Network Working Group
Request for Comments: 4396
Category: Standards Track

J. Rey
Y. Matsui
Panasonic

February 2006

RTP Payload Format for 3rd Generation Partnership Project (3GPP) Timed Text

Status of This Memo

Copyright Notice

Abstract

This document specifies an RTP payload format for the transmission of 3GPP (3rd Generation Partnership Project) timed text. 3GPP timed text is a time-lined, decorated text media format with defined storage in a 3GP file. Timed Text can be synchronized with audio/video contents and used in applications such as captioning, titling, and multimedia presentations. In the following sections, the problems of streaming timed text are addressed, and a payload format for streaming 3GPP timed text over RTP is specified.

Table of Contents

1. Introduction

3GPP timed text is a media format for time-lined, decorated text specified in the 3GPP Technical Specification TS 26.245, "Transparent end-to-end packet switched streaming service (PSS); Timed Text Format (Release 6)" [1]. Besides plain text, the 3GPP timed text format allows the creation of decorated text such as that for karaoke applications, scrolling text for newscasts, or hyperlinked text. These contents may or may not be synchronized with other media, such as audio or video.

The purpose of this document is to provide a means to stream 3GPP timed text contents using RTP [3]. This includes the streaming of timed text being read out of a (3GP) file, as well as the streaming of timed text generated in real-time, a.k.a. live streaming.

Section 2 contains the motivation for this document, an overview of the media format, the requirements, and the design rationale. Section 3 defines the terminology used. Section 4 specifies the payload headers, the fragmentation and re-assembly rules for text samples, the rules for payload aggregation, and the relations of this document to RFC 3640 [12] and RFC 2793 [22]. Section 5 specifies some simple schemes for resilient transport and gives pointers to other possible mechanisms. Section 6 addresses congestion control. Section 7 specifies scene description. Section 8 defines the media type. Section 9 specifies SDP for unicast and multicast sessions, including usage in the Offer/Answer model [13]. Sections 10 and 11 address IANA and security considerations. Section 12 lists references. Basics of the 3GP File Structure are in Section 13.

2. Motivation, Requirements, and Design Rationale

2.1. Motivation

The 3GPP timed text format was developed for use in the services specified in the 3GPP Transparent End-to-end Packet-switched Streaming Services (3GPP PSS) specification [16].

As of today, PSS allows downloading 3GPP timed text contents stored in 3GP files. However, due to the lack of a RTP payload format, it is not possible to stream 3GPP timed text contents over RTP.

This document specifies such a payload format.

2.2. Basic Components of the 3GPP Timed Text Media Format

Before going into the details of the design, it is necessary to know how the media format is constructed. We can identify four differentiated functional components: layout information, default formatting, text strings, and decoration. In the following, we shortly explain these and match them to their designations in a 3GP file:

- Initial spatial layout information related to the text strings: These are the height and width of the text region where text is displayed, the position of the text region in the display, and the layer or proximity of the text to the user. In 3GP files, this information is contained in the Track Header Box (3GP file designations are capitalized for clarity).

- Default settings for formatting and positioning of text: style (font, size, color,...), background color, horizontal and vertical justification, line width, scrolling, etc. For 3GP files, this corresponds to the Sample Descriptions.

- The actual text strings: encoded characters using either UTF-8 [18] or UTF-16 [19] encoding.

- The decoration: If some characters have different style, delay, blink, etc., this needs to be indicated. The decoration is only present in the text samples if it is actually needed. Otherwise, the default settings as above apply. In 3GP files, within each Text Sample, the decoration (i.e., Modifier Boxes) is appended to the text strings, if needed. At the time of writing this payload format, the following modifiers are specified in the 3GPP timed text media format specification [1]:

  - text highlight
  - highlight color
  - blinking text
  - karaoke feature
  - hyperlink
  - text delay
  - text style
  - positioning of the text box
  - text wrap indication

2.3. Requirements

Once the basic components are known, it is necessary to define which requirements the payload format shall fulfill:

1. It shall enable both live streaming and streaming from a 3GP file.

   Informative note: For the purpose of this document, the term "live streaming" refers to those scenarios where the timed text stream is sent from a live encoder. Upon reception, the content may or may not be stored in a 3GP file. Typically, in live streaming applications, the sender encapsulates the timed text content in RTP packets following the guidelines given in this document. At the receiving side, a buffer is used to cancel the network delay and delay jitter. If receiver and sender support packet loss resilience mechanisms (see Section 5), it may also be possible to recover from packet losses. Note that how sender and receiver actually manage and dimension the buffers is an implementation design choice.

2. Furthermore, it shall be possible for an RTP receiver using this payload format, and capable of storing in 3GP format, to obtain all necessary information from the RTP packets for storing the received text contents according to the 3GP file format. This file may or may not be the same as the original file.

   Informative note: The 3GP file format itself is based on the ISO Base Media File Format recommendation [2]. Section 13.1 gives some insight into the 3GP file structure. Further, Sections 4.3 and 7.3 specify where the information needed for filling in payload headers is found in a 3GP file. For live streaming, appropriate values complying with the format and units described in [1] shall be used. Where needed, clarifications on appropriate values are given in this document.

3. It shall enable efficient and resilient transport of timed text contents over RTP. In particular:

   a. Enable the transmission of the sample descriptions by both out-of-band and in-band means. Sample descriptions are important information, which potentially apply to several text samples. These default formatting settings are typically transmitted out-of-band (reliably) once at the initialization phase. If additional sample descriptions

are needed in the course of a session, these may also be sent out-of-band or in-band. In-band transmission, although unreliable, may be more appropriate for sending sample descriptions if these should be sent frequently, as opposed to establishing an additional communication channel for SDP, for example. It is also useful in cases where an out-of-band channel may not be available and for live streaming, where contents are not known a priori. Thus, the payload format shall enable out-of-band and in-band transmission of sample descriptions. Section 4.1.6 specifies a payload header for transmitting sample descriptions in-band. Section 9 specifies how sample descriptions are mapped to SDP.

b. Enable the fragmentation of a text sample into several RTP packets in order to cover a wide range of applications and network environments. In general, fragmentation should be a rare event, given the low bit rates and relatively small text sample sizes. However, the 3GPP Timed Text media format does allow for larger text samples. Therefore, the payload format shall take this into account and provide a means for coping with fragmentation and reassembly. Section 4.4 deals with fragmentation.

c. Enable the aggregation of units into an RTP packet for making the transport more efficient. In a mobile communication environment, a typical text sample size is around 100-200 bytes. If the available bit rate and the packet size allow it, units should be aggregated into one RTP packet. Section 4.6 deals with aggregation.

d. Enable the use of resilient transport mechanisms, such as repetition, retransmission [11], and FEC [7] (see Section 5). For a more general discussion, refer to RFC 2354 [8], which discusses available mechanisms for stream repair.

2.4. Limitations

The payload headers have been optimized in size for RTP. Instead of using 32-bit (S)LEN, SDUR, and SIDX header fields, which would carry many unused bits much of the time, it has been a design choice to reduce the size of these fields. As a consequence, this payload format has reduced maximum values with respect to sizes and durations of (text) samples and sample descriptions. These maximum values differ from those allowed in 3GP files, where they are expressed using 32-bit (unsigned) integers. In some cases,

extension mechanisms are provided to deal with larger values. However, it is noted that the values used here should be enough for the streaming applications targeted.

The following limitations apply:

1. The maximum size of text samples carried in RTP packets is restricted to be a 16-bit (unsigned) integer (this includes the text strings and modifiers). This means a maximum size for the unit would be about 64 Kbytes. No extension mechanism is provided.

2. The sample description index values are restricted to be an 8-bit (unsigned) integer. An extension mechanism is given in Section 4.3.

3. The text sample duration is restricted to be a 24-bit (unsigned) integer. This yields a maximum duration at a timestamp clockrate of 1000 Hz of about 4.6 hours. Nevertheless, an extension mechanism is provided in Section 4.3.

4. Sample descriptions are also restricted in size: If the size cannot be expressed as a 16-bit (unsigned) integer, the sample description shall not be conveyed. As in the case of the sample size, no extension mechanism is provided.

5. A further limitation concerns the UTF-16 encodings supported: Only transport of text strings following big endian byte order is supported. See Section 4.1.1 for details.

2.5. Design Rationale

The following design choices were made:

1. 'Unit' approach: The payload formats specified in this document follow a simple scheme: a 3-byte common header (Common Payload Header) followed by a specific header for each text sample (fragment) type. Following these headers, the text sample contents are placed (Section 4.1.1 and following). This structure is called a 'unit'.

   The following units have been devised to comply with the requirements mentioned in Section 2.3:

   a. A TYPE 1 unit that contains one complete text sample,

   b. A TYPE 2 unit that contains a complete text string or a fragment thereof,

c. A TYPE 3 unit that contains the complete modifiers or only the first fragment thereof,

d. A TYPE 4 unit that contains one modifier fragment other than the first, and

e. A TYPE 5 unit that contains one sample description.

This 'unit' approach was motivated by the following reasons:

1. Allows a simple classification of the text samples and text sample fragments that can be conveyed by the payload format.

2. Enables easy interoperability with RFC 3640 [12]. During the development of this payload format, interest was shown from MPEG-4 standardization participants in developing a common payload structure for the transport of 3GPP Timed Text. While interoperability is not strictly necessary for this payload format to work, it has been pursued in this payload format. Section 4.8 explains how this is done.

2. Character count is not implemented. This payload format does detect lost text samples fragments, but it does not enable an RTP receiver to find out the exact number of text characters lost. In fact, the fragment size included in the payload headers does not help in finding the number of lost characters because the UTF-8/UTF-16 [18][19] encodings used yield a variable number of bytes per character.

   For finding the exact number of lost characters, an additional field reflecting the character count (and possibly the character offset) upon fragmentation would be required. This would additionally require that the entity performing fragmentation count the characters included in each text fragment.

   One benefit of having a character count would be that the display application would be able to replace missing characters through some other character representing character loss. For example:

   If we take the "Some text is lost now" and assume the loss of a packet containing the text in the middle, this could be displayed (with a character count):

   "Some ############now"

As opposed to:

"Some #now"

which is what this payload format enables ("#" indicates a missing character or packet, respectively).

However, it is the consensus of the working group that for applications such as subtitling applications and multimedia presentations that use this payload format, such partial error correction is not worth the cost of including two additional fields; namely, character count and character offset. Instead, it is recommended that some more overhead be invested to provide full error correction by protecting the less text sample fragments using the measures outlined in Section 5.

3. Fragment re-assembly: In order to re-assemble the text samples, offset information is needed. Instead of a character or byte offset, a single byte, TOTAL/THIS, is used. These two values indicate the total number and current index of fragments of a text sample. This is simpler than having a character offset field in each fragment. Details in Section 4.1.3.

4. A length field, LEN, is present in the common header fields. While the length in the RTP payload format is not needed by most RTP applications (typically lower layers, like UDP, provide this information), it does ease interoperability with RFC 3640. This is because the Access Units (AUs) used for carriage of data in RFC 3640 must include a length indication. Details are in Section 4.8.

5. The header fields in the specific payload headers (TYPE headers in Sections 4.1.2 to 4.1.6) have been arranged for easy processing on 32-bit machines. For this reason, the fields SIDX and SDUR are swapped in TYPE 1 unit, compared to the other units.

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [5].

Furthermore, the following terms are used and have specific meaning within the context of this document:

text sample or whole text sample

In the 3GPP Timed Text media format [1], these terms refer to a unit of timed text data as contained in the source (3GP) file. This includes the text string byte count, possibly a Byte Order Mark, the text string and any modifiers that may follow. Its equivalent in audio/video would be a frame.

In this document, however, a text sample contains only text strings followed by zero or more modifiers. This definition of text sample excludes the 16-bit text string byte count and the 16-bit Byte Order Mark (BOM) present in 3GP file text samples (see Section 4.3 and Figure 9). The 16-bit BOM is not transported in RTP, as explained in Section 4.1.1.

text strings

The actual text characters encoded either as UTF-8 or UTF-16. When using this payload format, the text string does not contain any byte order mark (BOM). See Figure 9 for details.

fragment or text sample fragment

A fraction of a text sample. A fragment may contain either text strings or modifier (decoration) contents, but not both at the same time.

sample contents

General term to identify timed text data transported when using this payload format. Sample contents may be one or several text samples, sample descriptions, and sample fragments (note that, as per Section 4.6, there is only one case in which more than one fragment may be included in a payload).

decoration or modifiers

These terms are used interchangeably throughout the document to denote the contents of the text sample that modify the default text formatting. Modifiers may, for example, specify different font size for a particular sequence of characters or define karaoke timing for the sample.

sample description

Information that is potentially shared by more than one text sample. In a 3GP file, a sample description is stored in a place where it can be shared. It contains setup and default information such as scrolling direction, text box position, delay value, default font, background color, etc.

units or transport units

The payload headers specified in this document encapsulate text samples, fragments thereof, and sample descriptions by placing a common header and specific payload header (Sections 4.1.1 to 4.1.6) before them, thus building what is here called a (transport) unit.

aggregation or aggregate packet

The payload of an aggregate (RTP) packet consists of several (transport) units.

track or stream

3GP files contain audio/video and text tracks. This document enables streaming of text tracks using RTP. Therefore, these terms are used interchangeably in this document in the context of 3GP files.

Media Header Box / Track Header Box / ...

The 3GP file format makes use of these structures defined in the ISO Base File Format [2]. When referring to these in this document, initials are capitalized for clarity.

4. RTP Payload Format for 3GPP Timed Text

The format of an RTP packet containing 3GPP timed text is shown below:

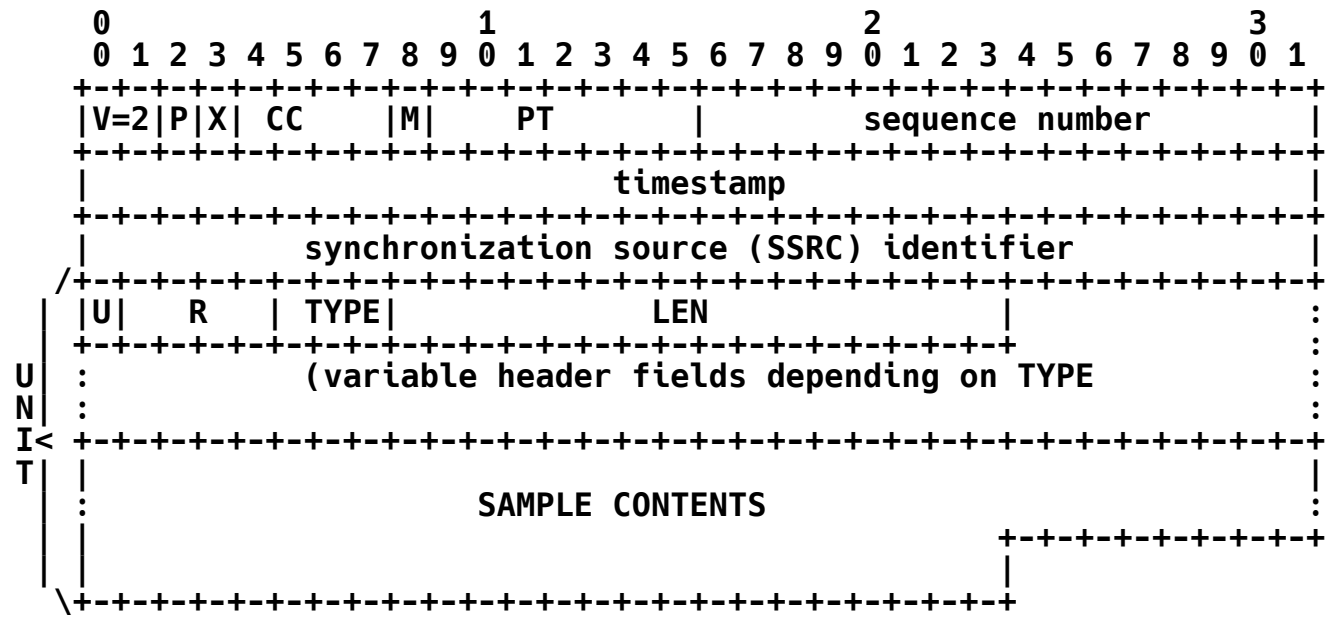


Figure 1. 3GPP Timed Text RTP Packet Format

Marker bit (M): The marker bit SHALL be set to 1 if the RTP packet includes one or more whole text samples or the last fragment of a text sample; otherwise, it is set to zero (0).

