L2/06-012

2006-01-20

Title: Comments and response regarding use of BOM and UTFs in the MPEG Streaming text format standard, for information to the UTC.

From: V.S. Umamaheswaran

Ref: ISO_IEC_FDIS_14496-17__E_.pdf: Coding of audio-visual objects -- Part 17: Streaming text format (Attached)

Comment accompanying the Canadian ballot on the above document after a discussion initiated by Uma in the Canadian committee for CAC JTC1:

'The Canadian National Body feels that section 7.4.2 does not clearly specify whether a Byte Order Mark is required in UTF text. This is an issue when a receiver decodes a UTF-16 bit stream. Without the Byte Order Mark, there is no way to determine whether the text has been encoded as UTF-16 BE or UTF-16 LE. Although it may be implicit that the marker must be present. The Canadian National Body suggests adding the following text in section 7.4.2. "The Byte Order Marker shall be present in all UTF encoded text" will make the standard much clearer.'

The Canadian National Body notes that when a receiver is only required to decode UTF-16 BE text, but UTF-16 LE text is allowed to be placed in the bitstream, that important text could be omitted during the decoding process. We feel that this means that to allow for interoperability, the encoder in a practical sense will only be able to encode in Big Endian. The Canadian National Body asks to have UTF-16 LE decoding be removed as optional and added as a requirement.. The Canadian National Body notes the following editorial issue. Section 7.4.2 The last sentence in the first paragraph should read: "3GPP text stream receivers shall be able to decode UTF-16 encoded text strings in big endian order.

Response received from the editor Jan vanderMeer [jan.vandermeer@philips.com] after separately communicating the above as a liaison from SC2 to SC29 secretariat and editor by Uma through Mike Ksar.

The UTF_16_flag semantics and note should read:

"UTF_16_flag - a one bit flag, indicating in TTU[1] and TTU[2] the encoding used for the characters in the text string; when set to â€˜1â€™, UTF-16 is applied, when set to '0', UTF-8 is applied. For TTU[3], TTU[4] and TTU[5] this bit has no meaning and shall be ignored. TF-16 encoded text strings shall be serialized in big endian order, also known as network byte order. 3GPP text stream receivers shall be able to decode UTF-8 encoded text strings as well as UTF-16 encoded text strings in big endian order. Support for little endian serialization in receivers is optional.

Note: 3GPP TS 26.245 requires a Byte Order Mark (BOM) by the beginning of each UTF-16 encoded text string within a text sample. With respect to receivers, 3GPP 26.245 mandates support of UTF-8 and UTF-16 big endian, while support of UTF-16 little endian is optional. "

Attachment:

FINAL DRAFT

INTERNATIONAL STANDARD

ISO/IEC FDIS 14496-17

ISO/IEC JTC **1**

Secretariat: **ANSI**

Voting begins on:
2005-12-08

Voting terminates on:
2006-02-08

Information technology — Coding of audio-visual objects —

Part 17:
Streaming text format

Technologies de l'information — Codage des objets audiovisuels —

Partie 17: Format de texte en flux

Please see the administrative notes on page iii

Reference number
ISO/IEC FDIS 14496-17:2005(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-17 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

— *Part 7: Optimized reference software for coding of audio-visual objects* [Technical Report]

— *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*

— *Part 9: Reference hardware description* [Technical Report]

— *Part 10: Advanced Video Coding (AVC)*

— *Part 11: Scene description and application engine*

— *Part 12: ISO base media file format*

— *Part 13: Intellectual Property Management and Protection (IPMP) extensions*

— *Part 14: MP4 file format*

— *Part 15: Advanced Video Coding (AVC) file format*

— *Part 16: Animation Framework eXtension (AFX)*

— *Part 17: Streaming text format*

— *Part 18: Font compression and streaming*

— *Part 19: Synthesized texture stream*

— *Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*

— *Part 21: MPEG-J GFX*

The following parts are under preparation:

— *Part 22: Open Font Format*

Introduction

This International Standard was developed in response to the need for a generic method for coding of text at very low bitrate as one of the multimedia components within audiovisual presentations. This International Standard allows for example subtitles and Karaoke song texts to be coded and transported as separate text streams for presentation jointly with other components of an audiovisual presentation at bitrates that are sufficently low for use in mobile services over IP.

Information technology — Coding of audio-visual objects —

Part 17: **Streaming text format**

1 Scope

This International Standard specifies the coded representation of textual information for timed presentation on screens. The text may be streamed in association with video and audio, in which case the text may represent subtitles e.g. with translations of the associated audio in another language, or as an aid to the hard of hearing; another example is the text of a song in a Karaoke application. However, the text may also be streamed as a stand-alone application without any associated video and audio. The streaming text format is specified in a transport agnostic manner, so as to allow transport over a large variety of transport means, while providing a reasonable level of random access and error robustness.

