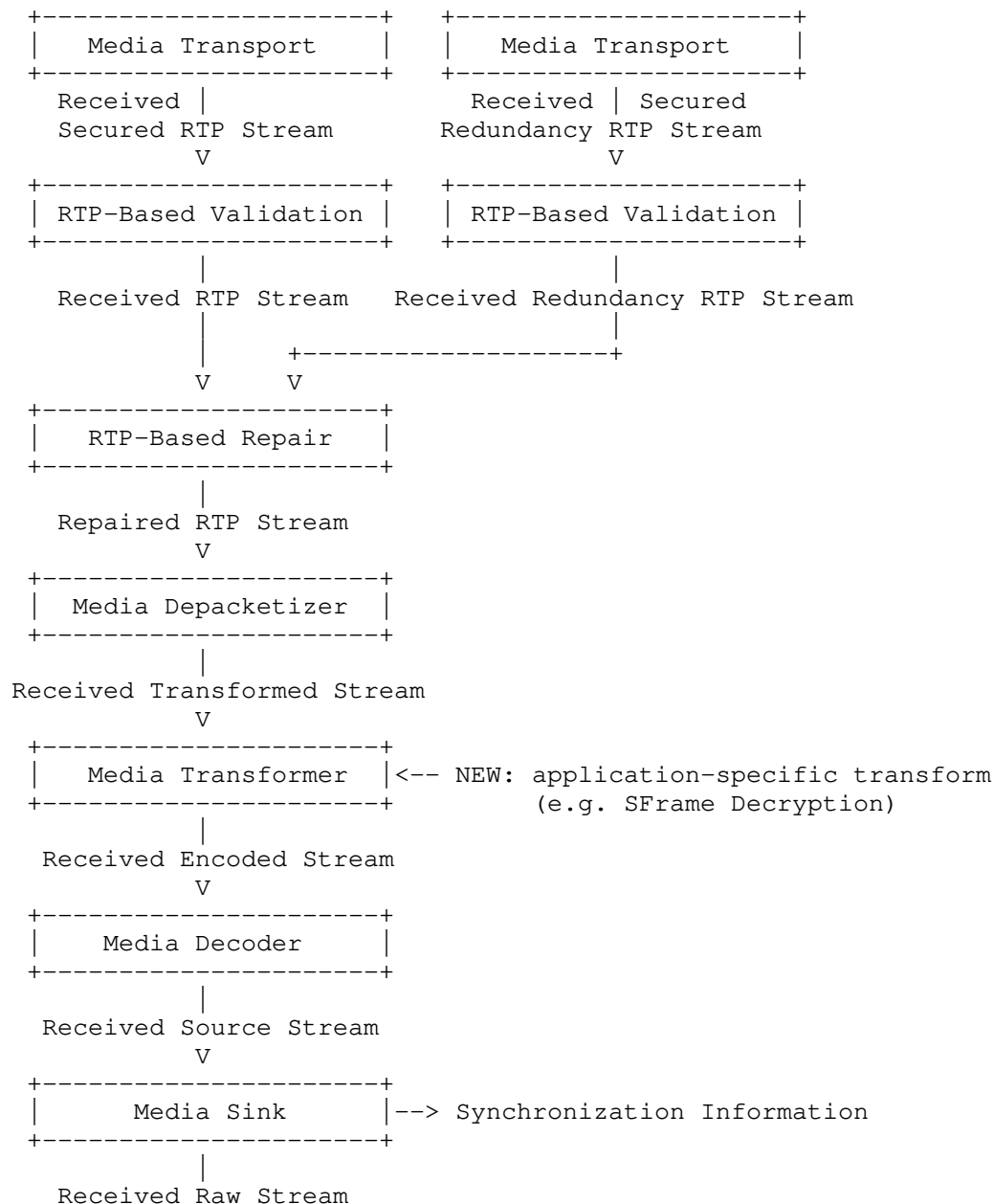


Figure 1: Sender side concepts in the Media Chain with application-level Media Transform

These RTP packets are sent over the wire to a receiver media chain matching the sender side, reaching the Media Depacketizer that will reconstruct the Encoded Stream before passing it to the Media Decoder.



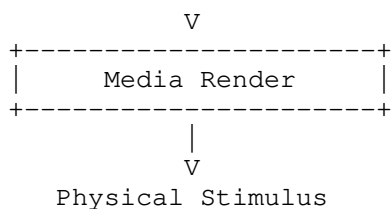


Figure 2: Receiver side concepts in the Media Chain with application-level Media Transform

This packetization does not change how the mapping between one or several encoded or dependant streams are mapped to the RTP streams or how the synchronization sources(s) (SSRC) are assigned.

