Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Standards Track
Expires: 4 September 2025

M. Engelbart
J. Ott
Technical University Munich
L. Kondrad
Nokia Technologies

3 March 2025

# RTP Payload Format for Geometry-based Point Cloud Compression
draft-engelbart-avtcore-rtp-gpcc-01

## Abstract

This memo describes an RTP payload format for geometry-based point cloud compression (G-PCC) ([ISO.IEC.23090-9]). The RTP payload format defined in this document supports the packetization of one or more data units in an RTP packet payload and the fragmentation of a single data unit into multiple RTP packets. This memo also describes congestion control for a practical solution for the real-time streaming of point clouds. Finally, the document defines parameters that may be used to select optional features of the payload format or signal properties of the RTP stream. The parameters can be used with the Session Description Protocol (SDP).

## About This Document

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

The latest revision of this draft can be found at https://mengelbart.github.io/draft-engelbart-avtcore-rtp-gpcc/draft-engelbart-avtcore-rtp-gpcc.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-engelbart-avtcore-rtp-gpcc/.

Discussion of this document takes place on the avtcore Working Group mailing list (mailto:avt@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/avt/. Subscribe at https://www.ietf.org/mailman/listinfo/avt/.

Source for this draft and an issue tracker can be found at https://github.com/mengelbart/draft-engelbart-avtcore-rtp-gpcc.

## Status of This Memo

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

This Internet-Draft will expire on 4 September 2025.

## Copyright Notice



## Table of Contents

## 1. Introduction

This document describes the Real-time Transport Protocol (RTP) payload format for Geometry-based point cloud compression as described in [ISO.IEC.23090-9]. Point clouds are commonly used to represent three-dimensional scans of environments or objects. The payload format enables the streaming of compressed point clouds in real-time applications. It applies to various use cases, such as autonomous vehicles and virtual reality.

In addition to describing the payload format, this document also includes examples of the payload format, provides guidance on how to negotiate the parameters to be used with the format, and discusses congestion control and rate adaptation of point cloud streaming applications.

### 1.1. Background on Point Clouds

A point cloud is a data structure used to represent three-dimensional scenes. Point clouds consist of a list of points in three-dimensional space where each point may optionally be associated with zero or more attributes, such as a color or reflectance. Point clouds have diverse use cases. For example, a point cloud can store a 3D representation of the vehicle's surrounding environment in autonomous cars. Another example is object scanning for the archival of historical objects or the creation of digital twins of real-world objects for further analysis. Point clouds are either acquired passively using multiple camera setups or actively, e.g., using a Lidar sensor to measure distances using beams of light.

## 1.2. Overview of the G-PCC Codec

Point clouds can contain large amounts of data, which requires efficient compression to reduce storage and transmission costs. The Moving Picture Experts Group (MPEG) has published a Geometry-based Point Cloud Compression (G-PCC) standard in [ISO.IEC.23090-9]. The G-PCC codec takes as input an unordered list of points, optional attributes, and metadata. All points have the same number, type, and order of attributes. Attributes can, for example, be the color, opacity, reflectance, or frame number of the associated point. Compression is defined separately for geometry and attributes. The attribute coding depends on the decoded geometry. Thus, geometry encoding and decoding happen before attribute encoding and decoding.

G-PCC users can choose between two methods for geometry coding and three for attribute coding. The geometry coding method called _Octree Coding_ is a general compression method, while _Predictive tree_ coding specifically targets low latency applications. The methods for attribute coding are called Region Adaptive Hierarchical Transform (RAHT) coding, Predicting Transform, and Lifting Transform.

The output of the encoding process is a sequence of Data Units (DUs). Each DU has a type that describes its layout. Table 1 lists the ten different types of DUs defined in [ISO.IEC.23090-9].

| Type | Description |
|---|---|
| 0 | Sequence parameter set data unit |
| 1 | Geometry parameter set data unit |
| 2 | Geometry data unit |
| 3 | Attribute parameter set data unit |
| 4 | Attribute data unit |
| 5 | Tile inventory data unit |
| 6 | Frame boundary marker data unit |
| 7 | Defaulted attribute data unit |
| 8 | Frame-specific attribute properties data unit |
| 9 | User data data unit |

Table 1: G-PCC data unit types

Sequence Parameter Set (SPS), Geometry Parameter Set (GPS), and Attribute Parameter Set (APS) hold the coding parameters of a point cloud sequence, the geometry coding in use, and the attribute coding in use, respectively.