The text streams are defined as byte streams that are capable of carrying text access units of a specified format, optionally interleaved with data needed for the decoding of the text stream. The format of text streams and text access units is specified, as well as signaling and decoding of text streams.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14496-18:2004, *Information technology — Coding of audio-visual objects — Part 18: Font compression and streaming*

3GPP TS 26.245: 2003, Timed text format (Release 6)

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
text stream
byte stream capable of carrying text access units of a specified format, optionally interleaved with data needed for the decoding of the text stream

3.2
text access unit
individually accessible portion of data within a text stream

NOTE Each text access unit contains the coded representation of text data. For presentation, the text access unit can be associated with a single time stamp.

3.3
3GPP text stream
text stream carrying 3GPP text access units

3.4
3GPP text access unit
text access unit carrying data from a text sample specified by 3GPP

3.5
text sample
when used in the context of a 3GPP text stream, a text sample, as specified in 3GPP TS 26.245, consisting of a text string, optionally followed by one or more text modifiers

3.6
text string
when used in the context of a 3GPP text stream, data within a text sample, representing a string of characters encoded using UTF-8 or UTF-16, as specified in 3GPP TS 26.245

3.7
text modifier
when used in the context of a 3GPP text stream, data within a text sample, specifying a modification to the presentation of the text string within that text sample, as specified in 3GPP TS 26.245

3.8
sample description
when used in the context of a 3GPP text stream, descriptive text data, providing global information about one or more text samples, such as font(s) to be used and positioning of the text, as specified in 3GPP TS 26.245

3.9
Timed Text Unit
TTU
syntactical structure within a 3GPP text stream for carriage of text access units, whereby its index j identifies which type of data (such as a complete text access unit, a fragment thereof or a sample description) is carried

4 Text stream format

A text stream is a byte stream that is capable of carrying text access units of a specified format, optionally interleaved with data needed for the decoding of the text stream. The text stream format is defined so as to conveniently allow for carriage in transport packets and files. However, such carriage itself depends on specific formats for file storage and transport and is beyond the scope of this standard.

5 Text access units

5.1 Timing and decoding of text access units

Each text access unit contains text data of a specified format; to each text access unit a single time stamp applies. The time stamp assigned to a text access unit indicates the time at which the text access unit is to be presented on the display. To decode a text stream, a receiver needs information on the text stream, as defined by the so-called decoder configuration in TextConfig. The TextConfig signals the format of the text data and may provide information specific to the format of the text data.

5.2 Format of text access units

```
text access unit{
   textData;             // the format of the textData is textFormat specific
   }
```

5.3 TextConfig

```
TextConfig(){
   bit(8)  textFormat;
   bit(16) textConfigLength;
   formatSpecificTextConfig();
   }
```

5.4 Semantics

`textFormat` – one byte signaling the format of the text data. The value 0x01 signals that the text data carries timed text data as defined in 3GPP TS 26.245, in a manner defined in clause 7.

Table 1 — textFormat

0x00	Reserved
0x01	Timed Text as specified in 3GPP TS 26.245
0x02 – 0xEF	Reserved
0xF0 – 0xFE	User-private
0xFF	Reserved

`textConfigLength` – unsigned integer that specifies the size in number of bytes of `formatSpecificTextConfig()`.

6 Usage of a text stream within an MPEG-4 system context

6.1 Signaling of a text stream

When used in an MPEG-4 system context, a text stream shall be signaled by a streamType value 0x0D and by an objectTypeIndication value of 0x08.

6.2 Usage in the scene description

When used within an MPEG-4 Scene description, a text stream object is used as follows:

- If the text stream object is used by an AnimationStream node, it shall be presented in the scene, regardless of whether the AnimationStream node is active or not;
- If the text stream object is not used by any AnimationStream node, it shall not be presented in the scene.

Spatial presentation of the text data is specified by the underlying streaming text format. In case the scene description has display size indication, results are undefined if the positioning of text data covers an out-of-display area.

All rules regarding time control and segmentation of stream objects apply to the text stream object.

7 Text data format for 3GPP text streams

7.1 Introduction

This clause specifies the text data format for 3GPP text streams.

7.2 Carriage of text samples and sample descriptions in 3GPP text access units

3GPP TS 26.245 defines timed text data to consist of text samples and sample descriptions, and that each text sample consists of one text string, optionally followed by one or more text modifiers. Each text string represents the characters that form the text to be displayed, while the text modifiers carry the changes that are to be applied to the text string during the time that the text is to be displayed within a text box, such as text colour changes synchronized with a song for a Karaoke application.

A sample description provides global information about a text sample, for example about font(s) to be used, about the positioning of the text within the text box, the background colour of that text box, etc. Multiple sample descriptions are allowed; to each sample description (SD) an index is assigned and to each text sample the index of the applicable sample description is associated. While a sample description will typically apply to multiple text samples, to each text sample exactly one sample description applies.