Given the use of post-encoder application-specific transforms, the whole Media Chain needs to be made aware of it. This includes the sender post-transform Media Chain, Media Transport intermediaries (SFUs typically) and receiver pre-transform Media Chain.

As these transforms can alter Encoded Streams in any possible way, the use of codec-specific Media Packetizers like [RFC6184] on Transformed Stream may be suboptimal on sender side. It may also be problematic on the receiving side in case codec-specific processing is done prior the Media Transformer. Media Transport intermediaries are often looking at the Media Content itself to fuel their packet selection algorithms.

2. Goals

The objective of this document is to support inserting any application-specific transform between encoders and packetizers in the Media Chain. For that purpose, this document will:

1. Provide a packetization format that supports multiple media content used by WebRTC applications (audio compressed by Opus, video compressed by H264 or VP8, encrypted content...) that allows reuse of existing RTP mechanisms in place in WebRTC applications such as RTX, RED or FEC.
2. Provide a way to negotiate use of this packetization format between sender and receiver, with minimum impact on existing negotiation approaches.
3. Provide a side-channel information so that network intermediaries (SFU in particular) can do their existing packet routing strategies without inspecting the media content.

3. RTP Packetization

This packetizer, by design, is not expected to understand the format of the media to transmit. The unit used by the packetizer to do processing is called a frame in the remainder of the document.

It is the responsibility of the application using the packetizer to group media content in meaningful frames. In the common case of a video codec, the packetizer frame is the frame in byte format (h264 annex b for example) generated by the encoder.

If the application wants to transform encoded content, the application needs to split the encoded content into frames prior the transform. Each frame is then transformed independently, for instance encrypted using [SFrame]. The content of each transformed frame is then processed by the packetizer.

In the case of a video codec supporting spatial scalability, each spatial layer MUST be split in its own frame by the application before passing it to the packetizer.

When the packetizer receives a frame from the application, it MUST fragment the frame content in multiple RTP packets to ensure packets do not exceed the network maximum transmission unit. The content of the frame will be treated as a binary blob by the packetizer, so the decision about the boundaries of each fragment is decided arbitrarily by the packetizer. The packetizer or any relying server MUST NOT modify the frame content and concatenating the RTP payload of the RTP packets for each frame MUST produce the exact binary content of the input frame content.

The marker bit of each RTP packet in a frame MUST be set according to the audio and video profiles specified in [RFC3551].

The spatial layer frames are sent in ascending order, with the same RTP timestamp, and only the last RTP packet of the last spatial layer frame will have the marker bit set to 1.

4. Payload Multiplexing

In order to reduce the number of payload type in the SDP exchange, a single payload type code for this multi-codec packetization can be used for all negotiated media formats that the multi-codec packetization supports. That requires to identify the original payload type code of the frame negotiated media format, called the associated payload type (APT) hereunder. The APT value is the payload type code of the associated format passed to the multi-codec Media Packetizer before any transformation is applied.

The APT value is sent in a dedicated header extension. The payload of this header extension can be encoded using either the one-byte or two-byte header defined in [RFC5285]. Figures 3 and 4 show examples with each one of these examples.

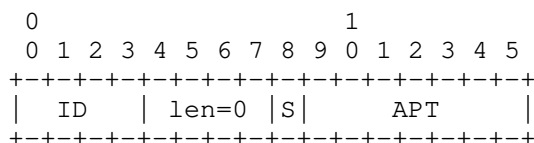


Figure 3: Frame associated payload type encoding using the One-Byte header format

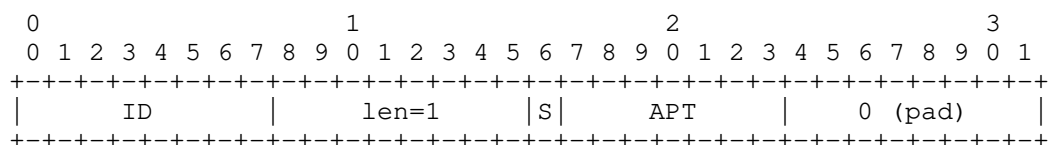


Figure 4: Frame associated payload type encoding using the Two-Byte header format

The APT value is the associated payload type value. The S bit indicates if the media stream can be forwarded safely starting from this RTP packet. Typically, it will be set to 1 on the first RTP packet of an intra video frame and in all RTP audio packets.