Geometry and attribute data units contain the coded representation of points geometry information and points attributes information. Geometry and attribute data units hold references to their associated parameter sets, and each referenced parameter set must be available before decoding of the data unit is possible.

Coded point clouds do not have dependencies between frames, i.e., decoding a point cloud frame is always possible without depending on a previous or following frame in a sequence. However, future versions of G-PCC might support inter-frame prediction.

Annex A of [ISO.IEC.23090-9] describes profiles and levels of the G-PCC codec. Decoders can support different profiles and levels. Profiles are subsets of algorithmic features of the codec specification. A decoder that supports a specific profile must be able to decode a bitstream conforming to that profile.

## 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

This document uses the definitions of [ISO.IEC.23090-9]. The following terms and abbreviations, defined in [ISO.IEC.23090-9], are repeated here for convenience.

### 3.1. General G-PCC related terms

*Attribute*
: Scalar or vector property associated with each _point_ in a _point cloud_.

*Bitstream*
: A sequence of bits.

*Bounding Box*
: Axis-aligned cuboid defining a spatial region that bounds a set of _points_.

*Coded Point Cloud Frame*
: Coded representation of a _point cloud frame_.

*Data unit*
: Sequence of bytes conveying a single _syntax structure_ of known length.

*Geometry*
: _Point positions_ associated with a set of _points_.

*Point*
: Fundamental element of a _point cloud_ comprising a position specified as _Cartesian coordinates_ and zero or more _attributes_.

*Point Cloud*
: Unordered list of _points_.

*Point Cloud Frame*
: _Point cloud_ in a _point cloud sequence_.

*Point Cloud Sequence*
Sequence of one or more _pont clouds_.

*Poisition*
Three dimensional coordinates of a _point_.

*Slice*
_Geometry_ and _attributes_ for part of, or an entire, _coded point cloud frame_.

*Syntax element*
Element of data represented in the _bitstream_.

*Syntax structure*
Zero or more _syntax elements_ present together in the _bitstream_ in a specified order.

*Tile*
Set of _slices_ identified by a common slice_tag _syntax element_ value whose _geometry_ should be contained within a _bounding box_ specified in a tile inventory _data unit_.

## 3.2. Abbreviations

*ADU*
Attribute Data Unit

*APS*
Attribute Parameter Set

*DU*
Data Unit

*GDU*
Geometry Data Unit

*GPS*
Geometry Parameter Set

*SPS*
Sequence Parameter Set

*TLV*
Type-Length-Value

# 4. Payload Format

This section describes the details of the RTP payload format. Section 4.2 describes the usage of the standard RTP header fields, Section 4.3 describes the details of the payload header following the RTP header, and Section 4.4 gives details about available packetization modes and the specifics of their respective formats.

## 4.1. Transmission Modes

*TODO*: Do we need this section on transmission modes similar to other payload formats to define SRST, MRST, and MRMT?

## 4.2. RTP Header Usage

The format of the RTP header is specified in [RFC3550] and replicated in Figure 1 for convenience. This payload format uses the fields of the header in a manner consistent with that specification.