The relationship between sample descriptions, text samples, a text string and text modifiers is depicted in Figure 1.

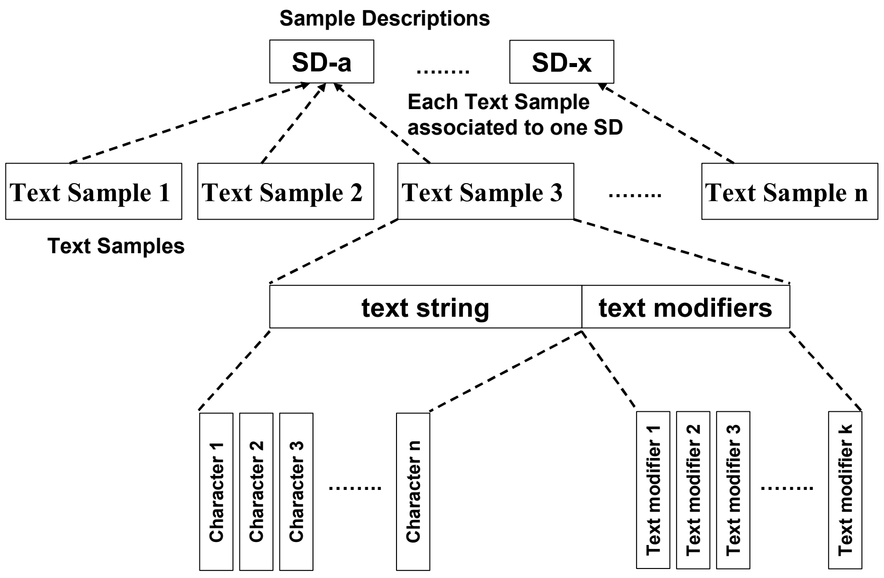

Figure 1 — Sample Descriptions, text samples, text strings and text modifiers in 3GPP text streams

A 3GPP text access unit contains data from exactly one text sample. Consequently, a 3GPP text access unit shall not contain data from more than one text sample. Each 3GPP text access unit is to be presented during a certain period of time, specified by duration information. In addition to a text sample, a 3GPP text access unit may contain zero or more complete sample descriptions. By allowing interleaving of text samples with sample descriptions in 3GPP text access units, 3GPP text streams are capable of carrying sample descriptions in-band1).

7.3 Transport of 3GPP text access units in TTUs

Typically, a 3GPP text access unit is small, around 100 – 200 bytes, and often much smaller than the size of the packets that carry the text data across a transport network. It is therefore expected that transport systems will often aggregate multiple 3GPP text access units into one transport packet. On the other hand, 3GPP text

1) Note however that 3GPP sample descriptions may also be provided out-of-band.

access units can also be large, for example when scrolling horizontal text at the bottom of the screen, in which case fragmentation of 3GPP text access units may be required prior to transport. In conclusion, transport of 3GPP text access units often requires aggregation and sometimes fragmentation. So as to conveniently aggregate and fragment 3GPP text access units in a transport independent manner, this Specification defines a flexible framing structure consisting of so-called TTUs, Timed Text Units.

Five different types of TTUs are defined; one for carriage of a complete 3GPP text access unit, three for carriage of text sample fragments, and one for carriage of a complete sample description, while three types are reserved for future use. Because sample descriptions are small, there is no support for carriage of sample description fragments.

The flexible framing structure provided by TTUs allows for easy and convenient adaptation to the various transport layers, while performing TTU alignment with the applied transport packets. For each transport layer the most suitable TTU structure can be chosen. For example, by using TTUs, small text samples can be aggregated into one transport packet, but TTUs can also be used to fragment text samples across multiple transport packets, while providing a reasonable level of error resilience in case of packet loss or non-recoverable packet errors. If so desired, the text data within a text access units can be re-partitioned into TTUs for most effective adaptation to other transport systems. See Figure 2. TTUs are defined for 3GPP text streams only, but comparable structures may be defined for other text streams in future versions of this Specification.