Timestamp: The timestamp MUST indicate the sampling instant of the earliest (or only) unit contained in the RTP packet. The initial value SHOULD be randomly determined, as specified in RTP [3].

> The timestamp value should provide enough timing resolution for expressing the duration of text samples, for synchronizing text with other media, and for performing RTP Control Protocol (RTCP) measurements such as the interarrival delay jitter or the RTCP Packet Receipt Times Report Block (Section 4.3 of RFC 3611 [20]). This is compliant to RTP, Section 5.1:
>
> > "The resolution of the clock MUST be sufficient for the desired synchronization accuracy and for measuring packet arrival jitter (one tick per video frame is typically not sufficient)".

The above observation applies to both timed text tracks included in a 3GP file and live streaming sessions. In the case of a 3GP timed text track, the timestamp clockrate is the value of the "timescale" parameter in the Media Header Box for that text track. Each track in a 3GP file MAY have its own clockrate as specified in the Media Header Box. Likewise, live streaming applications SHALL use an appropriate timestamp clockrate. A default value of 1000 Hz is RECOMMENDED. Other timestamp clockrates MAY be used. In this case, the typical behavior here is to match the 3GPP timed text clockrate to that used by an associated audio or video stream.

In an aggregate payload, units MUST be placed in play-out order, i.e., earliest first in the payload. If TYPE 1 units are aggregated, the timestamp of the subsequent units MUST be obtained by adding the timed text sample duration of previous samples to the RTP timestamp value. There are two exceptions to this rule: TYPE 5 units and an aggregate payload containing two fragments of the same text sample. The details of the timestamp calculation are given in Section 4.6.

Finally, timestamp clockrates MUST be signaled by out-of-band means at session setup, e.g., using the media type "rate" parameter in SDP. See Section 9 for details.

Payload Type (PT): The payload type is set dynamically and sent by out-of-band means.

The usage of the remaining RTP header fields (namely, V, P, X, CC, SN and SSRC) follows the rules of RTP and the profile in use.

4.1. Payload Header Definitions

The (transport) units specified in this document consist of a set of common fields (U, R, TYPE, LEN), followed by specific header fields (TYPES 1-5) and text sample contents. See Figure 1 and Figure 2.

In Figure 2, two example RTP packets are depicted. The first contains an aggregate RTP payload with two complete text samples, and the second contains one text sample fragment. After each unit header is explained, detailed payload examples follow in Section 4.7.

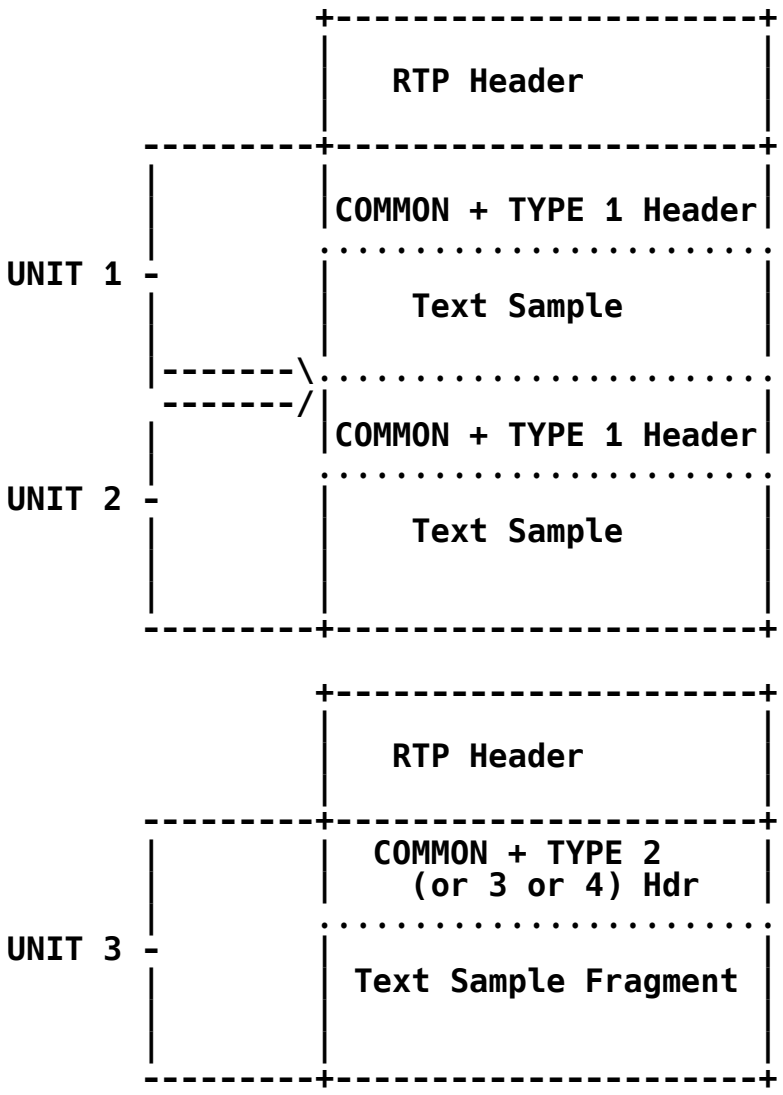


Figure 2. Example RTP packets

4.1.1. Common Payload Header Fields

The fields common to all payload headers have the following format:

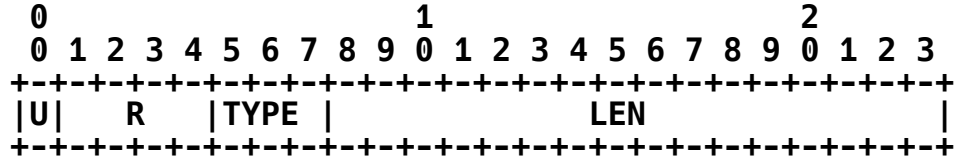


Figure 3. Common payload header fields

Where:

- U (1 bit) "UTF Transformation flag": This is used to inform RTP receivers whether UTF-8 (U=0) or UTF-16 (U=1) was used to encode the text string. UTF-16 text strings transported by this payload format MUST be serialized in big endian order, a.k.a. network byte order.

  Informative note: Timed text clients complying with the 3GPP Timed Text format [1] are only required to understand the big endian serialization. Thus, in order to ease interoperability, the reverse serialization (little endian) is not supported by this payload format.

  For the payload formats defined in this document, the U bit is only used in TYPE 1 and TYPE 2 headers. Senders MUST set the U bit to zero in TYPE 3, TYPE 4, and TYPE 5 headers. Consequently, receivers MUST ignore the U bit in TYPE 3, TYPE 4, and TYPE 5 headers.

- R (4 bits) "Reserved bits": for future extensions. This field MUST be set to zero (0x0) and MUST be ignored by receivers.

- TYPE (3 bits) "Type Field": This field specifies which specific header fields follow. The following TYPE values are defined:

  - TYPE 1, for a whole text sample.
  - TYPE 2, for a text string fragment (without modifiers).
  - TYPE 3, for a whole modifier box or the first fragment of a modifier box.
  - TYPE 4, for a modifier fragment other than first.
  - TYPE 5, for a sample description. Exactly one header per sample description.
  - TYPE 0, 6, and 7 are reserved for future extensions. Note that future extensions are possible, e.g., a unit that explicitly signals the number of characters present in a

fragment (see Section 2.5). In order to guarantee backwards-compatibility, it SHALL be possible that older clients ignore (newer) units they do not understand, without invalidating the timestamp calculation mechanisms or otherwise preventing them from decoding the other units.

- Finally, the LEN (16 bits) "Length Field": indicates the size (in bytes) of this header field and all the fields following, i.e., the LEN field followed by the unit payload: text strings and modifiers (if any). This definition only excludes the initial U/R/TYPE byte of the common header. The LEN field follows network byte order.

  The way in which LEN is obtained when streaming out of a 3GP file depends on the particular unit type. This is explained for each unit in the sections below.

  For live streaming, both sample length and the LEN value for the current fragment MUST be calculated during the sampling process or during fragmentation.

  In general, LEN may take the following values:

  - TYPE = 1, LEN >= 8
  - TYPE = 2, LEN > 9
  - TYPE = 3, LEN > 6
  - TYPE = 4, LEN > 6
  - TYPE = 5, LEN > 3

  Receivers MUST discard units that do not comply with these values. However, the RTP header fields and the rest of the units in the payload (if any) are still useful, as guaranteed by the requirement for future extensions above.

  In the following subsections the different payload headers for the values of TYPE are specified.

4.1.2. TYPE 1 Header

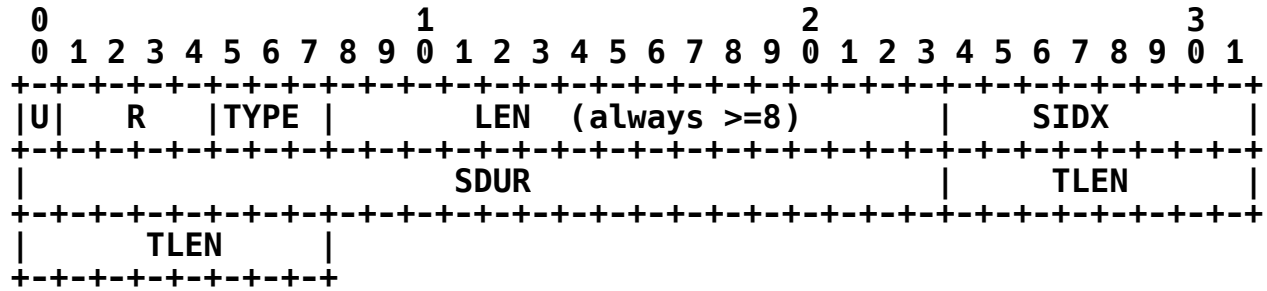


Figure 4. TYPE 1 Header Format

This header type is used to transport whole text samples. This unit should be the most common case, i.e., the text sample should usually be small enough to be transported in one unit without having to separate text strings from modifiers. In an aggregate (RTP packet) payload containing several text samples, every sample is preceded by its own TYPE 1 header (see Figure 12).

> Informative note: As indicated in Section 3, "Terminology", a text sample is composed of the text strings followed by the modifiers (if any). This is also how text samples are stored in 3GP files. The separation of a text sample into text strings and modifiers is only needed for large samples (or small available IP MTU sizes; see Section 4.4), and it is accomplished with TYPE 2 and TYPE 3 headers, as explained in the sections below.

Note also that empty text samples are considered whole text samples, although they do not contain sample contents. Empty text samples may be used to clear the display or to put an end to samples of unknown duration, for example. Units without sample contents SHALL have a LEN field value of 8 (0x0008).

The fields above have the following meaning:

- U, R, and TYPE, as defined in Section 4.1.1.
- LEN, in this case, represents the length of the (complete) text sample plus eight (8) bytes of headers. For finding the length of the text sample in the Sample Size Box of 3GP files, see Section 4.3.
- SIDX (8 bits) "Text Sample Entry Index": This is an index used to identify the sample descriptions.

The SIDX field is used to find the sample description corresponding to the unit's payload. There are two types of SIDX values: static and dynamic.

Static SIDX values are used to identify sample descriptions that MUST be sent out-of-band and MUST remain active during the whole session. A static SIDX value is unequivocally linked to one particular sample description during the whole session. Carrying many sample descriptions out-of-band SHOULD be avoided, since these may become large and, ultimately, transport is not the goal of the out-of-band channel. Thus, this feature is RECOMMENDED for transporting those sample descriptions that provide a set of minimum default format settings. Static SIDX values MUST fall in the (closed) interval [129,254].

Dynamic SIDX values are used for sample descriptions sent in-band. Sample descriptions MAY be sent in-band for several reasons: because they are generated in real time, for transport resiliency, or both. A dynamic SIDX value is unequivocally linked to one particular sample description during the period in which this is active in the session, and it SHALL NOT be modified during that period. This period MAY be smaller than or equal to the session duration. This period is not known a priori. A maximum of 64 dynamic simultaneously active SIDX values is allowed at any moment. Dynamic SIDX values MUST fall in the closed interval [0,127]. This should be enough for both recorded content and live streaming applications. Nevertheless, a wraparound mechanism is provided in Section 4.2.1 to handle streaming sessions where more than 64 SIDX values might be needed. Servers MAY make use of dynamic sample descriptions. Clients MUST be able to receive and interpret dynamic sample descriptions.

Finally, SIDX values 128 and 255 are reserved for future use.

- SDUR (24 bits) "Text Sample Duration": indicates the sample duration in RTP timestamp units of the text sample. For this field, a length of 3 bytes is preferred to 2 bytes. This is because, for a typical clockrate of 1000 Hz, 16 bits would allow for a maximum duration of just 65 seconds, which might be too short for some streams. On the other hand, 24 bits at 1000 Hz allow for a maximum duration of about 4.6 hours, while for 90 KHz, this value is about 3 minutes. These values should be enough for streaming applications. However, if a larger duration is needed, the extension mechanism specified in Section 4.3 SHALL be used.

  Apart from defining the time period during which the text is displayed, the duration field is also used to find the timestamp of subsequent units within the aggregate RTP packet payload (if any).

This is explained in Section 4.6.

Text samples have generally a known duration at the time of transmission. However, in some cases such as live streaming, the time for which a text piece shall be presented might not be known a priori. Thus, the value zero SDUR=0 (0x000000) is reserved to signal unknown duration. The amount of time that a sample of unknown duration is presented is determined by the timestamp of the next sample that shall be displayed at the receiver: Text samples of unknown duration SHALL be displayed until the next text sample becomes active, as indicated by its timestamp.

The next example illustrates how units of unknown duration MUST be presented. If no text sample following is available, it is an implementation issue what should be displayed. For example, a server could send an empty sample to clear the text box.

> Example: Imagine you are in an airport watching the latest news report while you wait for your plane. Airports are loud, so the news report is transcribed in the lower area of the screen. This area displays two lines of text: the headlines and the words spoken by the news speaker. As usual, the headlines are shown for a longer time than the rest. This time is, in principle, unknown to the stream server, which is streaming live. A headline is just replaced when the next headline is received.

However, upon storing a text sample with SDUR=0 in a 3GP file, the SDUR value MUST be changed to the effective duration of the text sample, which MUST be always greater than zero (note that the ISO file format [2] explicitly forbids a sample duration of zero). The effective duration MUST be calculated as the timestamp difference between the current sample (with unknown duration) and the next text sample that is displayed.

Note that samples of unknown duration SHALL NOT use features, which require knowledge of the duration of the sample up front. Such features are scrolling and karaoke in [1]. This also applies for future extensions of the Timed Text format. Furthermore, only sample descriptions (TYPE 5 units) MAY follow units of unknown duration in the same aggregate payload. Otherwise, it would not be possible to calculate the timestamp of these other units.

For text contents stored in 3GP files, see Section 4.3 for details on how to extract the duration value. For live streaming, live encoders SHALL assign appropriate values and units according to [1] and later releases.

- TLEN (16 bits), "Text String Length", is a byte count of the text string. The decoder needs the text string length in order to know where the modifiers in the payload start. TLEN is not present in text string fragments (TYPE 2) since it can be deductively calculated from the LEN values of each fragment.