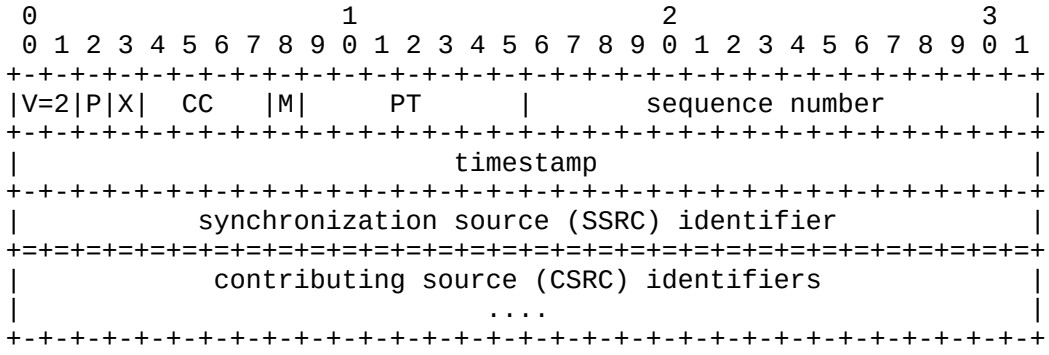


Figure 1: RTP Header format as specified in RFC 3550

*TODO*: if we add a dedicated subsection on SRST/SRMT/MRMT, add the following statement:

When MRST or MRMT is in use, if an access unit appears in multiple RTP streams, the marker bit is set on each RTP stream’s last packet of the access unit.

*Marker bit (M): 1 bit*
Set to 1 for the last packet of a point cloud frame carried in the current RTP stream.

*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. The sampling timestamp of the content is the reconstruction time. It denotes the earliest sampling time of all points in the point cloud frame to which the data unit transmitted in the packet belongs. If the data unit has no timing properties (e.g., parameter sets), the RTP timestamp is set to the RTP timestamp of the first data unit that references the parameter set. If the packet is an aggregation unit packet, all data units MUST have the same timestamp.

*Synchronization source (SSRC): 32 bits*
Used to identify the source of the RTP packets.

## 4.3. Payload Header Usage

The first bytes of the payload of an RTP packet are referred to as the payload header. The payload header consists of a packet type and unit type field.

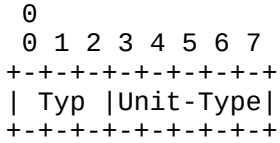


Figure 2: General Payload Header

*Typ: 3 bits*
Type of the packet. This field indicates if the packet is a single unit packet, an aggregation unit packet, or a fragmentation unit packet. See Section 4.4 for details.

*Unit-Type: 5 bits*
The type of the following data unit. This field specifies the type field from the type-length-value encapsulation defined in Annex-B of G-PCC [ISO.IEC.23090-9] and shown in Table 1.

### 4.4. Payload Structures

Three different RTP packet payload structures are specified: Single Unit Packets, Aggregation Unit Packets, and Fragmentation Unit Packets. A receiver can identify the payload structure by the type field of the payload header. The type field indicates whether the unit is a single unit, an aggregation unit, or the first, last, or middle part of a fragmentation unit packet. The type field MUST be set according to Table 2.

| Type | Description |
|---|---|
| 000 | Single Unit Packet Section 4.4.1 |
| 001 | Aggregation Unit Packet Section 4.4.2 |
| 010 | First packet of an fragmentation unit packet Section 4.4.3 |
| 011 | Fragmentation unit packet Section 4.4.3 |
| 100 | Last packet of an fragmentation unit packet Section 4.4.3 |
| 101 | Reserved |
| 110 | Reserved |
| 111 | Reserved |

Table 2: Type field values

The following sections explain the details and variations from this format for the three packet types.

#### 4.4.1. Single Unit

A Single Unit packet contains only one data unit. The first byte in the packet is the payload header, followed by the payload data. The data unit extends until the end of the packet. The packet type field of a single unit packet MUST be set to zero (binary 000).

Figure 3 shows an example of a single unit packet.

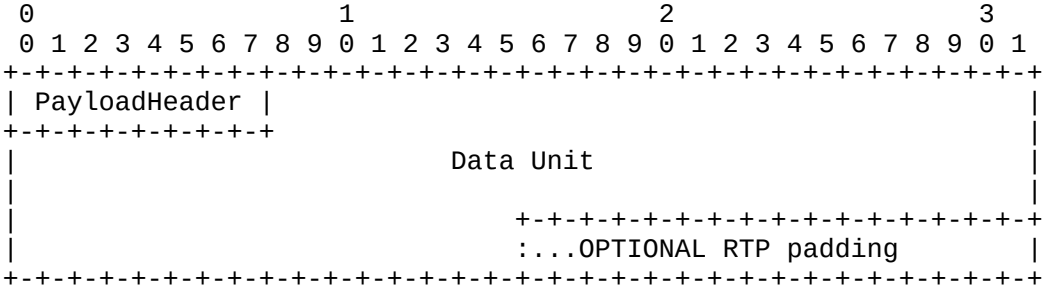


Figure 3: Single unit packet payload

### 4.4.2. Aggregation Packet

An aggregation packet contains two or more DUs. Aggregation packets can reduce packetization and header overhead for small DUs such as parameter sets. Each DU is prefixed with a separate payload header and an additional length field. The length field is 16-bit unsigned integer indicating the length of the following DU in bytes.