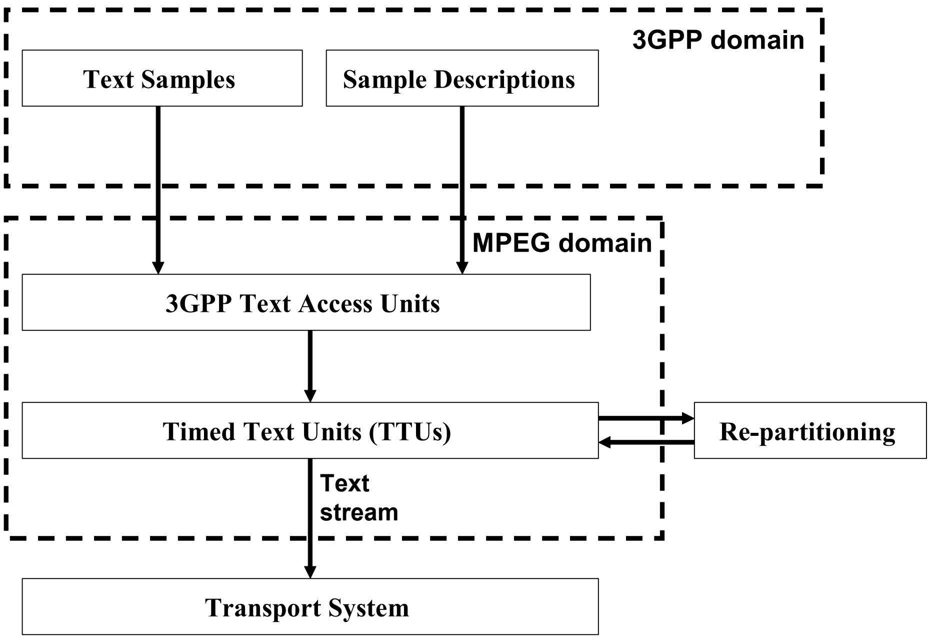

Figure 2 — Carriage of text samples and sample descriptions in 3GPP text access units and the use of TTUs for creating a 3GPP text stream

The format of TTUs is defined in this Specification, as well as some general requirements for their use, while the actual usage of TTUs is transport specific and beyond the scope of this Specification. However, as TTUs support text access unit aggregation and fragmentation in a generic and error resilient manner, transport systems are strongly recommended to use TTUs for aggregation and fragmentation instead of transport specific tools.

Each text access unit with a 3GPP text stream is defined to consist of one or more TTUs. Each TTU type is identified by its index j and referred to as TTU[j]. The following TTU types are defined:

- TTU[0]; this TTU type is reserved for future use;
- TTU[1]; this TTU type is capable of carrying one complete 3GPP text access unit;
- TTU[2]; this TTU type is capable of carrying one fragment of a text string from a text sample;
- TTU[3]; this TTU type is capable of carrying the first fragment of a text modifier from a text sample;
- TTU[4]; this TTU type is capable of carrying a non-first fragment of a text modifier from a text sample;
- TTU[5]; this TTU type is capable of carrying one complete sample description;
- TTU[6] and TTU[7]; these TTU types are reserved for future use.

Figure 3 depicts the options to construct TTUs from text samples and sample descriptions. Each complete sample description is contained in one TTU[5]. Each text sample that is not fragmented, is contained in one TTU[1]. When a text sample is fragmented, then the text string is separated from the text modifiers for carriage in different TTUs. Each text string fragment is contained in one TTU[2], so that there are as many TTU[2]s as text string fragments. Note however that the text string itself may not be fragmented, in which case there is only a single text string fragment carried in one TTU[2]. The first text modifier fragment is contained in one TTU[3]. Hence, TTU[3] signals the start of the modifiers. Each subsequent text modifier fragment, if any, is contained in one TTU[4].

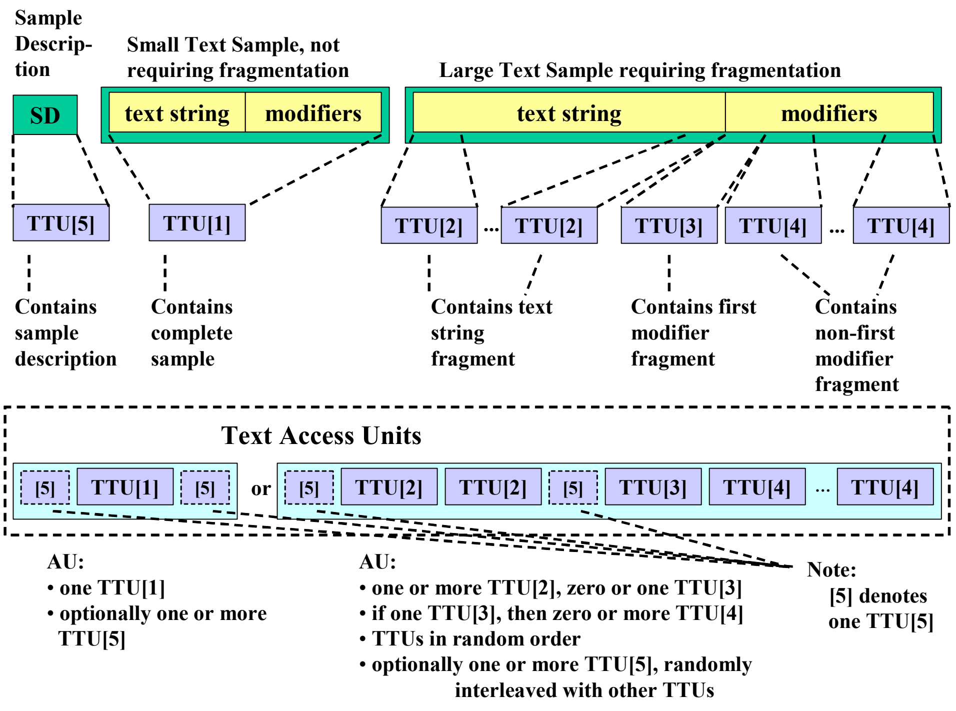
Sample Descrip- tion
Small Text Sample, not requiring fragmentation
Large Text Sample requiring fragmentation
SD
text string
modifiers
text string
modifiers
TTU[5]
TTU[1]
TTU[2]
...
TTU[2]
TTU[3]
TTU[4]
...
TTU[4]
Contains sample description
Contains complete sample
Contains text string fragment
Contains first modifier fragment
Contains non-first modifier fragment
Text Access Units
[5]
TTU[1]
[5]
or
[5]
TTU[2]
TTU[2]
[5]
TTU[3]
TTU[4]
...
TTU[4]
AU:
• one TTU[1]
• optionally one or more TTU[5]
AU:
• one or more TTU[2], zero or one TTU[3]
• if one TTU[3], then zero or more TTU[4]
• TTUs in random order
• optionally one or more TTU[5], randomly interleaved with other TTUs
Note:
[5] denotes one TTU[5]