  The TLEN value is obtained from the text samples as contained in 3GP files. Refer to Section 4.3. For live content, the TLEN MUST be obtained during the sampling process.

- Finally, the actual text sample is placed after the TLEN field. As defined in Section 3, a text sample consists of a string of characters encoded using either UTF-8 or UTF-16, followed by zero or more modifiers. Note also that no BOM and no byte count are included in the strings carried in the payload (as opposed to text samples stored in 3GP files [1]).

4.1.3. TYPE 2 Header

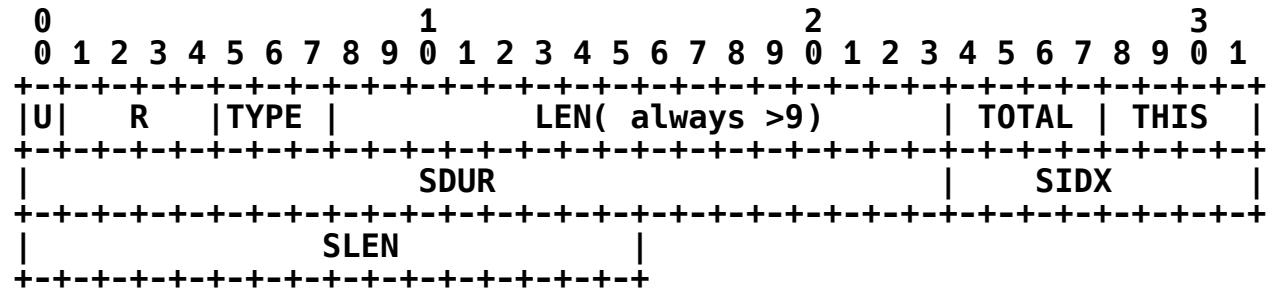


Figure 5. TYPE 2 Header Format

This header type is used to transport either a whole text string or a fragment of it. TYPE 2 units SHALL NOT contain modifiers. In detail:

- U, R, and TYPE, as defined in Section 4.1.1.

- SIDX and SDUR, as defined in Section 4.1.2.

  Note that the U, SIDX, and SDUR fields are meaningful since partial text strings can also be displayed.

- The LEN field (16 bits) indicates the length of the text string fragment plus nine (9) bytes of headers. Its value is calculated upon fragmentation. LEN MUST always be greater than nine (0x0009). Otherwise, the unit MUST be discarded.

According to the guidelines in Section 4.4, text strings MUST be split at character boundaries for allowing the display of text fragments. Therefore, a text fragment MUST contain at least one character in either UTF-8 or UTF-16. Actually, this is just a formalism since by observing the guidelines, much larger fragments should be created.

Note also that TYPE 2 units do not contain an explicit text string length, TLEN (see TYPE 1). This is because TYPE 2 units do not contain any modifiers after the text string. If needed, the length of the received string can be obtained using the LEN values of the TYPE 2 units.

- The SLEN field (16 bits) indicates the size (in bytes) of the original (whole) text sample to which this fragment belongs. This length comprises the text string plus any modifier boxes present (and includes neither the byte order mark nor the text string length as mentioned in Section 3, "Terminology").

  Regarding the text sample length: Timed text samples are not generated at regular intervals, nor is there a default sample size. If 3GP files are streamed, the length of the text samples is calculated beforehand and included in the track itself, while for live encoding it is the real time encoder that SHALL choose an appropriate size for each text sample. In this case, the amount of text 'captured' in a sample depends on the text source and the particular application (see examples below). Samples may, e.g., be tailored to match the packet MTU as closely as possible or to provide a given redundancy for the available bit rate. The encoding application MUST also take into account the delay constraints of the real-time session and assess whether FEC, retransmission, or other similar techniques are reasonable options for stream repair.

  The following examples shall illustrate how a real-time encoder may choose its settings to adapt to the scenario constraints.

  Example: Imagine a newscast scenario, where the spoken news is transcribed and synchronized with the image and voice of the reporter. We assume that the news speaker talks at an average speed of 5 words per second with an average word length of 5 characters plus one space per word, i.e., 30 characters per second. We assume an available IP MTU of 576 bytes and an available bitrate of 576*8 bits per second = 4.6 Kbps. We assume each character can be encoded using 2 bytes in UTF-16. In this scenario, several constraints may apply; for example: available IP MTU, available bandwidth, allowable delay, and required redundancy. If the target were to minimize the

packet overhead, a text sample covering 8 seconds of text would be closest to the IP MTU:

```
  IP/UDP/RTP/TYPE1 Header + (8-second text sample)
= 20 + 8 + 12 + 8 + (~6 chars/word * 5 word/s * 8 s * 2 chars/word)
= 528 bytes < 576 bytes
```

For other scenarios, like lossy networks, it may happen that just one packet per sample is too low a redundancy. In this case, a choice could be that the encoder 'collects' text every second, thus yielding text samples (TYPE 1 units) of 68 bytes, TYPE 1 header included. We can, e.g., include three contiguous text samples in one RTP payload: the current and last two text samples (see below). This accounts to a total IP packet size of 20 + 8 + 12 + 3*(8 + 60) = 244 bytes. Now, with the same available bitrate of 4.6 Kbps, these 244-byte packets can be sent redundantly up two times per second:

```
RTP payload (1,2,3)(1,2,3) (2,3,4)(2,3,4) (3,4,5)(3,4,5) ...
Time:       <----1s------> <----1s------> <-----1s-----> ...
```

This means that each text sample is sent at least six times, which should provide enough redundancy. Although not as bandwidth efficient (488*8 < 528*8 < 576*8 bps) as the previous packetization, this option increases the stream redundancy while still meeting the delay and bandwidth constraints.

Another example would be a user sending timed text from a type-in area in the display. In this case, the text sample is created as soon as the user clicks the 'send' button. Depending on the packet length, fragmentation may be needed.

In a video conferencing application, text is synchronized with audio and video. Thus, the text samples shall be displayed long enough to be read by a human, shall fit in the video screen, and shall 'capture' the audio contents rendered during the time the corresponding video and audio is rendered.

For stored content, see Section 4.3 for details on how to find the SLEN value in a 3GP file. For live content, the SLEN MUST be obtained during the sampling process.

Finally, note that clients MAY use SLEN to buffer space for the remaining fragments of a text sample.

- The fields TOTAL (4 bits) and THIS (4 bits) indicate the total number of fragments in which the original text sample (i.e., the

text string and its modifiers) has been fragmented and which order occupies the current fragment in that sequence, respectively. Note that the sequence number alone cannot replace the functionality of the THIS field, since packets (and fragments) may be repeated, e.g., as in repeated transmission (see Section 5). Thus, an indication for "fragment offset" is needed.

The usual "byte offset" field is not used here for two reasons: a) it would take one more byte and b) it does not provide any information on the character offset. UTF-8/UTF-16 text strings have, in general, a variable character length ranging from 1 to 6 bytes. Therefore, the TOTAL/THIS solution is preferred. It could also be argued that the LEN and SLEN fields be used for this purpose, but while they would provide information about the completeness of the text sample, they do not specify the order of the fragments.

In all cases (TYPEs 2, 3 and 4), if the value of THIS is greater than TOTAL or if TOTAL equals zero (0x0), the fragment SHALL be discarded.

- Finally, the sample contents following the SLEN field consist of a fragment of the UTF-8/UTF-16 character string; no modifiers follow.

4.1.4. TYPE 3 Header

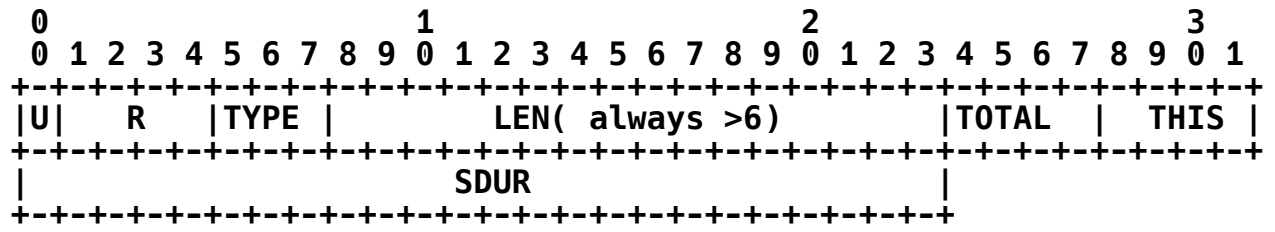


Figure 6. TYPE 3 Header Format

This header type is used to transport either the entire modifier contents present in a text sample or just the first fragment of them. This depends on whether the modifier boxes fit in the current RTP payload.

If a text sample containing modifiers is fragmented, this header MUST be used to transport the first fragment or, if possible, the complete modifiers.

In detail:

- The U, R, and TYPE fields are defined as in Section 4.1.1.

- LEN indicates the length of the modifier contents. Its value is obtained upon fragmentation. Additionally, the LEN field MUST be greater than six (0x0006). Otherwise, the unit MUST be discarded.

- The TOTAL/THIS field has the same meaning as for TYPE 2.

  For TYPE 3 units containing the last (trailing) modifier fragment, the value of TOTAL MUST be equal to that of THIS (TOTAL=THIS). In addition, TOTAL=THIS MUST be greater than one, because the total number of fragments of a text sample is logically always larger than one.

  Otherwise, if TOTAL is different from THIS in a TYPE 3 unit, this means that the unit contains the first fragment of the modifiers.

- The SDUR has the same definition for TYPE 1. Since the fragments are always transported in own RTP packets, this field is only needed to know how long this fragment is valid. This may, e.g., be used to determine how long it should be kept in the display buffer.

Note that the SLEN and SIDX fields are not present in TYPE 3 unit headers. This is because a) these fragments do not contain text strings and b) these types of fragments are applied over text string fragments, which already contain this information.

4.1.5. TYPE 4 Header

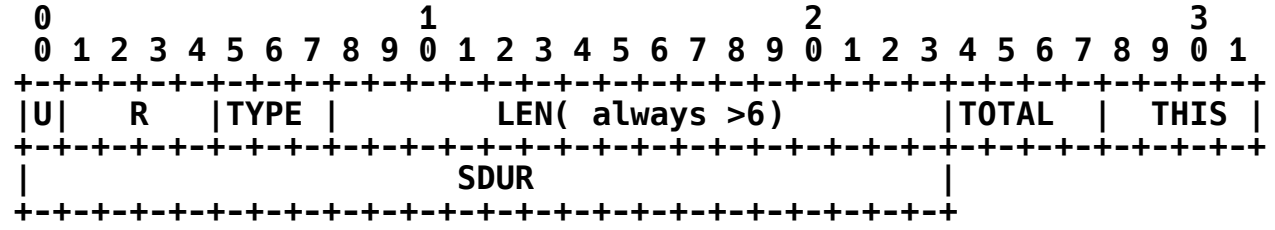


Figure 7. TYPE 4 Header Format

This header type is placed before modifier fragments, other than the first one.

The U, R, and TYPE fields are used as per Section 4.1.1.

LEN indicates as for TYPE 3 the length of the modifier contents and SHALL also be obtained upon fragmentation. The LEN field MUST be greater than six (0x0006). Otherwise, the unit MUST be discarded.

TOTAL/THIS is used as in TYPE 2.

The SDUR field is defined as in TYPE 1. The reasoning behind the absence of SLEN and SIDX is the same as in TYPE 3 units.

4.1.6. TYPE 5 Header

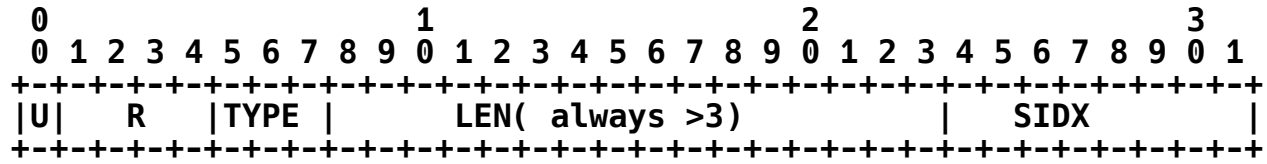


Figure 8. TYPE 5 Header Format

This header type is used to transport (dynamic) sample descriptions. Every sample description MUST have its own TYPE 5 header.

The U, R, and TYPE fields are used as per Section 4.1.1.

The LEN field indicates the length of the sample description, plus three units accounting for the SIDX and LEN field itself. Thus, this field MUST be greater than three (0x0003). Otherwise, the unit MUST be discarded.

If the sample is streamed from a 3GP file, the length of the sample description contents (i.e., what comes after SIDX in the unit itself) is obtained from the file (see Section 4.3).

The SIDX field contains a dynamic SIDX value assigned to the sample description carried as sample content of this unit. As only dynamic sample descriptions are carried using TYPE 5, the possible SIDX values are in the (closed) interval [0,127].

Senders MAY make use of TYPE 5 units. All receivers MUST implement support for TYPE 5 units, since it adds minimum complexity and may increase the robustness of the streaming session.

The next section specifies how SIDX values are calculated.

4.2. Buffering of Sample Descriptions

The buffering of sample descriptions is a matter of the client's timed text codec implementation. In order to work properly, this payload format requires that:

- Static sample descriptions MUST be buffered at the client, at least, for the duration of the session.

- If dynamic sample descriptions are used, their buffering and update of the SIDX values MUST follow the mechanism described in the next section.

4.2.1. Dynamic SIDX Wraparound Mechanism

The use of dynamic sample descriptions by senders is OPTIONAL. However, if they are used, senders MUST implement this mechanism. Receivers MUST always implement it.

Dynamic SIDX values remain active either during the entire duration of the session (if used just once) or in different intervals of it (if used once or more).

> Note: In the following, SIDX means dynamic SIDX.

For choosing the wraparound mechanism, the following rationale was used: There are 128 dynamic SIDX values possible, [0..127]. If one chooses to allow a maximum of 127 to be used as dynamic SIDXs, then any reordered packet with a new sample description would make the mechanism fail. For example, if the last packet received is SIDX=5, then all 127 values except SIDX=6 would be "active". Now, if a reordered packet arrives with a new description, SIDX=9, it will be mistakenly discarded, because the SIDX=9 is, at that moment, marked as "active" and active sample descriptions shall not be re-written. Therefore, a "guard interval" is introduced. This guard interval reduces the number of active SIDXs at any point in time to 64. Although most timed text applications will probably need less than 64 sample descriptions during a session (in total), a wraparound mechanism to handle the need for more is described here.

Thereby, a sliding window of 64 active SIDX values is used. Values within the window are "active"; all others are marked "inactive". An SIDX value becomes active if at least one sample description identified by that SIDX has been received. Since sample descriptions MAY be sent redundantly, it is possible that a client receives a given SIDX several times. However, active sample descriptions SHALL NOT be overwritten: The receiver SHALL ignore redundant sample descriptions and it MUST use the already cached copy. The "guard interval" of (64) inactive values ensures that the correct association SIDX <-> sample description is always used.

> Informative note: As for the "guard interval" value itself, 64 as 128/2 was considered simple enough while still meeting the expected maximum number of sample descriptions. Besides that, there's no other motivation for choosing 64 or a different value.

The following algorithm is used to buffer dynamic sample descriptions and to maintain the dynamic SIDX values:

Let X be the last SIDX received that updated the range of active sample descriptions. Let Y be a value within the allowed range for dynamic SIDX: [0,127], and different from X. Let Z be the SIDX of the last received sample description. Then:

1. Initialize all dynamic SIDX values as inactive. For stored contents, read the sample description index in the Sample to Chunk box ("stsc") for that sample. For live streaming, the first value MAY be zero or any other value in the interval above. Go to step 2.