An aggregation packet MUST carry at least two DUs. Aggregation packets MAY carry more than two DUs. The total amount of data in an aggregation packet MUST fit into an IP packet, and the size SHOULD be chosen so that the resulting IP packet is smaller than the MTU size in order to avoid IP layer fragmentation. An aggregation packet MUST NOT contain any fragmentation units.

An aggregation packet MUST NOT carry DUs of different point cloud frames, i.e., all DUs included in an aggregation packet MUST have the same timestamp.

The packet type field of every DU in an aggregation packet MUST be set to the value one (binary 001).

Figure 4 shows the extended payload header format of syntax structures in aggregation packets.

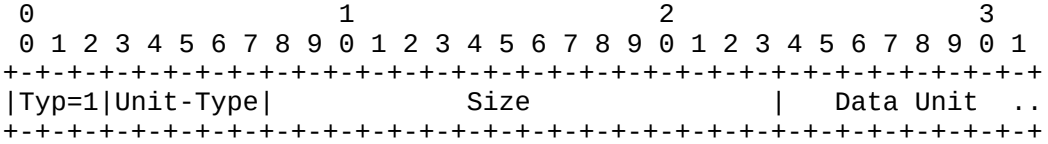


Figure 4: Aggregation Unit Payload

An example of an aggregation unit packet with two data unit payloads is shown in Figure 5.

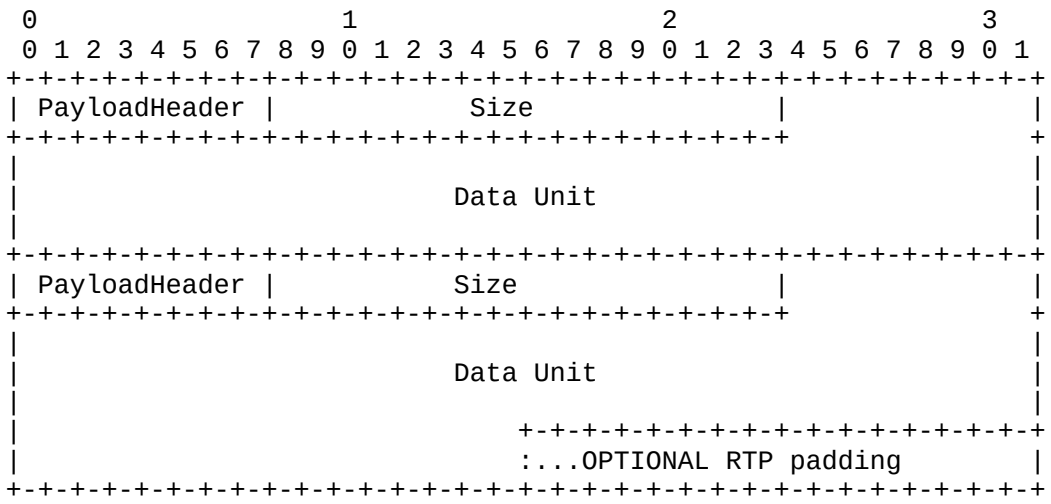


Figure 5: Packet Payload

#### 4.4.3. Fragmentation Unit

Fragmentation units allow fragmentation of single DUs into multiple RTP packets without cooperation from the G-PCC encoder. Fragments of the same DU 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).

Aggregation packets MUST NOT be fragmented. Fragmentation units MUST NOT be nested, i.e., a fragmentation unit cannot contain a subset of another fragmentation unit.

The timestamp of all fragmentation units MUST be set to the same value: the timestamp of the DU that is carried in the fragmentation unit packets.