Receivers MUST be ready to receive RTP packets with different associated payload types in the same way they would receive different payload type codes on the RTP packets.

The URI for declaring this header extension in an extmap attribute is "urn:ietf:params:rtp-hdrex:associated-payload-type".

5. SDP Negotiation

To use the multi-codec packetization, the SDP Offer/Answer exchange MUST negotiate: - The payload type of the negotiated codec format - The multi-codec payload type - The associated payload type header extension

Only the negotiated payload types are allowed to be used as associated payload types. Figure 5 illustrates a SDP that negotiates exchange of video using either VP8 or VP9 codecs with the possibility to use the multi-codec packetization. In this example, RTX is also negotiated and will be applied normally on each associated payload type.

```
m=video 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=setup:actpass
a=mid:1
a=extmap:1 urn:ietf:params:rtp-hdext:sdes:mid
a=extmap:2 urn:ietf:params:rtp-hdext:sdes:rtp-stream-id
a=extmap:3 urn:ietf:params:rtp-hdext:sdes:repaired-rtp-stream-id
a=extmap:4 urn:ietf:params:rtp-hdext:associated-payload-type
a=sendrecv
a=rtpmap:96 vp9/90000
a=rtpmap:97 vp8/90000
a=rtpmap:98 generic/90000
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=96
a=rtpmap:100 rtx/90000
a=fmtp:100 apt=97
a=rtpmap:101 rtx/90000
a=fmtp:101 apt=98
```

Figure 5: SDP example negotiating the multi-codec payload type and related header extension for video

6. SFU Packet Selection

SFUs need to have a basic understanding of each frame they receive so they can decide to forward it or not and to which endpoint. They might need similar information to support media content recording. This information is either generic to a group of frames (called a stream hereafter) or specific to each frame.

The information is transmitted as a RTP header extension as the RTP packet payload should be treated as opaque by the SFU. This is especially necessary if the payload is end-to-end encrypted. The amount of information should be limited to what is strictly necessary to the SFU task since it is not always as trusted as individual peers.

For audio, configuration information such as Opus TOC might be useful. For video, configuration information might include: - Stream configuration information: resolution, quality, frame rate... - Codec specific configuration information: codec profile like profile_idc... - Frame specific information: whether the stream is decodable when starting from this frame, whether the frame is skippable...

For video content, this information is sent using a Dependency Descriptor header extension. In that case, the first RTP packet of the frame will have its `start_of_frame` equal to 1 and the last packet will have its `end_of_frame` equal to 1.

7. Sender Processing Rules

The sender identifies the use of the multi-codec payload format by using the `urn:ietf:params:rtp-hdext:associated-payload-type` extension. When doing so, the sender follows these additional rules:

- For audio content, the associated payload type MUST reference an audio codec in the supported audio codec list. The supported audio codec list contains the audio codecs enumerated in [RFC7874]. This list may be extended in future versions of this specification.
- For video content, H.264 and VP8 are supported as described in [RFC7742], as well as VP9 and AV.1. In the case scalable video coding is used, the sender MUST generate a Dependency Descriptor header extension. This requires the associated payload type to reference a video codec that can be described using the Dependency Descriptor header extension. This also requires the sender to split the video encoder output in frames that can each be described using the Dependency Descriptor header extension.

These rules apply to both the originator of the content as well as SFUs that might route the content to end receivers.

8. Redundancy Techniques Considerations

The solution described in this document is expected to integrate well with the existing RTP ecosystem. This section describes how the multi-codec packetizer can be used jointly with existing techniques that allow to mitigate unreliable transports.

8.1. Retransmission Techniques

[RFC4588] defines a retransmission payload format (RTX) that can be used in case of packet loss. As defined in [RFC4588], RTX is able to handle any payload format, including the format described in this document. Given RTX preserves both RTP packet payload and headers, the receiver will be able to identify the payload type of the recovered packet and whether multi-codec packetization is used. RTX will also allow recovering RTP header extensions that convey information on the media content itself.

8.2. Forward Error Correction (FEC) Techniques

FEC is another technique used in RTP Media Chains to protect media content against packet loss. [RFC5109] defines such a payload format used to transmit FEC for specific packets protection.