2. First, in-band sample description with SIDX=Z is received and stored; set X=Z. Go to step 3.

3. Any SIDX within the interval [X+1 modulo(128), X+64 modulo(128)] is marked as inactive, and any corresponding sample description is deleted. Any SIDX within the interval [X+65 modulo(128), X] is set active. Go to step 4 (wait state).

4. Wait for next sample description. Once the client is initialized, the interval of active SIDX values MUST change whenever a sample description with an SIDX value in the inactive set is received. That is, upon reception of a sample description with SIDX=Z, do the following:

   a. If Z is in the (closed) interval [X+1 modulo(128), X+64 modulo(128)] then set X=Z, store the sample description, and go to step 3.

   b. Else, Z must be in the interval [X+65 modulo(128), X], thus:

      i. If SIDX=Z is not stored, then store the sample description. Go to beginning of step 4 (wait state).
      ii. Else, go to the beginning of step 4 (wait state).

   Informative note: It is allowed that any value of SIDX=X be sent in the interval [0,127]. For example, if [64..127] is the current active set and SIDX=0 is sent, a new sample description is defined (0) and an old one deleted (64); thus [65..127] and [0] are active. Similarly, one could now send SIDX=64, thus inverting the active and inactive sets.

Example:
   If X=4, any SIDX in the interval [5,68] is inactive. Active SIDX values are in the complementary interval [69,127] plus

[0,4]. For example, if the client receives a SIDX=6, then the active interval is now different: [0,6] plus [71,127]. If the received SIDX is in the current active interval, no change SHALL be applied.

4.3. Finding Payload Header Values in 3GP Files

For the purpose of streaming timed text contents, some values in the boxes contained in a 3GP file are mapped to fields of this payload header. This section explains where to find those values.

Additionally, for the duration and sample description indexes, extension mechanisms are provided. All senders MUST implement the extension mechanisms described herein.

If the file is streamed out of a 3GP file, the following guidelines SHALL be followed.

Note: All fields in the objects (boxes) of a 3GP file are found in network byte order.

Information obtained from the Sample Table Box (stbl):

- Sample Descriptions and Sample Description length: The Sample Description box (stsd, inside the stbl) contains the sample descriptions. For timed text media, each element of stsd is a timed text sample entry (type "tx3g").

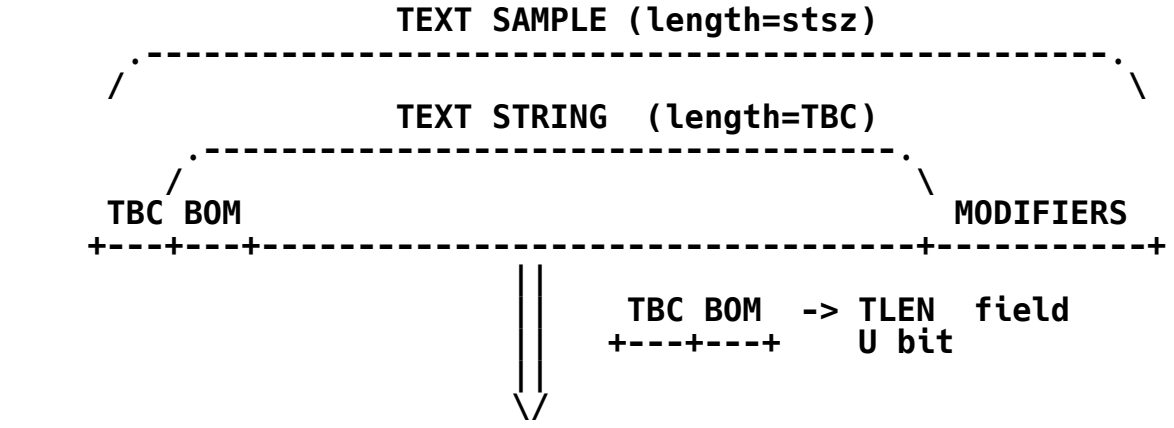
  The (unsigned) 32 bits of the "size" field in the stsd box represent the length (in bytes) of the sample description, as carried in TYPE 5 units. On the other hand, the LEN field of TYPE 5 units is restricted to 16 bits. Therefore, if the value of "size" is greater than (2^16-1-3)[bytes], then the sample description SHALL NOT be streamed with this payload format. There is no extension mechanism defined in this case, since fragmentation of sample descriptions is not defined (sample descriptions are typically up to some 200 bytes in size). Note: The three (3) accounts for the TYPE 5 header fields included in the LEN value.

- SDUR from the Decoding Time to Sample Box (stts). The (unsigned) 32 bits of the "sample delta" field are used for calculating SDUR. However, since the SDUR field is only 3 bytes long, text samples with duration values larger than (2^24-1)/(timestamp clockrate)[seconds] cannot be streamed directly. The solution is simple: Copies of the corresponding text sample SHALL be sent. Thereby, the timestamp and duration values SHALL be adjusted so that a continuous display

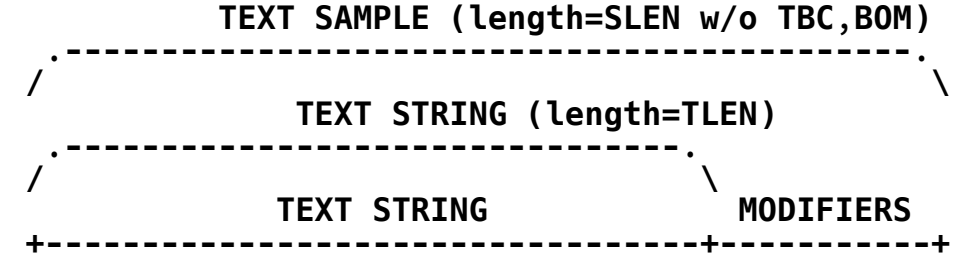
is guaranteed as if just one sample would have been sent. That is, a sample with timestamp TS and duration SDUR can be sent as two samples having timestamps TS1 and TS2 and durations SDUR1 and SDUR2, such that TS1=TS, TS2=TS1+SDUR1, and SDUR=SDUR1+SDUR2.

o Text sample length from the Sample Size Box (stsz). The (unsigned) 32 bits of the "sample size" or "entry size" (one of them, depending on whether the sample size is fixed or variable) indicate the length (in bytes) of the 3GP text sample. For obtaining the length of the (actual) streamed text sample, the lengths of the text string byte count (2 bytes) and, in case of UTF-16 strings, the length the BOM (also 2 bytes) SHALL be deducted. This is illustrated in Figure 9.

Text Sample according to 3GPP TS 26.245



Text Sample according to this Payload Format



KEY:
TBC = Text string Byte Count
BOM = Byte Order Mark

Figure 9. Text sample composition

Moreover, since the LEN field in TYPE 1 unit header is 16 bits long, larger text sample sizes than (2^16-1-8) [bytes] SHALL NOT be streamed. Also, in this case, no extension mechanism is defined. This is because this maximum is considered enough for the targeted streaming applications. (Note: The eight (8) accounts for the TYPE 1 header fields included in the LEN value).

- SIDX from the Sample to Chunk Box (stsc): The stsc Box is used to find samples and their corresponding sample descriptions. These are referenced by the "sample description index", a 32-bit (unsigned) integer. If possible, these indices may be directly mapped to the SIDX field. However, there are several cases where this may not be possible:

  a) The total number of indices used is greater than the number of indices available, i.e., if the static sample descriptions are more than 127 or the dynamic ones are more than 64.

  b) The original SIDX value ranges do not fit in the allowed ranges for static (129-254) or dynamic (0-127) values.

  Therefore, when assigning SIDX values to the sample descriptions, the following guidelines are provided:

  - Static sample descriptions can simply be assigned consecutive values within the range 129-254 (closed interval). This range should be well enough for static sample descriptions.

  - As for dynamic sample descriptions:

    a) Streams that use less than 64 dynamic sample descriptions SHOULD use consecutive values for SIDX anywhere in the range 0-127 (closed interval).

    b) For streams with more than 64 sample descriptions, the SIDX values MUST be assigned in usage order, and if any sample description shall be used after it has been set inactive, it will need to be re-sent and assigned a new SIDX value (according to the algorithm in Section 4.2.1).

Information obtained from the Media Data Box:

- Text strings, TLEN, U bit, and modifiers from the Media Data Box (mdat). Text strings, 16-bit text string byte count, Byte Order Mark (BOM, indicating UTF encoding), and modifier boxes can be found here.

  For TYPE 1 units, the value of TLEN is extracted from the text string byte count that precedes the text string in the text sample, as stored in the 3GP file. If UTF-16 encoding is used, two (2) more bytes have to be deducted from this byte count beforehand, in order to exclude the BOM. See Figure 9.

4.4. Fragmentation of Timed Text Samples

This section explains why text samples may have to be fragmented and discusses some of the possible approaches to doing it. A solution is proposed together with rules and recommendations for fragmenting and transporting text samples.

3GPP Timed Text applications are expected to operate at low bitrates. This fact, added to the small size of timed text samples (typically one or two hundred bytes) makes fragmentation of text samples a rare event. Samples should usually fit into the MTU size of the used network path.

Nevertheless, some text strings (e.g., ending roll in a movie) and some modifier boxes (i.e., for hyperlinks, for karaoke, or for styles) may become large. This may also apply for future modifier boxes. In such cases, the first option to consider is whether it is possible to adjust the encoding (e.g., the size of sample) in such a way that fragmentation is avoided. If it is, this is preferred to fragmentation and SHOULD be done.

Otherwise, if this is not possible or other constraints prevent it, fragmentation MAY be used, and the basic guidelines given in this document MUST be followed:

- It is RECOMMENDED that text samples be fragmented as seldom as possible, i.e., the least possible number of fragments is created out of a text sample.

- If there is some bitrate and free space in the payload available, sample descriptions (if at hand) SHOULD be aggregated.

- Text strings MUST split at character boundaries; see TYPE 2 header. Otherwise, it is not possible to display the text contents of a fragment if a previous fragment was lost. As a consequence, text

string fragmentation requires knowledge of the UTF-8/UTF-16 encoding formats to determine character boundaries.

- Unlike text strings, the modifier boxes are NOT REQUIRED to be split at meaningful boundaries. However, it is RECOMMENDED that this be done whenever possible. This decreases the effects of packet loss. This payload format does not ensure that partially received modifiers are applied to text strings. If only part of the modifiers is received, it is an application issue how to deal with these, i.e., whether or not to use them.

  Informative note: Ensuring that partially received modifiers can be applied to text strings in all cases (for all modifier types and for all fragment loss constellations) would place additional requirements on the payload format. In particular, this would require that: a) senders understand the semantics of the modifier boxes and b) specific fragment headers for each of the modifier boxes are defined, in addition to the payload formats defined below. Understanding the modifiers semantics means knowing, e.g., where each modifier starts and ends, which text fragments are affected, which modifiers may or may not be split, or what the fields indicate. This is necessary to be able to split the modifiers in such a way that each fragment can be applied independently of previous packet losses. This would require a more intelligent fragmentation entity and more complex headers. Given the low probability of fragmentation and the desire to keep the requirements low, it does not seem reasonable to specify such modifier box specific headers.

- Modifier and text string fragments SHOULD be protected against packet losses, i.e., using FEC [7], retransmission [11], repetition (Section 5), or an equivalent technique. This minimizes the effects of packet loss.

- An additional requirement when fragmenting text samples is that the start of the modifiers MUST be indicated using the payload header defined for that purpose, i.e., a TYPE 3 unit MUST be used (see Section 4.1.4). This enables a receiver to detect the start of the modifiers as long as there are not two or more consecutive packet losses.

- Finally, sample descriptions SHALL NOT be fragmented because they contain important information that may affect several text samples.

4.5. Reassembling Text Samples at the Receiver

The payload headers defined in this document allow reassembling fragmented text samples. For this purpose, the standard RTP timestamp, the duration field (SDUR), and the fields TOTAL/THIS in the payload headers are used.

Units that belong to the same text sample MUST have the same timestamp. TYPE 5 units do not comply with this rule since they are not part of any particular text sample.

The process for collecting the different fragments (units) of a text sample is as follows:

1. Search for units having the same timestamp value, i.e., units that belong to the same text sample or sample descriptions that shall become available at that time instant. If several units of the same sample are repeated, only one of them SHALL be used. Repeated units are those that have the same timestamp and the same values for TOTAL/THIS.

   Note that, as mentioned in Section 4.1.1, the receiver SHALL ignore units with unrecognized TYPE value. However, the RTP header fields and the rest of the units (if any) in the payload are still useful.

2. Check within this set whether any of the units from the text sample is missing. This is done using the TOTAL and THIS fields; the TOTAL field indicates how many fragments were created out of the text sample, and the THIS field indicates the position of this fragment in the text sample. As result of this operation, two outcomes are possible:

   a. No fragment is missing. Then, the THIS field SHALL be used to order the fragments and reassemble the text sample before forwarding it to the decoding application. Special care SHALL be taken when reassembling the text string as indicated in bullet 4 below.

   b. One or more fragments are missing: Check whether this fragment belongs to the text string or to the modifiers. TYPE 2 units identify text string fragments, and TYPE 3 and 4 identify modifier fragments:

      i. If the fragment or fragments missing belong to the text string and the modifiers were received complete, then the received text characters may, at least, be displayed as plain text. Some modifiers may only be

applied as long as it is possible to identify the character numbers, e.g., if only the last text string fragment is lost. This is the case for modifiers defining specific font styles ('styl'), highlighted characters ('hlit'), karaoke feature ('krok'), and blinking characters ('blnk'). Other modifiers such as 'dlay' or 'tbox' can be applied without the knowledge of the character number. It is an application issue to decide whether or not to apply the modifiers.

ii. If the fragment missing belongs to the modifiers and the text strings were received complete, then the incomplete modifiers may be used. The text string SHOULD at least be displayed as plain text. As mentioned in Section 4.4, modifiers may split without observing meaningful boundaries. Hence, it may not always be possible to make use of partially received modifiers. However, to avoid this, it is RECOMMENDED that the modifiers do split at meaningful boundaries.

iii. A third possibility is that it is not possible to discern whether modifiers or text strings were received complete. For example, if the TYPE 3 unit of a sample plus the following or preceding packet is lost, there is no way for the RTP receiver to know if one or both packets lost belong to the modifiers or if there are also some missing text strings. Repetition, FEC, retransmission, or other protection mechanisms as per section 4.6 are RECOMMENDED to avoid this situation.

iv. Finally, if it is sure that neither text strings nor modifiers were received complete, then the text strings and the modifiers may be rendered partially or may be discarded. This is an application choice.

3. Sample descriptions can be directly associated with the reassembled text samples, via the sample description index (SIDX).

4. Reassembling of text strings: Since the text strings transported in RTP packets MUST NOT include any byte order mark (BOM), the receiver MUST prepend it to the reassembled UTF-16 string before handling it to the timed text decoder (see Figure 9). The value of the BOM is 0xFEFF because only big endian serialization of UTF-16 strings is supported by this payload format.

4.6. On Aggregate Payloads

Units SHOULD be aggregated to avoid overhead, whenever possible. The aggregate payloads MUST comply with one of the following ordered configurations:

1. Zero or more sample descriptions (TYPE 5) followed by zero or more whole text samples (TYPE 1 units). At least one unit of either type MUST be present.

2. Zero or more sample descriptions followed by zero or one modifier fragment, either TYPE 3 or TYPE 4. At least one unit MUST be present.

3. Zero or more sample descriptions, followed by zero or one text string fragment (TYPE 2), followed by zero or one TYPE 3 unit. If a TYPE 2 unit and a TYPE 3 unit are present, then they MUST belong to the same text sample. At least one unit MUST be present.

Some observations:

- Different aggregates than the ones listed above SHALL NOT be used.

- Sample descriptions MUST be placed in the aggregate payload before the occurrence of any non-TYPE 5 units.