The type field of the first packet of a series of fragmentation units MUST be set to 2 (binary 010), the type field of the last packet in the series MUST be set to four (binary 100), and the type field of all packets between the first and last packet of the fragmented unit MUST have the value three (binary 011).

## 5. Packetization Rules and Depacketization Rules

* A DU of a small size SHOULD be encapsulated in an aggregation packet together with one or more other DUs in order to avoid the unnecessary packetization overhead for small DUs. For example, parameter sets are typically small and can often be aggregated with without violating MTU size constraints.

* For carrying exactly one DU in an RTP packet, a single unit packet MUST be used.

The de-packetization process is implementation dependent. Therefore, the following de-packetization rules SHOULD be taken as an example.

* 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.

## 6. Payload Format Parameters

This section defines the media type to use with the payload format and the optional parameters that can be used with it.

### 6.1. Media Type Definition

*Note*: Template from Section 10 of [RFC4288]

*Type name*:
application

*Subtype name*:
GPCC

*Required Parameters*:
None

*Optional Parameters*:
profile-level-id *TODO*: add list from section Section 6.2

*Encoding considerations*:
This type is only defined for transfer via RTP [RFC3550].

*Security considerations*:
See Section 9.

*Interoperability considerations*:
N/A

*Published specification*:
Please refer to <*TODO*: Reference this document> and [ISO.IEC.23090-9].

*Applications that use this media type*:
Any application that relies on GPCC-based point cloud transmission over RTP

*Fragment identifier considerations*:
N/A

*Additional information*:
N/A

*Person & email address to contact for further information*:
Mathis Engelbart (mathis.engelbart@gmail.com)

*Intended usage*:
COMMON

*Restrictions on usage*:
N/A

*Author*:
See Authors' Addresses section of <*TODO*: Reference this document>

*Change controller*:
IETF avtcore@ietf.org (mailto:avtcore@ietf.org)

### 6.2. Optional Parameters Definition

*profile-level-id*: The profile-level-id is a base16 [RFC4648] (hexadecimal) representation of one byte containing the flags indicating support or conformance to the G-PCC profiles and levels. The first four bits are flags indicating support for the profiles _Simple_, _Predictive_, _Dense_ and _Main_, while the last four bits indicate the highest level that the decoder supports or a bitstream conforms to.

*pcc-resolution*: Describes the resolution of the point cloud as number of lines, number of points per line and depth resolution.

<#lines> <#points-per-line> <#depth-resolution>

*pcc-coverage*: The field of view coverage of the sensor described by the horizontal and vertical angles.

<h-angle> <v-angle>

*pcc-anchor*: An anchor reference point for the point cloud, relative to which coordinates are given. This can be useful to relate multiple point clouds to each other.

<x> <y> <z>

*pcc-orientation*: The orientation of the point cloud sensor.

<tilt> <pan>

*pcc-position*: The location of the LiDAR or point cloud sensor relative to _pcc-anchor_ and _pcc-orientation_:

<delta-x> <delta-y> <delta-z> <delta-tilt> <delta-pan>

*pcc-point-attr*: A list of attributes associated with each point. Example: color, reflectivity.

## 6.3. SDP Parameters

### 6.3.1. Mapping of Payload Type Parameters to SDP

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

*TODO*: Add all parameters from above

* The media name in the "m=" line of SDP MUST be application.

* The encoding name in the "a=rtpmap" line of SDP MUST be GPCC (the media subtype).

* The clock rate in the "a=rtpmap" line MUST be 90000.

* The optional parameters _profile-level-id_, ... 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.

### 6.3.2. Example

```
m=application 54321 RTP/AVPF 95
a=rtpmap:95 GPCC/90000
a=fmtp:95 profile-level-id=84; // 0x84 => 0b10000100 => simple profile, level 4
          --
          -- point cloud resolution
          pcc-resolution: #lines #points-per-line #depth-resolution;
          --
          -- field of view
          pcc-coverage: h-angle v-angle;
          --
          -- optional anchor point
          pcc-anchor: <type> <type-specific format> "mm" x y z | "gps" lat lon alt;
          pcc-orientation: tilt pan;
          --
          -- where is this lidar located (relative to possible others)
          pcc-position: delta-x, delta-y, delta-z delta-tilt delta-pan;
          --
          -- which attributes are encoded per point: list of properties | profile
          pcc-point-attr: "attr" color, reflectance;
```

#### 6.3.3. Offer/Answer Considerations

*TODO*: Are theses rules correct for G-PCC and do we need considerations for other parameters?