Figure 3 — Carriage of Sample Descriptions and Text Samples in TTU[j]s, and construction of Text Access Units by means of TTUs

In Figure 3, also the basics of the construction of text access units by means of TTUs are depicted. If a text sample is not fragmented, then a text access unit contains the TTU[1] that carries the entire text sample. If the text sample is fragmented, then the text access unit contains one or more TTU[2]s, each with a fragment of the text string. If the text sample also carries modifiers, then the text access unit carries one TTU[3] and, depending on the size of the modifiers, zero or more TTU[4]s. In addition, each text access unit contains optionally one or more TTU[5]s, each carrying a sample description that may apply to this or future text samples in the text stream.

An empty text sample, that is a text sample without data (hence without a text string and modifiers), may be used to terminate the display of a text string prior to the start of the display of the next text sample in the stream. Such empty text sample shall be carried in a TTU[1] as a separate access unit.

The order of TTUs within a text access unit is random within some constraints. When text samples are fragmented, it is allowed to repeat a fragment that is considered more important than others, and to transmit modifiers prior to text string fragments, which may be useful for large text strings. TTU[5]s may be interleaved randomly with other TTUs. For ordering TTUs within a text access unit, only the following constraint applies:

- When a sample description is provided in-band, then a sample description shall be included in the stream prior to any text sample that utilizes that sample description. This means that an in-band sample description, carried in a TTU[5], shall precede any TTU[1], TTU[2], TTU[3] and TTU[4] that references that sample description. Note that such TTU[5] may be carried in the same text access unit, or in one or more preceding text access unit(s).

7.3.1 Constraints for fragmentation

A number of constraints shall be applied when fragmenting text samples for carriage in TTU[2], TTU[3] and TTU[4]. The following constraints shall be applied:

- Each text string shall be fragmented on character boundary, so that each fragment contains complete character codes only, so as to ensure that each received text fragment can be decoded even in the case that a previous fragment gets lost. As a consequence, text string fragmentation requires knowledge of UTF-8/-16 formats to determine character boundaries.

It is not required to fragment text modifiers on modifier boundary; consequently, when a modifier fragment is lost, receivers will typically not be capable of decoding any remaining fragments.

7.3.2 TTU structure

Each TTU starts with an 8 bit TTU header; this header carries the type of TTU, signals whether characters in the text string of the text sample are encoded using UTF-8 or UTF-16, and signals the length (size) of the TTU in bytes. The remaining fields after the TTU header are TTU type specific and described in the following subclauses. Note that in addition to the transport agnostic requirements defined in this Specification, transport specific requirements not defined in this Specification may apply.

7.3.2.1 TTU[1] signaling

A TTU[1] carries a complete text sample; the TTU header and the TTU data length field are followed by fields that provide the following information:

- the index of the applicable sample description;

- the duration of the text sample, and

- the length in bytes of the text string in the text sample. Note that the length of the text string in bytes does not give the character count, as a character can vary in length from, in general, 1 to 6 bytes.

The length of the text modifiers is not specified, as it is assumed that this information is derived from the total text sample length also provided by the TTU header.

Carriage of an empty text sample in a TTU[1] is signaled by the presence of the TTU data length field encoded with a specific value (see subclause 7.4.2).

NOTE 1 When the transported text string was stored in a 3GP file, it is to be ensured that the values of the two fields that precede the text string in a 3GP file, the 16 bit string length field and the byte order mark are correctly encoded in the equivalent fields of the TTU header.

NOTE 2 If a timed text stream is to be stored in a 3GP file, then the byte order mark and the text string length need to be recovered from the TTU headers in order to correctly store the stream in the 3GP file.