FEC may protect some parts of the media content more than others. For instance, intra video frame encoded data or important network abstraction layer units (NALUs) like SPS/PPS may be more protected. With a post-encoder transform and the use of a multi-codec packetization, the granularity of the recovery mechanism is no longer at the NALU level but at the level of the frame generated by the post-encoder transform. In case a SVC codec is used, each spatial layer will be processed as an independent frame. In that case, base layers can be protected more heavily than higher resolution layers.

8.3. Redundant Audio Data Techniques

As defined in [RFC7656] RTP-based redundancy is defined here as a transformation that generates redundant or repair packets sent out as a Redundancy RTP Stream to mitigate Network Transport impairments, like packet loss and delay.

[RFC2198] defines a payload format for sending the same audio data encoded multiple times at different quality levels. This allows to use a lower quality encoding of the audio data, should the higher quality encoding of the audio data is lost during the transmission.

If a Media Transformation is in use, both the primary and redundant encoding must be transformed independently and the redundant packet created normally. As the RTP headers present in the redundant packet are only applicable to the primary encoding, if the payload type for a redundant encoding block is mapped to the multi-codec packetizer, the value of the associated payload type for the primary encoding is applied to the redundant encoding block as well.

9. Alternatives

Various alternatives can be used to implement and negotiate multi-codec packetization. This section describes a few additional alternatives. This section is to be removed before finalization of the document.

9.1. Generic Packetization With In-Payload APT

Instead of using a RTP header extension to convey the APT value, it is prepended in the RTP payload itself. As the value cannot change for a whole frame, its value is prepended to the first packet generated of the frame only. This removes the need to negotiate a dedicated header extension, but may require the SFU to update the payload when sending or recording content.

9.2. A Payload Type for Generic Packetization AND Media Format

The payload type is negotiated in the SDP so as to identify both the negotiated codec format and the multi-codec packetization use. There is no network cost but this increases the number of payload types used in the SDP.

```
m=video 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=setup:actpass
a=mid:1
a=extmap:1 urn:ietf:params:rtp-hdext:sdes:mid
a=extmap:2 urn:ietf:params:rtp-hdext:sdes:rtp-stream-id
a=extmap:3 urn:ietf:params:rtp-hdext:sdes:repaired-rtp-stream-id
a=sendrecv
a=rtpmap:96 vp9/90000
a=rtpmap:97 generic/90000
a=fmtp:97 apt=96
a=rtpmap:98 vp8/90000
a=rtpmap:99 generic/90000
a=fmtp:99 apt=98
a=rtpmap:100 rtx/90000
a=fmtp:100 apt=96
a=rtpmap:101 rtx/90000
a=fmtp:101 apt=97
a=rtpmap:102 rtx/90000
a=fmtp:102 apt=98
a=rtpmap:103 rtx/90000
a=fmtp:103 apt=99
```

Figure 6: SDP example negotiating a payload type for format and multi-codec packetization

A variation of this approach is to consider defining several multi-codec payload types, each of them having an identified codec format.

```
m=video 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=setup:actpass
a=mid:1
a=extmap:1 urn:ietf:params:rtp-hdext:sdes:mid
a=extmap:2 urn:ietf:params:rtp-hdext:sdes:rtp-stream-id
a=extmap:3 urn:ietf:params:rtp-hdext:sdes:repaired-rtp-stream-id
a=sendrecv
a=rtpmap:96 generic/90000
a=fmtp:96 codec=vp9
a=rtpmap:97 generic/90000
a=fmtp:97 codec=vp8
a=rtpmap:98 rtx/90000
a=fmtp:98 apt=96
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=97
```

Figure 7: Alternative SDP example negotiating a payload type for format and multi-codec packetization

9.3. A RTP Header To Choose Packetization

A RTP header extension can be used to flag content as opaque so that the receiver knows whether to use or not the multi-codec packetization. As for the API header extension, the RTP header extension may not need to be sent for every packet, it could for instance be sent for the first packet of every intra video frame. The main advantage of this approach is the reduced impact on SDP negotiation.

```
m=video 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=setup:actpass
a=mid:1
a=extmap:1 urn:ietf:params:rtp-hdext:sdes:mid
a=extmap:2 urn:ietf:params:rtp-hdext:sdes:rtp-stream-id
a=extmap:3 urn:ietf:params:rtp-hdext:sdes:repaired-rtp-stream-id
a=extmap:4 urn:ietf:params:rtp-hdext:multi-codec-packetization-use
a=sendrecv
a=rtpmap:96 vp9/90000
a=rtpmap:97 vp8/90000
a=rtpmap:98 rtx/90000
a=fmtp:98 apt=96
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=97
```