- Correct reception of TYPE 5 units is important since their contents may be referenced by several other units in the stream.

  Receivers are unable to use text samples until their corresponding sample descriptions are received. Accordingly, a sender SHOULD send multiple copies of a sample description to ensure reliability (see Section 5). Receivers MAY use payload-specific feedback messages [21] to tell a sender that they have received a particular sample description.

- Regarding timestamp calculation: In general, the rules for calculating the timestamp of units in an aggregate payload depend on the type of unit. Based on the possible constellations for aggregate payloads, as above, we have:

  - Sample descriptions MUST receive the RTP timestamp of the packet in which they are included.

    Note that for TYPE 5 units, the timestamp actually does not represent the instant when they are played out, but instead the instant at which they become available for use.

- For the first configuration: The first TYPE 1 unit receives the RTP timestamp. The timestamp of any subsequent TYPE 1 unit MUST be obtained by adding sample duration and timestamp, both of the preceding TYPE 1 unit.

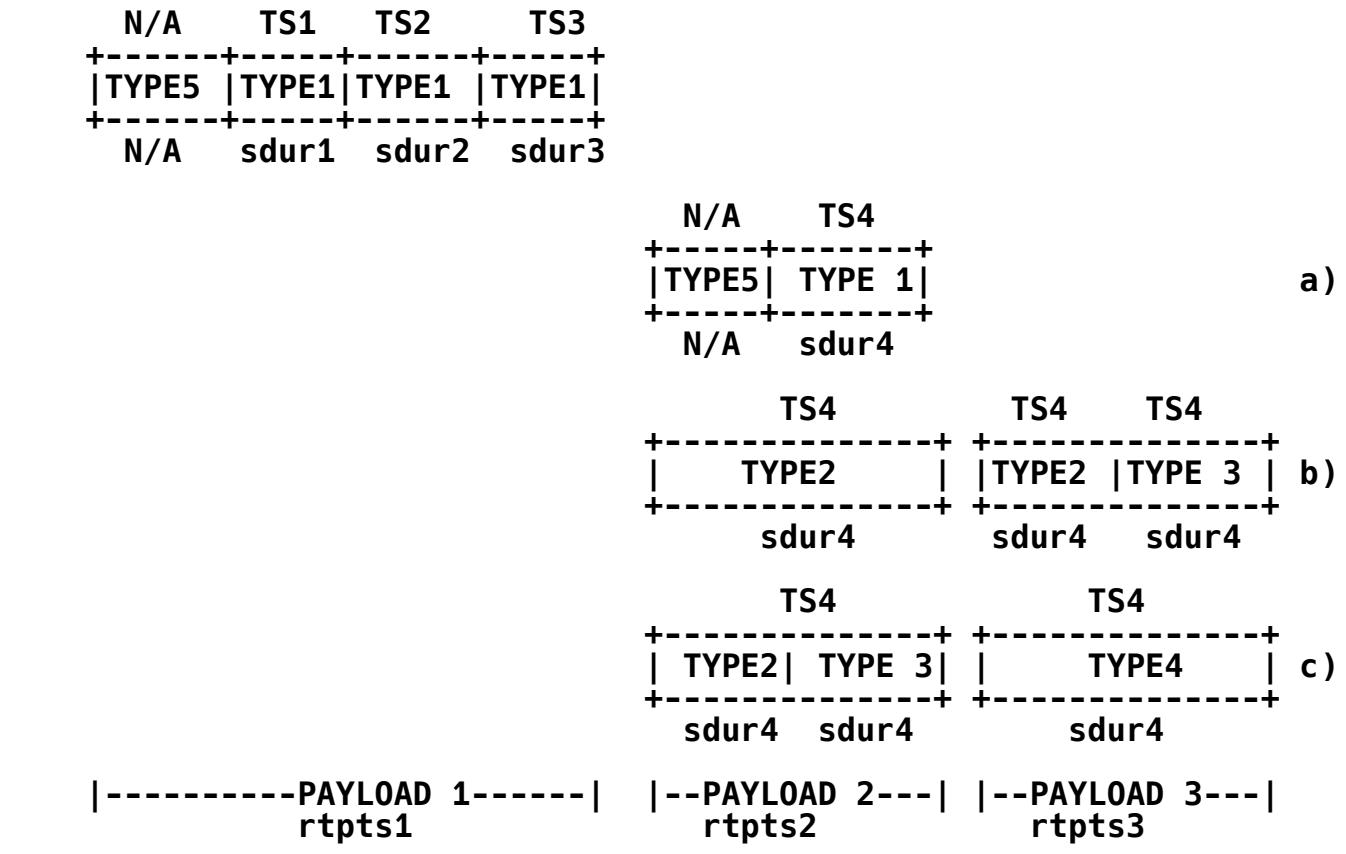
- For the second and third configuration, all units, TYPE 2, 3, and 4, MUST receive the RTP timestamp.

Refer to detailed examples on the timestamp calculation below.

- As per configuration 3 above, a payload MAY contain several fragments of one (and only one) text sample. If it does, then exactly one TYPE 2 unit followed by exactly one TYPE 3 unit is allowed in the same payload. This is in line with RFC 3640 [12], Section 2.4, which explicitly disallows combining fragments of different samples in the same RTP payload. Note that, in this special case, no timestamp calculation is needed. That is, the RTP timestamp of both units is equal to the timestamp in the packet's RTP header.

- Finally, note that the use of empty text samples allows for aggregating non-consecutive TYPE 1 units in the same payload. Two text samples, with timestamps TS1 and TS3 and durations SDUR1 and SDUR3, are not consecutive if it holds TS1+SDUR1 < TS3. A solution for this is to include an empty TYPE 1 unit with duration SDUR2 between them, such that TS2+SDUR2 = TS1+SDUR1+SDUR2 = TS3.

Some examples of aggregate payloads are illustrated in Figure 10. (Note: The figure is not scaled.)



KEY:
TSx = Text Sample x
rtptsy = the standard RTP timestamp for PAYLOAD y
sdurx = the duration of Text Sample x
N/A = not applicable

Figure 10. Example aggregate payloads

In Figure 10, four text samples (TS1 through TS4) are sent using three RTP packets. These configurations have been chosen to show how the 5 TYPE headers are used. Additionally, three different possibilities for the last text sample, TS4, are depicted: a), b), and c).

In Figure 11, option b) from Figure 10 is chosen to illustrate how the timestamp for each unit is found.

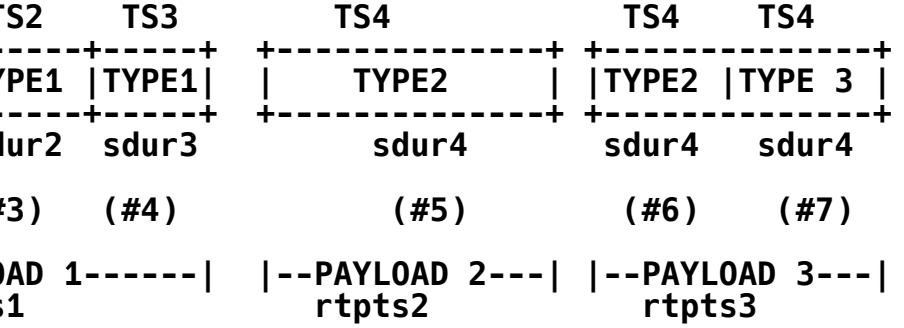


Figure 11. Selected payloads from Figure 10

Assuming TSx means Text Sample x, rtptsy represents the standard RTP timestamp for PAYLOAD y and sdurx, the duration of Text Sample x, the timestamp for unit #z, ts(#z), can be found as the sum of rtptsy and the cumulative sum of the durations of preceding units in that payload (except in the case of PAYLOAD 3 as per rule 3 above). Thus, we have:

1. for the units in the first aggregate payload, PAYLOAD 1:

```
ts(#1) = rtpts1
ts(#2) = rtpts1
ts(#3) = rtpts1 + sdur1
ts(#4) = rtpts1 + sdur1 + sdur2
```

Note that the TYPE 5 and the first TYPE 1 unit have both the RTP timestamp.

2. for PAYLOAD 2:

```
ts(#5) = rtpts2
```

3. for PAYLOAD 3:

```
ts(#6) = ts(#7) = rtpsts2 = rtpts3
```

According to configuration 3 above, the TYPE2 and the TYPE 3 units shall belong to the same sample. Hence, rtpts3 must be equal to rtpts2. For the same reason, the value of SDUR is not be used to calculate the timestamp of the next unit.

4.7. Payload Examples

Some examples of payloads using the defined headers are shown below:

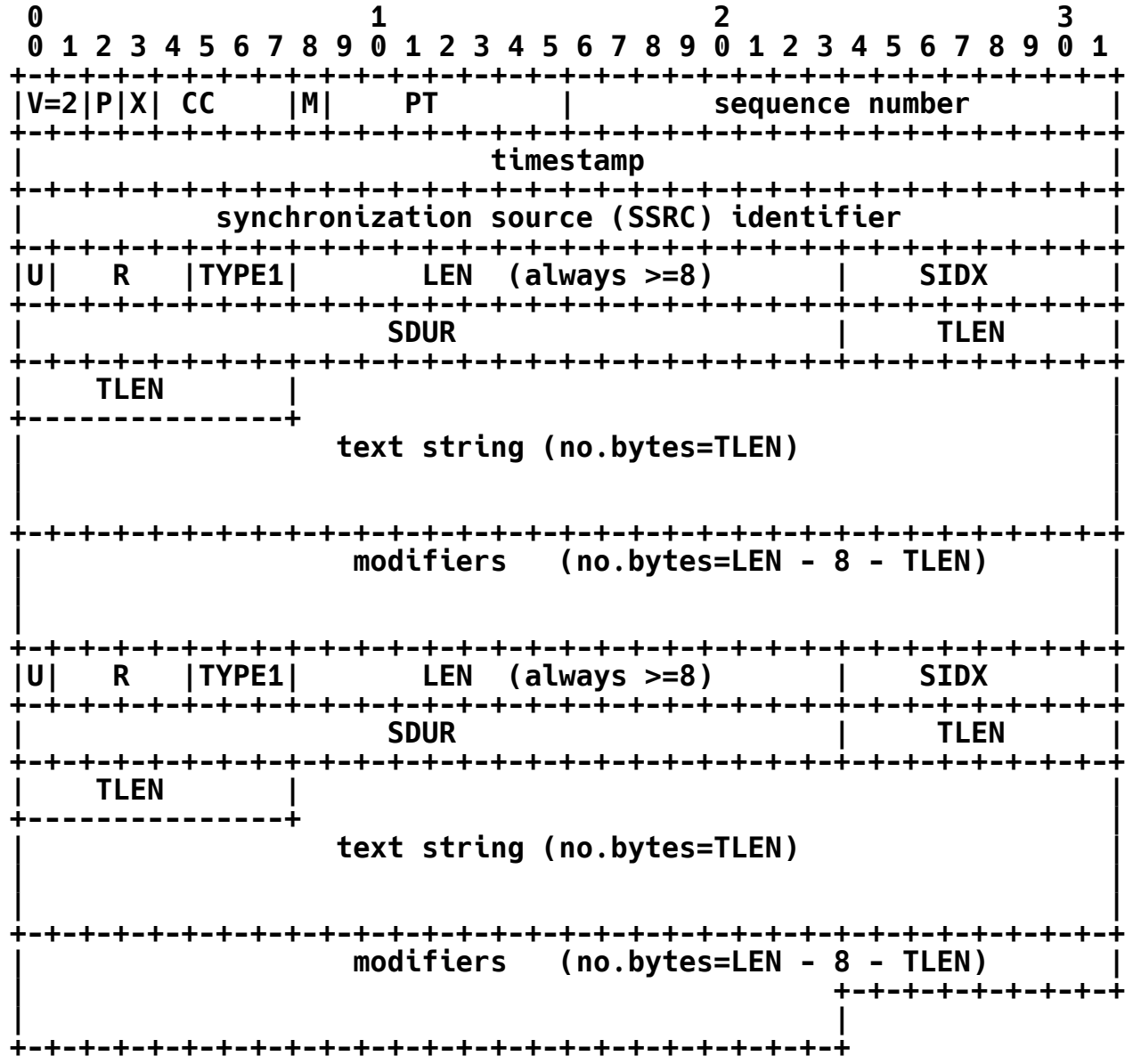


Figure 12. A payload carrying two TYPE 1 units

In Figure 12, an RTP packet carrying two TYPE 1 units is depicted. It can be seen how the length fields LEN and TLEN can be used to find the start of the next unit (LEN), the start of the modifiers (TLEN), and the length of the modifiers (LEN-TLEN).

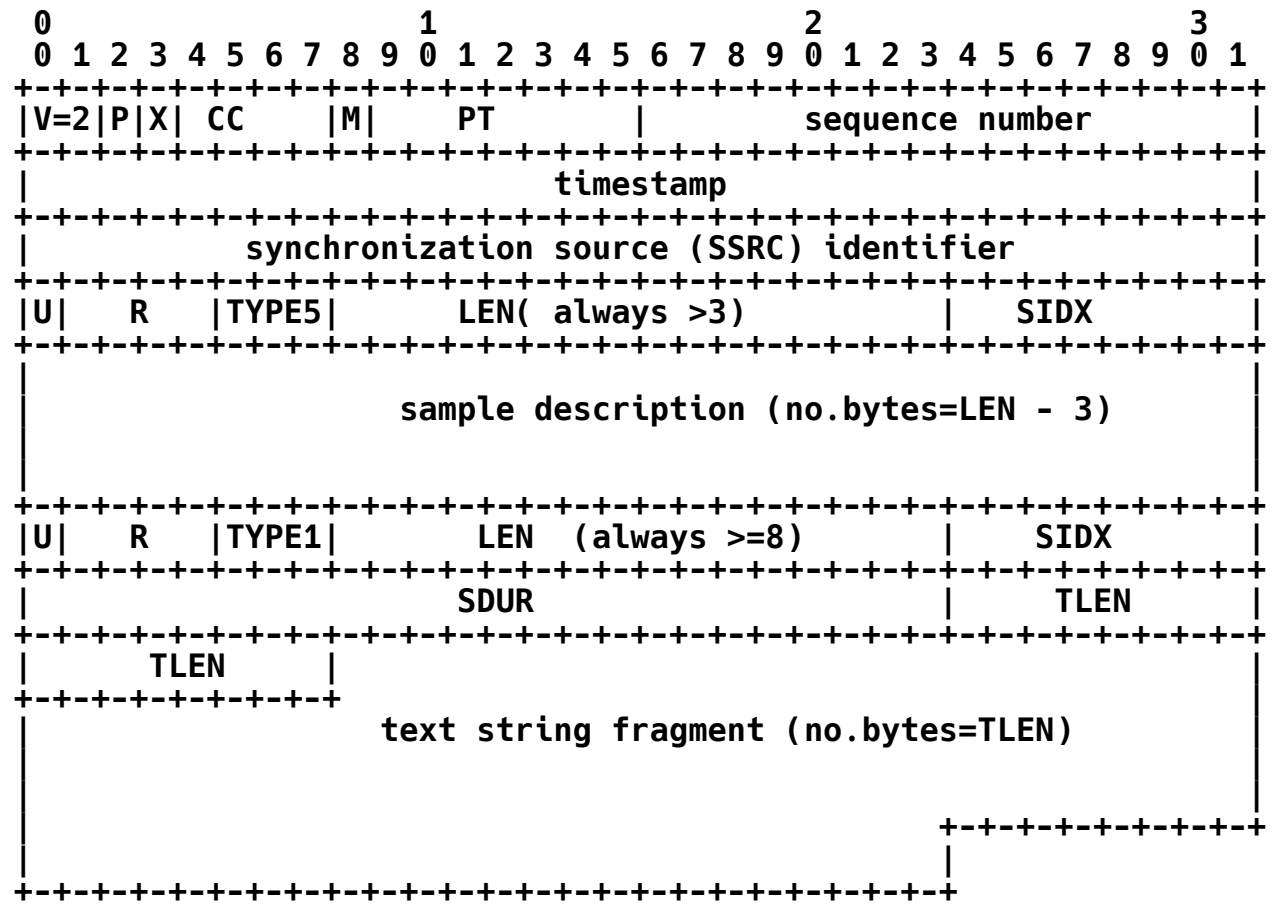


Figure 13. An RTP packet carrying a TYPE 5 and a TYPE 1 unit

In Figure 13, a sample description and a TYPE 1 unit are aggregated. The TYPE 1 unit happens to contain only text strings and is small, so an additional TYPE 5 unit is included to take advantage of the available bits in the packet.