7.3.2.2 TTU[2] signaling

A TTU[2] carries a text string fragment from a text sample. The TTU header and the TTU data length field are followed by fields that signal:

- the total number of text sample fragments; this number shall include each text string fragment and each modifier fragment.
- the sequence number of the fragment contained in this TTU[2]; the sequence number of the first text sample fragment is equal to 0, and for each subsequent fragment the sequence number is incremented by one, so that in case of N fragments, the sequence number of the last fragment is equal to N-1. A text sample can be fragmented in at most 16 different fragments;
- the duration of the text sample;
- the index of the applicable sample description;
- the total length of the original non-fragmented text sample; this information allows the receiver to allocate buffer space for the decoding of the text sample; this information is provided for each text string fragment so as to also allow for buffer space allocation after losing one or more text string fragments.

7.3.2.3 TTU[3] and TTU[4] signaling

In a TTU[3] and TTU[4], the first and a non-first modifiers fragment is carried, respectively; the TTU header and the TTU data length field are followed by fields that signal:

- the total number of text sample fragments (see subclause 7.3.2.2);
- the sequence number of the fragment contained in this TTU[2] (see subclause 7.3.2.2);
- the duration of the text sample.

Note that the index of the applicable sample description, the duration of the text sample, and the total length of the fragmented text sample are not signaled, as modifiers fragments apply to text strings for which this information is already signaled.

7.3.2.4 TTU[5] signaling

A TTU[5] carries a sample description. In TTU[5], the TTU header and the TTU data length field are followed by a field that signals:

- the index of the applicable sample description.

7.3.3 Constraints for using Sample Descriptions

Within text streams 256 index values are available to unambiguously identify a sample description. The value 255 is reserved. Sample descriptions can be provided either in-band or out-of-band. The index values in the inclusive range between 0 and 127 are available for in-band usage, while values between 128 and 254 (inclusive) can be used to provide sample descriptions out-of-band. See Table 2.

Table 2 — Index values for Sample Descriptions

Index value	Assignment
0 – 127	Available for in-band usage
128 – 254	Available for out-of- band usage
255	Reserved

When sample descriptions are provided in-band, there may be a need to regularly update sample descriptions. For this purpose a wrap-up mechanism is defined for in-band sample descriptions, that allows to continuously update such sample descriptions from a range of 64 valid index values. Hence, at any point in time at most 64 in-band sample descriptions can be in use simultaneously. Each sample description with an index value not in the valid range is discarded from further use. Valid sample descriptions can be transmitted in-band as often as desirable for random access, without any impact on the range of invalid index values. However, when receiving a sample description with an index value from the invalid range, then the valid range is changed, so that the received sample description becomes valid. When a sample description is received with index value i not in the valid range, then the receiver shall discard each sample description with an index value in the range between (i+1)mod128 and (i+64)mod128, inclusive. For example, assume that the valid range is between 41 and 104 inclusive, and hence, that the values 105 to 127 and 1 to 40 are invalid values. When now the sample description with index value 114 is received, the receiver shall invalidate the index values 115 to 127 and 0 to 50, inclusive, so as to change the valid range to values between 51 and 114.

7.4 TTU Syntax and Semantics

7.4.1 TextData()

TextData() consists of a concatenation of TTUs that carry data from Text Access Units. The TTUs shall carry Text Access Units in display order. TTUs with data belonging to one Text Access Unit shall not be interleaved with TTUs with data belonging to another Text Access Unit.

```
textData(){
      for (i=0; i<N;i++){
            TTU[j]()
            }
      }
```

7.4.2 TTU[j]()

```
TTU[j](){
      UTF_16_flag                                 1     bslbf
      reserved                                    4     bslbf
      TTU_type                                    3     uimsbf
      TTU_data_length                             16    uimsbf
      TTU_data[j]()
      }
```

`UTF_16_flag` - a one bit flag, indicating in TTU[1] and TTU[2] the encoding used for the characters in the text string; when set to ‘1’, UTF-16 is applied, when set to ‘0’, UTF-8 is applied. For TTU[3], TTU[4] and TTU[5] this bit has no meaning and shall be ignored. TF-16 encoded text strings shall be serialized in big endian order, also known as network byte order. 3GPP text stream receivers shall be able to decode TF-16 encoded text strings in big endian order. Support for little endian serialization is optional.

NOTE 3GPP TS 26.245 requires only big endian serialization support in receivers.

`TTU_type` – a three bit integer that specifies the type of TTU; the TTU-type field is encoded with a value that is equal to the index j in TTU[j], according to the following Table.