Figure 8: SDP example negotiating multi-codec packetization as RTP header extension

10. Security Considerations

RTP packets using the payload format defined in this specification are subject to the general security considerations discussed in [RFC3550]. It is not expected that the proposed solution presented in this document can create new security threats. The use and implementation of RTP Media Chains containing Media Transformers needs to be done carefully. It is important to refer to the security considerations discussed in [SFrame] and [WebRTCInsertableStreams]. In particular Media Transformers on the receiver side need to be prepared to receive arbitrary content, like decoders already do. Similarly, since Media Transformers can be implemented as JavaScript in browsers, RTP Packetizers should be prepared to receive arbitrary content.

11. IANA Considerations

Two new media subtypes have been registered with IANA, as described in this section.

11.1. Registration of audio/generic

Type name: audio

Subtype name: generic

Required parameters: none

Optional parameters: none

Encoding considerations: This format is framed (see Section 4.8 in the template document) and contains binary data.

Security considerations: TBD.

Interoperability considerations: TBD

Published specification: TBD.

Applications that use this media type: TBD.

Additional information: none

Intended usage: COMMON

Restrictions on usage: TBD

Author:

Change controller:

12. Registration of video/generic

Type name: video

Subtype name: generic

Required parameters: none

Optional parameters: none

Encoding considerations: This format is framed (see Section 4.8 in the template document) and contains binary data.

Security considerations: TBD.

Interoperability considerations: TBD

Published specification: TBD.

Applications that use this media type: TBD.

Additional information: none

Intended usage: COMMON

Restrictions on usage: TBD

Author:

Change controller:

13. References

13.1. 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/info/rfc2119>>.

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <<https://www.rfc-editor.org/info/rfc3551>>.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <<https://www.rfc-editor.org/info/rfc3711>>.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <<https://www.rfc-editor.org/info/rfc4566>>.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <<https://www.rfc-editor.org/info/rfc5285>>.
- [RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <<https://www.rfc-editor.org/info/rfc7656>>.
- [RFC8285] Singer, D., Desineni, H., and R. Even, Ed., "A General Mechanism for RTP Header Extensions", RFC 8285, DOI 10.17487/RFC8285, October 2017, <<https://www.rfc-editor.org/info/rfc8285>>.

13.2. Informative References

- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J.C., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, DOI 10.17487/RFC2198, September 1997, <<https://www.rfc-editor.org/info/rfc2198>>.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <<https://www.rfc-editor.org/info/rfc4588>>.

- [RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <<https://www.rfc-editor.org/info/rfc5109>>.
- [RFC6184] Wang, Y.-K., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, DOI 10.17487/RFC6184, May 2011, <<https://www.rfc-editor.org/info/rfc6184>>.
- [RFC6464] Lennox, J., Ed., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, DOI 10.17487/RFC6464, December 2011, <<https://www.rfc-editor.org/info/rfc6464>>.
- [RFC6465] Ivov, E., Ed., Marocco, E., Ed., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, DOI 10.17487/RFC6465, December 2011, <<https://www.rfc-editor.org/info/rfc6465>>.
- [RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, DOI 10.17487/RFC6904, April 2013, <<https://www.rfc-editor.org/info/rfc6904>>.
- [RFC7742] Roach, A.B., "WebRTC Video Processing and Codec Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016, <<https://www.rfc-editor.org/info/rfc7742>>.
- [RFC7874] Valin, JM. and C. Bran, "WebRTC Audio Codec and Processing Requirements", RFC 7874, DOI 10.17487/RFC7874, May 2016, <<https://www.rfc-editor.org/info/rfc7874>>.
- [SFrame] "Secure Frame (SFrame)", n.d., <<https://tools.ietf.org/html/draft-omara-sframe>>.
- [WebRTCInsertableStreams]
"WebRTC Insertable Media using Streams", n.d., <<https://w3c.github.io/webrtc-insertable-streams>>.

Authors' Addresses

Sergio Garcia Murillo
CoSMo

Email: sergio.garcia.murillo@cosmosoftware.io

Youenn Fablet
Apple Inc.

Email: youenn@apple.com

Alex Gouaillard
CoSMo

Email: alex.gouaillard@cosmosoftware.io

Justin Uberti
Clubhouse

Email: justin@uberti.name