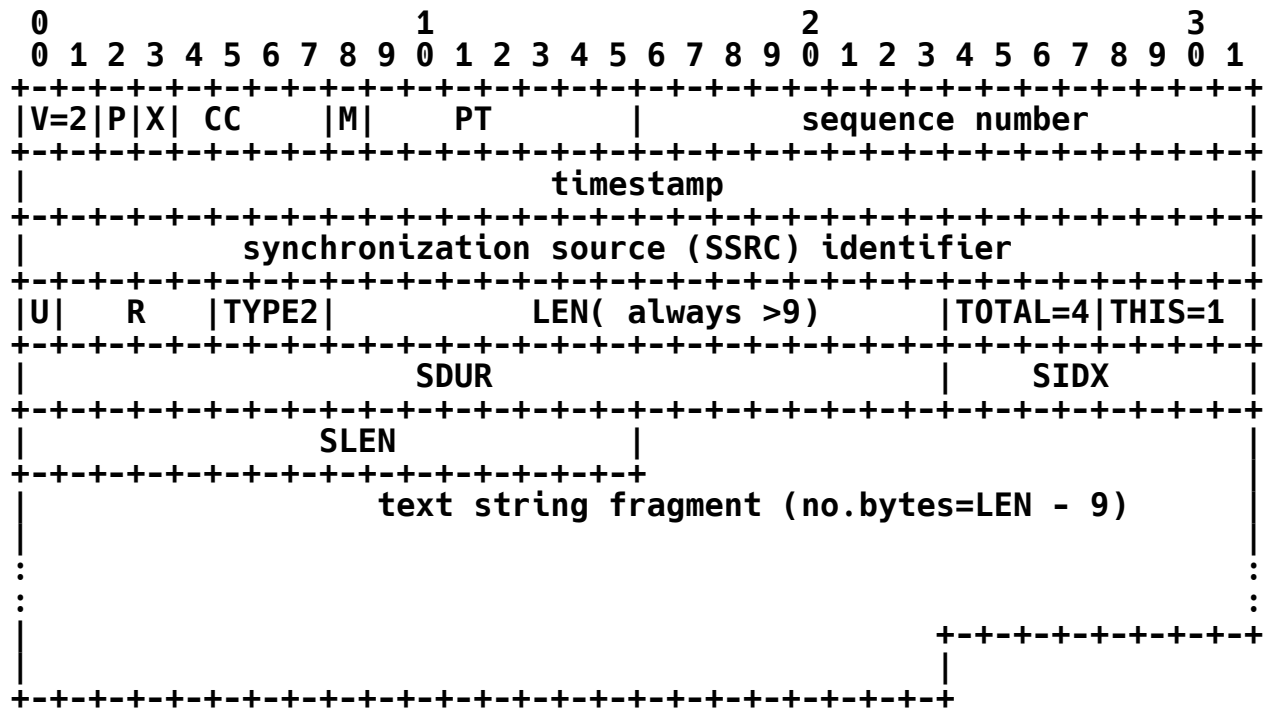


Figure 14. Payload with first text string fragment of a sample

In Figures 14, 15, and 16, a text sample is split into three RTP packets. In Figure 14, the text string is big and takes the whole packet length. In Figure 15, the only possibility for carrying two fragments of the same text sample is represented (see configuration 3 in Section 4.6). The last packet, shown in Figure 16, carries the last modifier fragment, a TYPE 4.

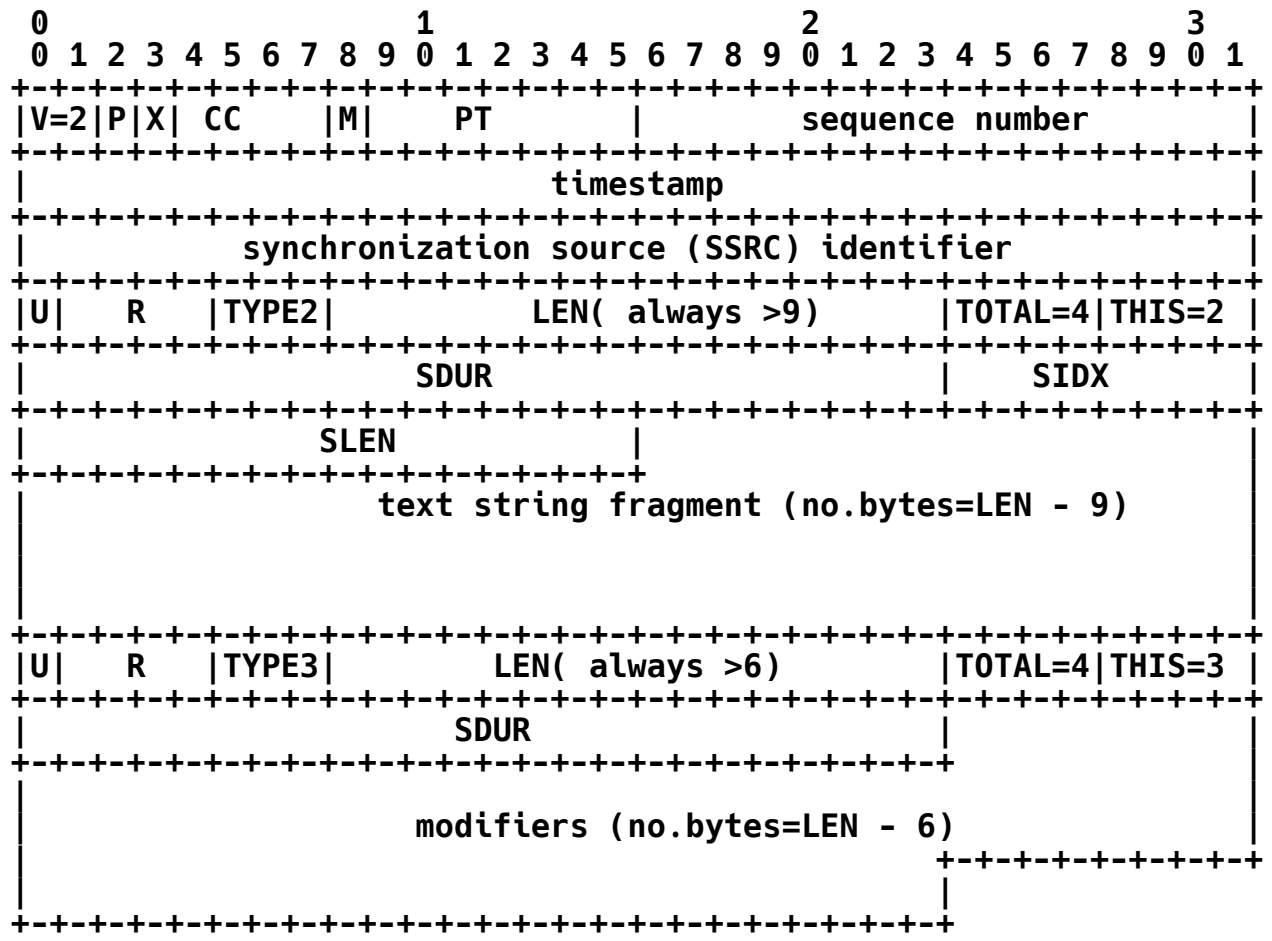


Figure 15. An RTP packet carrying a TYPE 2 unit and a TYPE 3 unit

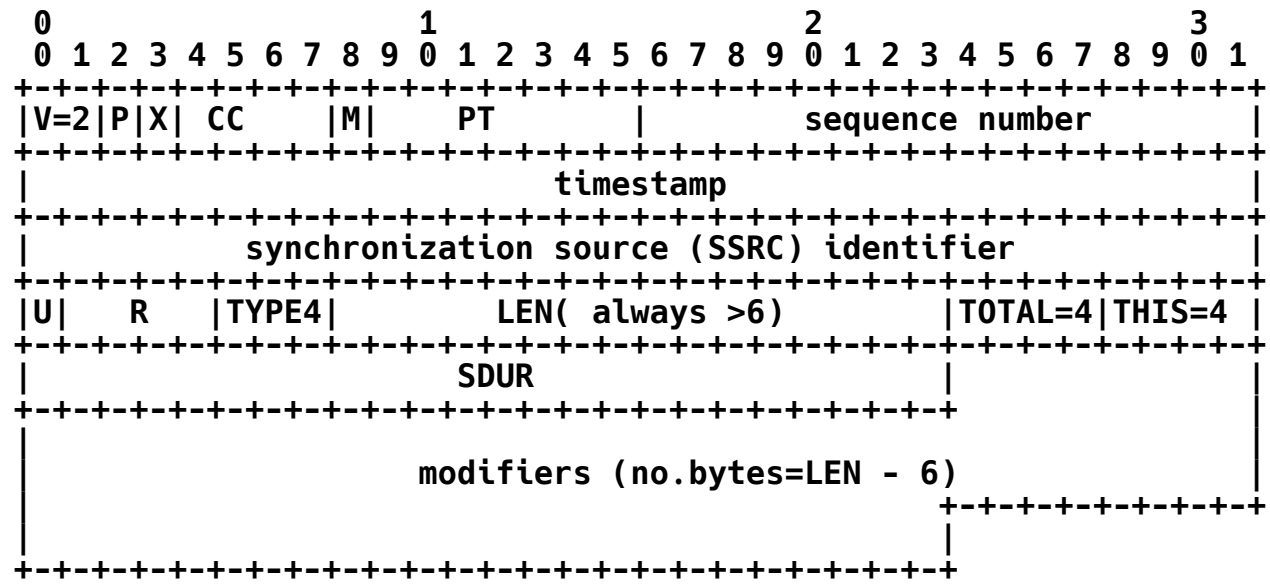


Figure 16. An RTP packet carrying last modifiers fragment (TYPE 4)

4.8. Relation to RFC 3640

RFC 3640 [12] defines a payload format for the transport of any non-multiplexed MPEG-4 elementary stream. One of the various MPEG-4 elementary stream types is MPEG-4 timed text streams, specified in MPEG-4 part 17 [26], also known as ISO/IEC 14496-17. MPEG-4 timed text streams are capable of carrying 3GPP timed text data, as specified in 3GPP TS 26.245 [1].

MPEG-4 timed text streams are intentionally constructed so as to guarantee interoperability between RFC 3640 and this payload format. This means that the construction of the RTP packets carrying timed text is the same. That is, the MPEG-4 timed text elementary stream as per ISO/IEC 14496-17 is identical to the (aggregate) payloads constructed using this payload format.

Figure 17 illustrates the process of constructing an RTP packet containing timed text. As can be seen in the partition block, the (transport) units used in this payload format are identical to the Timed Text Units (TTUs) defined in ISO/IEC 14496-17. Likewise, the rules for payload aggregation as per Section 4.6 are identical to those defined in ISO/IEC 14496-17 and are compliant with RFC 3640. As a result, an RTP packet that uses this payload format is identical to an RTP packet using RFC 3640 conveying TTUs according to ISO/IEC 14496-17. In particular, MPEG-4 Part 17 specifies that when using

RFC 3640 for transporting timed text streams, the "streamType" parameter value is set to 0x0D, and the value of the "objectTypeIndication" in "config" takes the value 0x08.

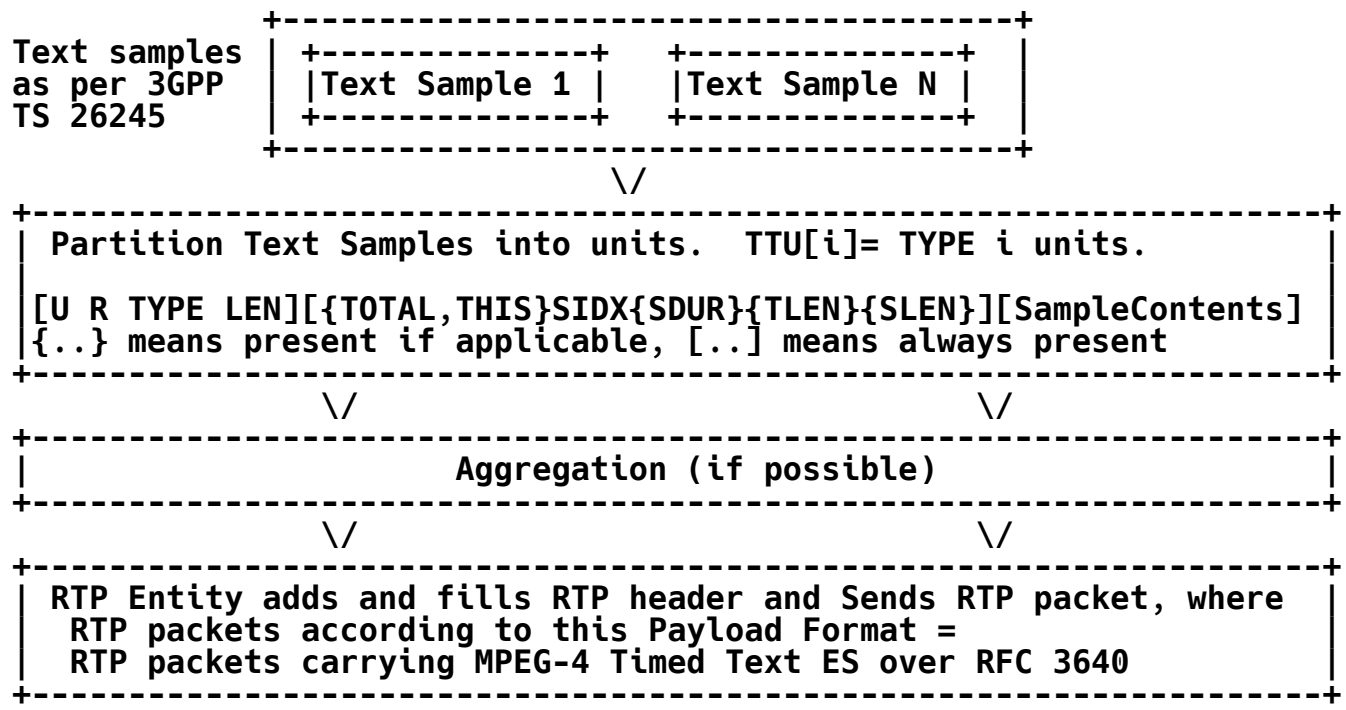


Figure 17. Relation to RFC 3640

Note: The use of RFC 3640 for transport of ISO/IEC 14496-17 data does not require any new SDP parameters or any new mode definition.

4.9. Relation to RFC 2793

RFC 2793 [22] and its revision, RFC 4103 [23], specify a protocol for enabling text conversation. Typical applications of this payload format are text communication terminals and text conferencing tools. Text session contents are specified in ITU-T Recommendation T.140 [24]. T.140 text is UTF-8 coded as specified in T.140 [24] with no extra framing. The T140block contains one or more T.140 code elements as specified in T.140. Code elements are control sequences such as "New Line", "Interrupt", "String Terminator", or "Start of String". Most T.140 code elements are single ISO 10646 [25] characters, but some are multiple character sequences. Each character is UTF-8 encoded [18] into one or more octets.