Table 3 — TTU types

TTU_type value	TTU type	TTU usage
0	TTU[0]	Reserved for future use
1	TTU[1]	Carriage of a complete 3GPP text access unit
2	TTU[2]	Carriage of a text string fragment
3	TTU[3]	Carriage of a first fragment of a text modifier
4	TTU[4]	Carriage of non-first fragment of a text modifier
5	TTU[5]	Carriage of a complete sample description
6	TTU[6]	Reserved for future use
7	TTU[7]	Reserved for future use

`TTU_data_length` – a 16 bit unsigned integer that specifies the total number of bytes of the TTU_data_length field and the immediately following TTU_data[j](). In a TTU[1], carriage of an empty text sample (for empty text samples see subclause 7.3) is signaled by a TTU_data_length value of 8.

7.4.3 TTU_data[0]()

```
TTU_data[0](){
      for (i=0; i<N;i++){
            reserved                              8     bslbf
            }
      }
```

7.4.4 TTU_data[1]()

```
TTU_data[1](){
      sample_index                                8     uimsbf
      sample_duration                             24    uimsbf
      text_string_length                          16    uimsbf
      whole_text_sample()
      }
```

`sample_index` – an 8 bit unsigned integer that specifies the index of the sample description that applies to the text sample contained in this TTU. The sample_index value shall unambiguously refer to one particular sample description that shall be present in the text stream prior to its use. The sample_index values 0 and 255 are reserved.

`sample_duration` – a 24 bit unsigned integer that specifies the intended duration of the presentation of the text sample in this TTU in units of the durationClock, specified in textConfig; see subclause 7.6. A sample_duration value of zero signals an unknown duration. A text sample with a sample_duration equal to zero is to be presented until the subsequent text sample is to be presented. A text sample with a sample_duration equal to zero shall not be the last text sample of a 3GPP text stream. When appropriate, a text sample with a sample_duration equal to zero may be followed by an empty text sample; see subclauses 7.3 and 7.4.2 for using of and constraints for empty samples.

NOTE if a 3GPP text stream is to be stored in a 3GP file, then for each text sample with a sample_duration equal to zero, the duration value MUST be changed from zero to the effective duration of the text sample, as the ISO file format explicitly forbids a sample duration of zero. The effective duration is the timestamp difference between the current sample with unknown duration and the subsequent text sample in the stream.

`text_string_length` – a 16 bit unsigned integer that specifies the number of bytes of the text string in this TTU.

`whole_text_sample()` – TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see subclauses 5.3, 5.4 and 7.6).

7.4.5 TTU_data[2]()

```
TTU_data[2](){
      total_number_of_sample_fragments          4     uimsbf
      number_of_this_sample_fragment            4     uimsbf
      sample_duration                           24    uimsbf
      sample_index                              8     uimsbf
      sample_length                             16    uimsbf
      text_string_fragment()
      }
```

`total_number_of_sample_fragments` – a 4 bit unsigned integer that specifies for the TextSample of which a fragment is carried in this TTU the total number of fragments in which that TextSample is split.

`number_of_this_sample_fragment` – a 4 bit unsigned integer that specifies the sequence number of the TextSample fragment that is carried in this TTU. When the TextSample is split into N fragments, then the sequence number reflects the logical order of the fragments, that is the first fragment has sequence number 0, and the last fragment sequence number N-1, while for each subsequent fragment the sequence number is increased by 1.

`sample_duration` – See `sample_duration` semantics in clause 7.4.4

`sample_index` – See `sample_index` semantics in clause 7.4.4

`sample_length` – a 16 bit unsigned integer that specifies the total number of bytes in the TextSample of which a fragment is carried in this TTU.

`text_string_fragment()` – a fragment of the text string from the TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see subclauses 5.3, 5.4 and 7.6).

7.4.6 TTU_data[3]()

```
TTU_data[3](){
      total_number_of_sample_fragments          4     uimsbf
      number_of_this_sample_fragment            4     uimsbf
      sample_duration                           24    uimsbf
      first_modifier_box_fragment()
      }
```

`total_number_of_sample_fragments` – See `total_number_of_sample_fragments` semantics in subclause 7.4.5.

`number_of_this_sample_fragment` – See `number_of_this_sample_fragment` semantics in clause 7.3.5.

`sample_duration` – See `sample_duration` semantics in clause 7.3.4

`first_modifier_box_fragment()` – the first fragment of the text modifiers from the TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see subclauses 5.3, 5.4 and 7.6).

7.4.7 TTU_data[4]()

```
TTU_data[4](){
      total_number_of_sample_fragments          4     uimsbf
      number_of_this_sample_fragment            4     uimsbf
      sample_duration                           24    uimsbf
      non_first_modifier_box_fragment()
      }
```

total_number_of_sample_fragments – See total_number_of_sample_fragments semantics in clause 7.4.5.

number_of_this_sample_fragment – See number_of_this_sample_fragment semantics in clause 7.4.5.

sample_duration – See sample_duration semantics in clause 7.3.4

non_first_modifier_box_fragment() – a non-first fragment of the text modifiers from the TextSample, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in the textConfig (see Subclauses 5.3, 5.4 and 7.6).