When the payload format is offered over RTP using SDP in an Offer/Answer model as described in [RFC3264] for negotiation for unicast usage, the following limitations and rules apply:

* The parameter identifying a media format configuration for G-PCC is profile-levle-id. This media format configuration parameter MUST be used symmetrically; that is, the answerer MUST either maintain this configuration parameter or remove the media format (payload type) completely if it is not supported.

* The G-PCC stream sent by either the offerer or the answerer MUST be encoded with a profile and level, lesser or equal to the values of the level and profile declared in the SDP by the receiving agent.

#### 6.3.4. Declarative SDP Considerations

*TODO*: Are theses rules correct for G-PCC and do we need considerations for other parameters?

When G-PCC 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 stream properties and receiver capabilities are used to indicate only stream properties. In this case, the parameters profile and level declare only the values used by the stream, not the capabilities for receiving streams.

* 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. Congestion Control

Congestion control for RTP SHALL be used in accordance with [RFC3550], and with any applicable RTP profile: e.g., [RFC3551]. An additional requirement if best-effort service is being used is users of this payload format MUST monitor packet loss to ensure that the packet loss rate is within acceptable parameters. Circuit Breakers [RFC8083] is an update to RTP [RFC3550] that defines criteria for when one is required to stop sending RTP Packet Streams. The circuit breakers is to be implemented and followed.

The bitrate can be dynamically adapted when real-time encoding is used. If the packet payload is not encrypted and intermediate network elements have access to the payload, they can select packets to drop based on the payload header.

Attribute decoding depends on geometry decoding and ADUs can become obsolete, when the corresponding geometry data is dropped. Thus, intermediates SHOULD prefer dropping ADUs first. Similarly, fragmentation units can only be reconstructed if all fragments arrive. Thus, it is reasonable to drop either all or none of the packets that are part of the same fragmentation unit.

## 8. IANA Considerations

*TODO*: Check if more registrations are necessary.

This memo requests that IANA registers a new media type as specified in Section 6.1. The media type is also requested to be added to the IANA registry for "RTP Payload Format MIME types" http://www.iana.org/assignments/rtp-parameters (http://www.iana.org/assignments/rtp-parameters).

## 9. Security Considerations

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 Protocol 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]. Applications SHOULD use one or more appropriate strong security mechanisms. The rest of this Security Considerations section discusses the security impacting properties of the payload format itself.

This RTP payload format and its media decoder do not exhibit any significant non-uniformity in the receiver-side computational complexity for packet processing, and thus are unlikely to pose a denial-of-service threat due to the receipt of pathological data. Nor does the RTP payload format contain any active content.

## 10. RFC Editor Considerations

Note to RFC Editor: This section may be removed after carrying out all the instructions of this section.

TODO: Consider

## 11. References

### 11.1. Normative References

[ISO.IEC.23090-9]
ISO/IEC, "Information technology — Coded representation of immersive media — Part 9: Geometry-based point cloud compression", ISO/IEC 23090-9, 2023, <https://www.iso.org/standard/78990.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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/rfc3711>.

[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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[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/rfc/rfc5124>.

[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/rfc/rfc7826>.

[RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <https://www.rfc-editor.org/rfc/rfc8083>.

[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>.

[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>.

### 11.2. Informative References

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974, October 2000, <https://www.rfc-editor.org/rfc/rfc2974>.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", RFC 4288, DOI 10.17487/RFC4288, December 2005, <https://www.rfc-editor.org/rfc/rfc4288>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <https://www.rfc-editor.org/rfc/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/rfc/rfc7202>.

## Acknowledgments

TODO acknowledge.

## Authors' Addresses

Mathis Engelbart
Technical University Munich
Email: mathis.engelbart@tum.de

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

Lukasz Kondrad
Nokia Technologies
Email: lukasz.kondrad@nokia.com