This payload format may also be used for conversational applications (even for instant messaging). However, this is not its main target. The differentiating feature of 3GPP Timed Text media format is that it allows text decoration. This is especially useful in multimedia presentations, karaoke, commercial banners, news tickers, clickable text strings, and captions. T.140 text contents used in RFC 2793 do not allow the use of text decoration.

Furthermore, the conversational text RTP payload format recommends a method to include redundant text from already transmitted packets in order to reduce the risk of text loss caused by packet loss. Thereby payloads would include a redundant copy of the last payload sent. This payload format does not describe such a method, but this is also applicable here. As explained in Section 5, packet redundancy SHOULD be used, whenever possible. The aggregation guidelines in Section 4.6 allow redundant payloads.

5. Resilient Transport

Apart from the basic fragmentation guidelines described in the section above, the simplest option for packet-loss-resilient transport is packet repetition. This mechanism may consist of a strict window-based repetition mechanism or, simply, a repetition mechanism in a wider sense, where new and old packets are mixed, for example.

A server MAY decide to use repetition as a measure for packet loss resilience. Thereby, a server MAY send the same RTP payloads or just some of the units from the payloads.

As for the case of complete payloads, single repeated units MUST exactly match the same units sent in the first transmission; i.e., if fragmentation is needed, it SHALL be performed only once for each text sample. Only then, a receiver can use the already received and the repeated units to reconstruct the original text samples. Since the RTP timestamp is used to group together the fragments of a sample, care must taken to preserve the timing of units when constructing new RTP packets.

> For example, if a text sample was originally sent as a single non-fragmented text sample (one TYPE 1 unit), a repetition of that sample MUST be sent also as a single non-fragmented text sample in one unit. Likewise, if the original text sample was fragmented and spread over several RTP packets (say, a total of 3 units), then the repeated fragments SHALL also have the same byte boundaries and use the same unit headers and bytes per fragment.

With repetition, repeated units resolve to the same timestamp as their originals. Where redundant units are available, only one of them SHALL be used.

Regarding the RTP header fields:

- If the whole RTP payload is repeated, all payload-specific fields in the RTP header (the M, TS and PT fields) MUST keep their original values except the sequence number, which MUST be incremented to comply with RTP (the fields TOTAL/THIS enable to re-assemble fragments with different sequence numbers).

- In packets containing single repeated units, the general rules in Section 3 for assigning values to the RTP header fields apply. Keeping the value of the RTP timestamp to preserve the timing of the units is particularly relevant here.

Apart from repetition, other mechanisms such as FEC [7], retransmission [11], or similar techniques could be used to cope with packet losses.

6. Congestion Control

Congestion control for RTP SHALL be implemented in accordance with RTP [3] and the applicable RTP profile, e.g., RTP/AVP [17].

When using this payload format, mainly two factors may affect the congestion control:

- The use of (unit) aggregation may make the payload format more bandwidth efficient, by avoiding header overhead and thus reducing the used bitrate.

- The use of resilient transport mechanisms: Although timed text applications typically operate at low bitrates, the increase due to resilient transport shall be considered for congestion control mechanisms. This applies to all mechanisms but especially to less efficient ones like repetition.

7. Scene Description

7.1. Text Rendering Position and Composition

In order to set up a timed text session, regardless of the stream being stored in a 3GP file or streamed live, some initial layout information is needed by the communicating peers.

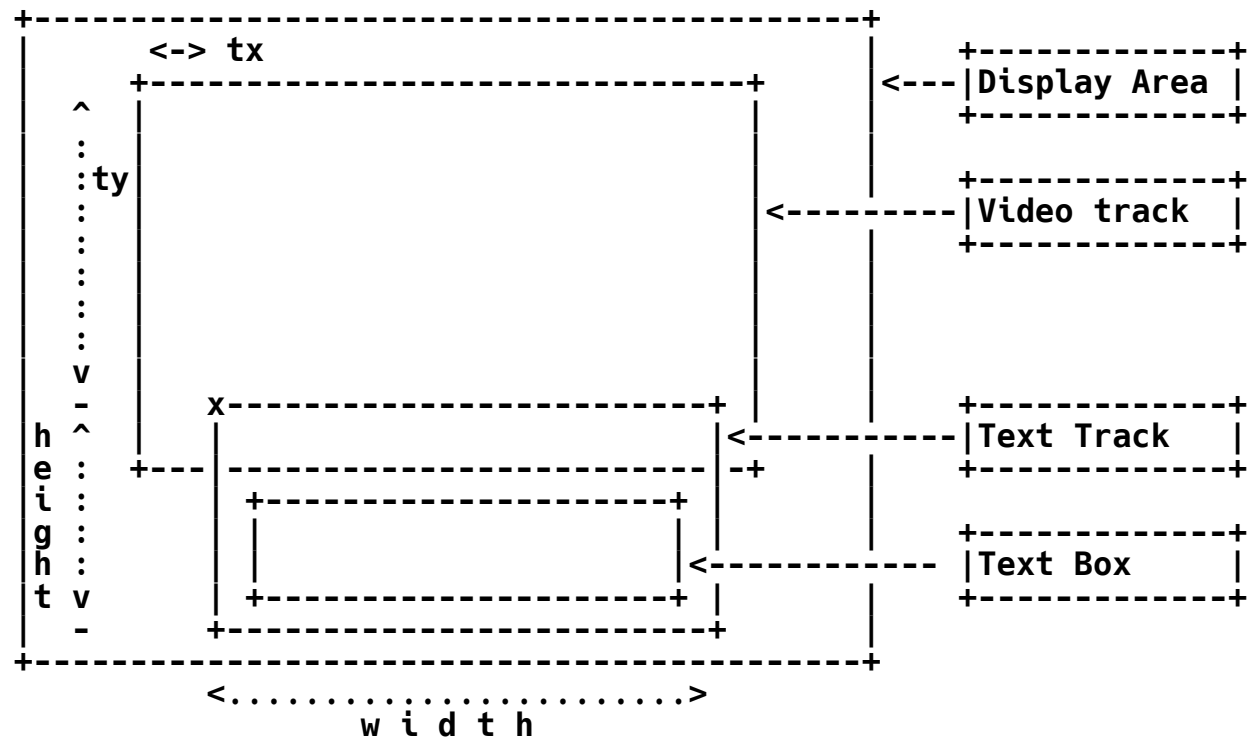


Figure 18. Illustration of text rendering position and composition

The parameters used for negotiating the position and size of the text track in the display area are shown in Figure 18. These are the "width" and "height" of the text track, its translation values, "tx" and "ty", and its "layer" or proximity to the user.

At the same time, the sender of the stream needs to know the receiver's capabilities. In this case, the maximum allowable values for the text track height and width: "max-h" and "max-w", for the stream the receiver shall display.

This layout information MUST be conveyed in a reliable form before the start of the session, e.g., during session announcement or in an Offer/Answer (O/A) exchange. An example of a reliable transport may be the out-of-band channel used for SDP. Sections 8 and 9 provide

details on the mapping of these parameters to SDP descriptions and their usage in O/A.

For stored content, the layout values expressing stream properties MUST be obtained from the Track Header Box. See Section 7.3.

For live streaming, appropriate values as negotiated during session setup shall be used.

7.2. SMIL Usage

The attributes contained in the Track Header Boxes of a 3GP file only specify the spatial relationship of the tracks within the given 3GP file.

If multiple 3GP files are sent, they require spatial synchronization. For example, for a text and video stream, the positions of the text and video tracks in Figure 18 shall be determined. For this purpose, SMIL [9] MAY be used.

SMIL assigns regions in the display to each of those files and places the tracks within those regions. Generally, in SMIL, the position of one track (or stream) is expressed relative to another track. This is different from the 3GP file, where the upper left corner is the reference for all translation offsets. Hence, only if the position in SMIL is relative to the video track origin, then this translation offset has the same value as (tx, ty) in the 3GP file.

Note also that the original track header information is used for each track only within its region, as assigned by SMIL. Therefore, even if SMIL scene description is used, the track header information pieces SHOULD be sent anyway, as they represent the intrinsic media properties. See 3GPP SMIL Language Profile in [27] for details.

7.3. Finding Layout Values in a 3GP File

In a 3GP file, within the Track Header Box (tkhd):

- tx, ty: These values specify the translation offset of the (text) track relative to the upper left corner of the video track, if present. They are the second but last and third but last values in the unity matrix; values are fixed-point 16.16 values, restricted to be (signed) integers (i.e., the lower 16 bits of each value shall be all zeros). Therefore, only the first 16 bits are used for obtaining the value of the media type parameters.

- width, height: They have the same name in the tkhd box. All (unsigned) 32 bits are meaningful.

- layer: All (signed) 16 bits are used.

8. 3GPP Timed Text Media Type

The media subtype for the 3GPP Timed Text codec is allocated from the standards tree. The top-level media type under which this payload format is registered is 'video'. This registration is done using the template defined in [29] and following RFC 3555 [28].

The receiver MUST ignore any unrecognized parameter.

Media type: video

Media subtype: 3gpp-tt

Required parameters

rate:
: Refer to Section 3 in RFC 4396.

sver:
: The parameter "sver" contains a list of supported backwards-compatible versions of the timed text format specification (3GPP TS 26.245) that the sender accepts to receive (and that are the same that it would be willing to send). The first value is the value preferred to receive (or preferred to send). The first value MAY be followed by a comma-separated list of versions that SHOULD be used as alternatives. The order is meaningful, being first the most preferred and last the least preferred. Each entry has the format Zi(xi*256+yi), where "Zi" is the number of the Release and "xi" and "yi" are taken from the 3GPP specification version (i.e., vZi.xi.yi). For example, for 3GPP TS 26.245 v6.0.0, Zi(xi*256+yi)=6(0), the version value is "60". (Note that "60" is the concatenation of the values Zi=6 and (xi*256+yi)=0 and not their product.)

  If no "sver" value is available, for example, when streaming out of a 3GP file, the default value "60", corresponding to the 3GPP Release 6 version of 3GPP TS 26.245, SHALL be used.

Optional parameters:

tx:
This parameter indicates the horizontal translation offset in pixels of the text track with respect to the origin of the video track. This value is the decimal representation of a 16-bit signed integer. Refer to TS 3GPP 26.245 for an illustration of this parameter.

ty:
This parameter indicates the vertical translation offset in pixels of the text track with respect to the origin of the video track. This value is the decimal representation of a 16-bit signed integer. Refer to TS 3GPP 26.245 for an illustration of this parameter.

layer:
This parameter indicates the proximity of the text track to the viewer. More negative values mean closer to the viewer. This parameter has no units. This value is the decimal representation of a 16-bit signed integer.

tx3g:
This parameter MUST be used for conveying sample descriptions out-of-band. It contains a comma-separated list of base64-encoded entries. The entries of this list MAY follow any particular order and the list SHALL NOT be empty. Each entry is the result of running base64 encoding over the concatenation of the (static) SIDX value as an 8-bit unsigned integer and the (static) sample description for that SIDX, in that order. The format of a sample description entry can be found in 3GPP TS 26.245 Release 6 and later releases. All servers and clients MUST understand this parameter and MUST be capable of using the sample description(s) contained in it. Please refer to RFC 3548 [6] for details on the base64 encoding.

width:
This parameter indicates the width in pixels of the text track or area of the text being sent. This value is the decimal representation of a 32-bit unsigned integer. Refer to TS 3GPP 26.245 for an illustration of this parameter.

height:
This parameter indicates the height in pixels of the text track being sent. This value is the decimal representation of a 32-bit unsigned integer. Refer to TS 3GPP 26.245 for an illustration of this parameter.

max-w:
This parameter indicates display capabilities. This is the maximum "width" value that the sender of this parameter supports. This value is the decimal representation of a 32-bit unsigned integer.

max-h:
This parameter indicates display capabilities. This is the maximum "height" value that the sender of this parameter supports. This value is the decimal representation of a 32-bit unsigned integer.

Encoding considerations:

This media type is framed (see Section 4.8 in [29]) and partially contains binary data.

Restrictions on usage:

This media type depends on RTP framing, and hence is only defined for transfer via RTP [3]. Transport within other framing protocols is not defined at this time.

Security considerations:

Please refer to Section 11 of RFC 4396.

Interoperability considerations:

The 3GPP Timed Text media format and its file storage is specified in Release 6 of 3GPP TS 26.245, "Transparent end-to-end packet switched streaming service (PSS); Timed Text Format (Release 6)". Note also that 3GPP may in future releases specify extensions or updates to the timed text media format in a backwards-compatible way, e.g., new modifier boxes or extensions to the sample descriptions. The payload format defined in RFC 4396 allows for such extensions. For future 3GPP Releases of the Timed Text Format, the parameter "sver" is used to identify the exact specification used.

The defined storage format for 3GPP Timed Text format is the 3GPP File Format (3GP) [30]. 3GP files may be transferred using the media type video/3gpp as registered by RFC 3839 [31]. The 3GPP File Format is a container file that may contain, e.g., audio and video that may be synchronized with the 3GPP Timed Text.

Published specification: RFC 4396

Applications which use this media type:

Multimedia streaming applications.

Additional information:

The 3GPP Timed Text media format is specified in 3GPP TS 26.245, "Transparent end-to-end packet switched streaming service (PSS); Timed Text Format (Release 6)". This document and future extensions to the 3GPP Timed Text format are publicly available at http://www.3gpp.org.

Magic number(s): None.

File extension(s): None.

Macintosh File Type Code(s): None.

Person & email address to contact for further information:

Jose Rey, jose.rey@eu.panasonic.com
Yoshinori Matsui, matsui.yoshinori@jp.panasonic.com
Audio/Video Transport Working Group.

Intended usage: COMMON

Authors:
Jose Rey
Yoshinori Matsui

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

9. SDP Usage

9.1. Mapping to SDP

The information carried in the media type specification has a specific mapping to fields in SDP [4]. If SDP is used to specify sessions using this payload format, the mapping is done as follows:

- The media type ("video") goes in the SDP "m=" as the media name.

  m=video <port number> RTP/<RTP profile> <dynamic payload type>

- The media subtype ("3gpp-tt") and the timestamp clockrate "rate" (the RECOMMENDED 1000 Hz or other value) go in SDP "a=rtpmap" line as the encoding name and rate, respectively:

  a=rtpmap:<payload type> 3gpp-tt/1000

- The REQUIRED parameter "sver" goes in the SDP "a=fmtp" attribute by copying it directly from the media type string as a semicolon-separated parameter=value pair.

- The OPTIONAL parameters "tx", "ty", "layer", "tx3g", "width", "height", "max-w" and "max-h" go in the SDP "a=fmtp" attribute by copying them directly from the media type string as a semicolon separated list of parameter=value(s) pairs:

  a=fmtp:<dynamic payload type> <parameter name>=<value>[,<value>][; <parameter name>=<value>]

- Any parameter unknown to the device that uses the SDP SHALL be ignored. For example, parameters added to the media format in later specifications MAY be copied into the SDP and SHALL be ignored by receivers that do not understand them.

9.2. Parameter Usage in the SDP Offer/Answer Model

In this section, the meaning of the SDP parameters defined in this document within the Offer/Answer [13] context is explained.

In unicast, sender and receiver typically negotiate the streams, i.e., which codecs and parameter values are used in the session. This is also possible in multicast to a lesser extent.

Additionally, the meaning of the parameters MAY vary depending on which direction is used. In the following sections, a "<directionality> offer" means an offer that contains a stream set to <directionality>. <directionality> may take the values sendrecv,

sendonly, and recvonly. Similar considerations apply for answers. For example, an answer to a sendonly offer is a recvonly answer.

9.2.1. Unicast Usage

The following types of parameters are used in this payload format:

1. Declarative parameters: Offerer and answerer declare the values they will use for the incoming (sendrecv/recvonly) or outgoing (sendonly) stream. Offerer and answerer MAY use different values.

   a. "tx", "ty", and "layer": These are parameters describing where the received text track is placed. Depending on the directionality:

      i. They MUST appear in all sendrecv offers and answers and in all recvonly offers and answers (thus applying to the incoming stream). In the case of sendrecv offers and answers and in recvonly offers, these values SHOULD be used by the sender of the stream unless it has a particular preference, in which case, it MUST make sure that these different values do not corrupt the presentation. For recvonly answers, the answerer MAY accept the proposed values for the incoming stream (in a sendonly offer; see ii. below) or respond with different ones. The offerer MUST use the returned values.

      ii. They MAY appear in sendonly offers and MUST appear in sendonly answers. In sendonly offers, they specify the values that the offerer proposes for sending (see example in Section 9.3). In sendonly answers, these values SHOULD be copied from the corresponding recvonly offer upon accepting the stream, unless a particular preference by the receiver of the stream exists, as explained in the previous point.

2. Parameters describing the display capabilities, "max-h" and "max-w", which indicate the maximum dimensions of the text track (text display area) for the incoming stream "tx" and "ty" values (see Figure 18). "max-h" and "max-w" MUST be included in all offers and answers where "tx" and "ty" refer to the incoming stream, thus excluding sendonly offers and answers (see example in Section 9.3), where they SHALL NOT be present.

3. Parameters describing the sent stream properties, i.e., the sender of the stream decides upon the values of these:

   a. "width" and "height" specify the text track dimensions. They SHALL ALWAYS be present in sendrecv and sendonly offers and answers. For recvonly answers, the answerer MUST include the offered parameter values (if any) verbatim in the answer upon accepting the stream.

   b. "tx3g" contains static sample descriptions. It MAY only be present in sendrecv and sendonly offers and answers. This parameter applies to the stream that offerers or answerers send.

4. Negotiable parameters, which MUST be agreed on. This is the case of "sver". This parameter MUST be present in every offer and answer. The answerer SHALL choose one supported value from the offerer's list, or else it MUST remove the stream or reject the session.

5. Symmetric parameters: "rate", timestamp clockrate, belongs to this class. Symmetric parameters MUST be echoed verbatim in the answer. Otherwise, the stream MUST be removed or the session rejected.

The following table summarizes all options:

Type of Parameter \ Directionality/ O or A		sendrecv O/A	recvonly O/A	sendonly O/A
Declarative	tx, ty, layer	M/M	M/M	m/M
Display Capabilities	max-h, max-w	M/M	M/M	-/-
Stream properties	height, width	M/M	-/(M)	M/M
	tx3g	m/m	-/-	m/m
Negotiable	sver	M/M	M/M	M/M
Symmetric	rate	M/M	M/M	M/M

Table 1. Parameter usage in Unicast Offer / Answer.

KEY:

- M means MUST be present.
- m means MAY be present (such as proposed values).
- (M) or (m) means MUST or MAY, if applicable.
- a hyphen ("-") means the parameter MUST NOT be present.

Other observations regarding parameter usage:

- Translation and transparency values: In sendonly offers, "tx", "ty", and "layer" indicate proposed values. This is useful for visually composed sessions where the different streams occupy different parts of the display, e.g., a video stream and the captions. These are just suggested values; the peer rendering the text ultimately decides where to place the text track.

- Text track (area) dimensions, "height" and "width": In the case of sendonly offers, an answerer accepting the offer MUST be prepared to render the stream using these values. If any of these conditions are not met, the stream MUST be removed or the session rejected.

- Display capabilities, "max-h" and "max-w": An answerer sending a stream SHALL ensure that the "height" and "width" values in the answer are compatible with the offerer's signaled capabilities.

- Version handling via "sver": The idea is that offerer and answerer communicate using the same version. This is achieved by letting the answerer choose from a list of supported versions, "sver". For recvonly streams, the first value in the list is the preferred version to receive. Consequently, for sendonly (and sendrecv) streams, the first value is the one preferred for sending (and receiving). The answerer MUST choose one value and return it in the answer. Upon receiving the answer, the offerer SHALL be prepared to send (sendonly and sendrecv) and receive (recvonly and sendrecv) a stream using that version. If none of the versions in the list is supported, the stream MUST be removed or the session rejected. Note that, if alternative non-compatible versions are offered, then this SHALL be done using different payload types.

9.2.2. Multicast Usage

In multicast, the parameter usage is similar to the unicast case, except as follows:

- the parameters "tx", "ty", and "layer" in multicast offers only have meaning for sendrecv and recvonly streams. In order for all clients to have the same vision of the session, they MUST be used symmetrically.

- for "height", "width", and "tx3g" (for sendrecv and sendonly), multicast offers specify which values of these parameters the participants MUST use for sending. Thus, if the stream is accepted, the answerer MUST also include them verbatim in the answer (also "tx3g", if present).

- The capability parameters, "max-h" and "max-w", SHALL NOT be used in multicast. If the offered text track should change in size, a new offer SHALL be used instead.

- Regarding version handling:

  In the case of multicast offers, an answerer MAY accept a multicast offer as long as one of the versions listed in the "sver" is supported. Therefore, if the stream is accepted, the answerer MUST choose its preferred version, but, unlike in unicast, the offerer SHALL NOT change the offered stream to this chosen version because there may be other session participants that do support the newer extensions. Consequently, different session participants may end up using different backwards-compatible media format versions. It is RECOMMENDED that the multicast offer contains a limited number of versions, in order for all participants to have the same view of the session. This is a responsibility of the session creator. If

none of the offered versions is supported, the stream SHALL be removed or the session rejected. Also in this case, if alternative non-compatible versions are offered, then this SHALL be done using different payload types.

9.3. Offer/Answer Examples

In these unicast O/A examples, the long lines are wrapped around. Static sample descriptions are shortened for clarity.

For sendrecv:

O -> A

```
m=video <port> RTP/AVP 98
a=rtpmap:98 3gpp-tt/1000
a=fmtp:98 tx=100; ty=100; layer=0; height=80; width=100; max-h=120;
max-w=160; sver=6256,60; tx3g=81...
a=sendrecv
```

A -> O

```
m=video <port> RTP/AVP 98..
a=rtpmap:98 3gpp-tt/1000
a=fmtp:98 tx=100; ty=95; layer=0; height=90; width=100; max-h=100;
max-w=160; sver=60; tx3g=82...
a=sendrecv
```

In this example, the offerer is telling the answerer where it will place the received stream and what is the maximum height and width allowable for the stream that it will receive. Also, it tells the answerer the dimensions of the text track for the stream sent and which sample description it shall use. It offers two versions, 6256 and 60. The answerer responds with an equivalent set of parameters for the stream it receives. In this case, the answerer's "max-h" and "max-w" are compatible with the offerer's "height" and "width". Otherwise, the answerer would have to remove this stream, and the offerer would have to issue a new offer taking the answerer's capabilities into account. This is possible only if multiple payload types are present in the initial offer so that at least one of them matches the answerer's capabilities as expressed by "max-h" and "max-w" in the negative answer. Note also that the answerer's text box dimensions fit within the maximum values signaled in the offer. Finally, the answerer chooses to use version 60 of the timed text format.

For recvonly:

Offerer -> Answerer

```
m=video <port> RTP/AVP 98
a=rtpmap:98 3gpp-tt/1000
a=fmtp:98 tx=100; ty=100; layer=0; max-h=120; max-w=160; sver=6256,60
a=recvonly
```

A -> O

```
m=video <port> RTP/AVP 98..
a=rtpmap:98 3gpp-tt/1000
a=fmtp:98 tx=100; ty=100; layer=0; height=90; width=100; sver=60;
tx3g=82...
a=sendonly
```

In this case, the offer is different from the previous case: It does not include the stream properties "height", "width", and "tx3g". The answerer copies the "tx", "ty", and "layer" values, thus acknowledging these. "max-h" and "max-w" are not present in the answer because the "tx" and "ty" (and "layer") in this special case do not apply to the received stream, but to the sent stream. Also, if offerer and answerer had very different display sizes, it would not be possible to express the answerer's capabilities. In the example above and for an answerer with a 50x50 display, the translation values are already out of range.

For sendonly:

O -> A

```
m=video <port> RTP/AVP 98
a=rtpmap:98 3gpp-tt/1000
a=fmtp:98 tx=100; ty=100; layer=0; height=80; width=100;
sver=6256,60; tx3g=81...
a=sendonly
```

A -> O

```
m=video <port> RTP/AVP 98..
a=rtpmap:98 3gpp-tt/1000
a=fmtp:98 tx=100; ty=100; layer=0; height=80; width=100; max-h=100;
max-w=160; sver=60
a=recvonly
```

Note that "max-h" and "max-w" are not present in the offer. Also, with this answer, the answerer would accept the offer as is (thus echoing "tx", "ty", "height", "width", and "layer") and additionally inform the offerer about its capabilities: "max-h" and "max-w".

Another possible answer for this case would be:

A -> O

```
m=video <port> RTP/AVP 98..
a=rtpmap:98 3gpp-tt/1000
a=fmtp:98 tx=120; ty=105; layer=0; max-h=95; max-w=150; sver=60
a=recvonly
```

In this case, the answerer does not accept the values offered. The offerer MUST use these values or else remove the stream.

9.4. Parameter Usage outside of Offer/Answer

SDP may also be employed outside of the Offer/Answer context, for instance for multimedia sessions that are announced through the Session Announcement Protocol (SAP) [14] or streamed through the Real Time Streaming Protocol (RTSP) [15].

In this case, the receiver of a session description is required to support the parameters and given values for the streams, or else it MUST reject the session. It is the responsibility of the sender (or creator) of the session descriptions to define the session parameters so that the probability of unsuccessful session setup is minimized. This is out of the scope of this document.

10. IANA Considerations

IANA has registered the media subtype name "3gpp-tt" for the media type "video" as specified in Section 8 of this document.

11. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [3] and any applicable RTP profile, e.g., AVP [17].

In particular, an attacker may invalidate the current set of active sample descriptions at the client by means of repeating a packet with an old sample description, i.e., replay attack. This would mean that the display of the text would be corrupted, if displayed at all. Another form of attack may consist of sending redundant fragments, whose boundaries do not match the exact boundaries of the originals

(as indicated by LEN) or fragments that carry different sample lengths (SLEN). This may cause a decoder to crash.

These types of attack may easily be avoided by using source authentication and integrity protection.

Additionally, peers in a timed text session may desire to retain privacy in their communication, i.e., confidentiality.

This payload format does not provide any mechanisms for achieving these. Confidentiality, integrity protection, and authentication have to be solved by a mechanism external to this payload format, e.g., SRTP [10].

12. References

12.1. Normative References

[1] Transparent end-to-end packet switched streaming service (PSS); Timed Text Format (Release 6), TS 26.245 v 6.0.0, June 2004.

[2] ISO/IEC 14496-12:2004 Information technology - Coding of audio-visual objects - Part 12: ISO base media file format.

[3] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[4] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[5] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[6] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003.

12.2. Informative References

[7] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, December 1999.

[8] Perkins, C. and O. Hodson, "Options for Repair of Streaming Media", RFC 2354, June 1998.

[9] W3C, "Synchronised Multimedia Integration Language (SMIL 2.0)", August, 2001.

[10] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[11] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", Work in Progress, September 2005.

[12] van der Meer, J., Mackie, D., Swaminathan, V., Singer, D., and P. Gentric, "RTP Payload Format for Transport of MPEG-4 Elementary Streams", RFC 3640, November 2003.

[13] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[14] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

[15] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[16] Transparent end-to-end packet switched streaming service (PSS); Protocols and codecs (Release 6), TS 26.234 v 6.1.0, September 2004.

[17] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[18] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[19] Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000.

[20] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[21] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for RTCP-based Feedback (RTP/AVPF)", Work in Progress, August 2004.

[22] Hellstrom, G., "RTP Payload for Text Conversation", RFC 2793, May 2000.

[23] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, June 2005.

[24] ITU-T Recommendation T.140 (1998) - Text conversation protocol for multimedia application, with amendment 1, (2000).

[25] ISO/IEC 10646-1: (1993), Universal Multiple Octet Coded Character Set.

[26] ISO/IEC FCD 14496-17 Information technology - Coding of audio-visual objects - Part 17: Streaming text format, Work in progress, June 2004.

[27] Transparent end-to-end Packet-switched Streaming Service (PSS); 3GPP SMIL language profile, (Release 6), TS 26.246 v 6.0.0, June 2004.

[28] Casner, S. and P. Hoschka, "MIME Type Registration of RTP Payload Formats", RFC 3555, July 2003.

[29] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[30] Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP) (Release 6), TS 26.244 V6.3. March 2005.

[31] Castagno, R. and D. Singer, "MIME Type Registrations for 3rd Generation Partnership Project (3GPP) Multimedia files", RFC 3839, July 2004.

13. Basics of the 3GP File Structure

This section provides a coarse overview of the 3GP file structure, which follows the ISO Base Media file Format [2].

Each 3GP file consists of "Boxes". In general, a 3GP file contains the File Type Box (ftyp), the Movie Box (moov), and the Media Data Box (mdat). The File Type Box identifies the type and properties of the 3GP file itself. The Movie Box and the Media Data Box, serving as containers, include their own boxes for each media. Boxes start with a header, which indicates both size and type (these fields are called, namely, "size" and "type"). Additionally, each box type may include a number of boxes.

In the following, only those boxes are mentioned that are useful for the purposes of this payload format.

The Movie Box (moov) contains one or more Track Boxes (trak), which include information about each track. A Track Box contains, among others, the Track Header Box (tkhd), the Media Header Box (mdhd), and the Media Information Box (minf).

The Track Header Box specifies the characteristics of a single track, where a track is, in this case, the streamed text during a session. Exactly one Track Header Box is present for a track. It contains information about the track, such as the spatial layout (width and height), the video transformation matrix, and the layer number. Since these pieces of information are essential and static (i.e., constant) for the duration of the session, they must be sent prior to the transmission of any text samples.

The Media Header Box contains the "timescale" or number of time units that pass in one second, i.e., cycles per second or Hertz. The Media Information Box includes the Sample Table Box (stbl), which contains all the time and data indexing of the media samples in a track. Using this box, it is possible to locate samples in time and to determine their type, size, container, and offset into that container. Inside the Sample Table Box, we can find the Sample Description Box (stsd, for finding sample descriptions), the Decoding Time to Sample Box (stts, for finding sample duration), the Sample Size Box (stsz), and the Sample to Chunk Box (stsc, for finding the sample description index).

Finally, the Media Data Box contains the media data itself. In timed text tracks, this box contains text samples. Its equivalent to audio and video is audio and video frames, respectively. The text sample consists of the text length, the text string, and one or several Modifier Boxes. The text length is the size of the text in bytes.

The text string is plain text to render. The Modifier Box is information to render in addition to the text, such as color, font, etc.

14. Acknowledgements

The authors would like to thank Dave Singer, Jan van der Meer, Magnus Westerlund, and Colin Perkins for their comments and suggestions about this document.

The authors would also like to thank Markus Gebhard for the free and publicly available JavE ASCII Editor (used for the ASCII drawings in this document) and Henrik Levkowetz for the Idnits web service.

Authors' Addresses

Jose Rey
Panasonic R&D Center Germany GmbH
Monzastr. 4c
D-63225 Langen, Germany

EMail: jose.rey@eu.panasonic.com
Phone: +49-6103-766-134
Fax: +49-6103-766-166

Yoshinori Matsui
Matsushita Electric Industrial Co., LTD.
1006 Kadoma
Kadoma-shi, Osaka, Japan

EMail: matsui.yoshinori@jp.panasonic.com
Phone: +81 6 6900 9689
Fax: +81 6 6900 9699

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).