7.4.8 TTU_data[5]()

```
TTU_data[5](){
      sample_index                              8     uimsbf
      sample_description()
      }
```

sample_index – the index of the sample description that is contained in this TTU. The sample_index value shall unambiguously refer to one particular sample description that shall be present in the text stream prior to its use. The sample_index values 0 and 255 are reserved. The sample_index values between 128 and 254 inclusive are assigned to out-of-band provided sample descriptions and shall not be used for in-band provided sample descriptions. Similarly, the sample_index values between 1 and 127 inclusive are assigned to in-band provided sample descriptions and shall not be used for out-of-band provided sample descriptions.

The following shall apply to in-band provided sample descriptions. At any instant in time, at most 64 different sample_index values shall be in use for in-band sample descriptions, identified by a range of valid sample_index values. When a receiver starts the decoding of a text stream, the receiver has no knowledge of the valid range, but upon receiving the first sample description the receiver shall define the range of valid sample_index values as follows. If the first received sample has the sample_index value j, then the receiver shall define the sample_index values in the range between (j+1)mod128 and (j+64)mod128, inclusive, invalid and shall consider all other sample_index values between 1 and 127 (inclusive) valid. When subsequent sample descriptions are received with a sample_index value in the valid range, then the valid range shall not be modified. However, when a sample description is received with index value i not in the valid range, then the receiver shall invalidate sample_index values in the range between (i+1)mod128 and (i+64)mod128, inclusive. Each sample description with an sample_index value assigned as invalid shall be discarded. See also Table 4. Text samples in the text stream shall not reference any invalidated sample description. A sample description is valid for each subsequent text sample in the text stream until invalidated by another sample description.

Table 4 — Sample Descriptions that are invalidated by the Sample Description with sample_index value i

Received sample_index value i in currently invalid range	New invalid range (inclusive)	New valid range (inclusive)
1 ≤ i ≤ 63	(i+1) – (i+64)	1 – i (i+65) – (127)
64 ≤ i ≤ 127	(i+1) – 127 1 – (i-64)	(i-63) – i

sample_description() – a sample description for a TextSample, referred to as TextDescription, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see subclauses 5.3, 5.4 and 7.6).

7.4.9 TTU_data[6]()

```
TTU_data[6]{
      for (i=0; i<N;i++){
            reserved                      8     bslbf
            }
      }
```

7.4.10 TTU_data[7]()

```
TTU_data[7]{
      for (i=0; i<N;i++){
            reserved                      8     bslbf
            }
      }
```

7.5 Positioning of 3GPP text streams

Text from a decoded 3GPP text stream is rendered within a Text Box. The Text Box can be positioned within a Text Track; see 3GPP TS 26.245. How to render the text in the Text Box and how to position the Text Box within the Text Track is specified within the text stream by means of sample descriptions and/or text modifiers. The width and the height of the Text Track shall be defined in units of square pixels in the formatSpecificTextConfig(), specified in 7.6.

The Text Track can be positioned relative to a scene. The scene may be defined explicitly by means beyond the scope of this Specification; the scene may consist of only a video picture; in that case, the size of the scene corresponds to the size of the video picture, while the scene origin corresponds to the most top left pixel of the video picture. TextConfig specifies whether or not the Text Track is intended to be positioned relative to a scene. Note however that a Text Track can also be positioned with respect to a scene or a video picture by means of information that is provided by external means, such as information contained in an MP4 file or by an application. External information, if provided, takes priority over information provided within textConfig.

If the Text Track is positioned relative to a scene, then textConfig specifies the scene-width and scene-height of that scene. The scene-width and scene-height, as well as the position of the Text Track relative to the scene origin are specified in units of the same square pixels as the width and the height of the Text Track, so as to enable scaling and positioning of the text with respect to the scene in a manner independent of the scene resolution. This allows the use of same text stream with scenes of different resolutions. For example, for a scene that only consist of a video picture, the scene size and the relative position of the Text Track within the scene will typically have a different dimension than the actually coded video pixels. Hence, if the scene-width and scene-height specify a picture with a size of 640*480, then the video picture may actually be coded at 720*480, 540*480, 360*240 or any other resolution.

The relative position of the Text Track with respect to the scene origin is specified by horizontal-offset and vertical-offset, specifying the horizontal and vertical distance in pixels between the scene origin and the top left pixel of the Text Box, respectively; see Figure 4 for an example where the scene consists of only a video picture. A positive value of horizontal-offset and vertical-offset indicate that the top left pixel of the Text Track is right and below, respectively, of the scene origin. For example, in Figure 4, both horizontal-offset and vertical-offset have a positive value. Accordingly, a negative value of horizontal-offset and vertical-offset indicates that the top left pixel of the Text Box is left and above, respectively, of the scene origin or the top left pixel of the video picture. It is allowed that the Text Box is positioned partly or completely outside of the scene.

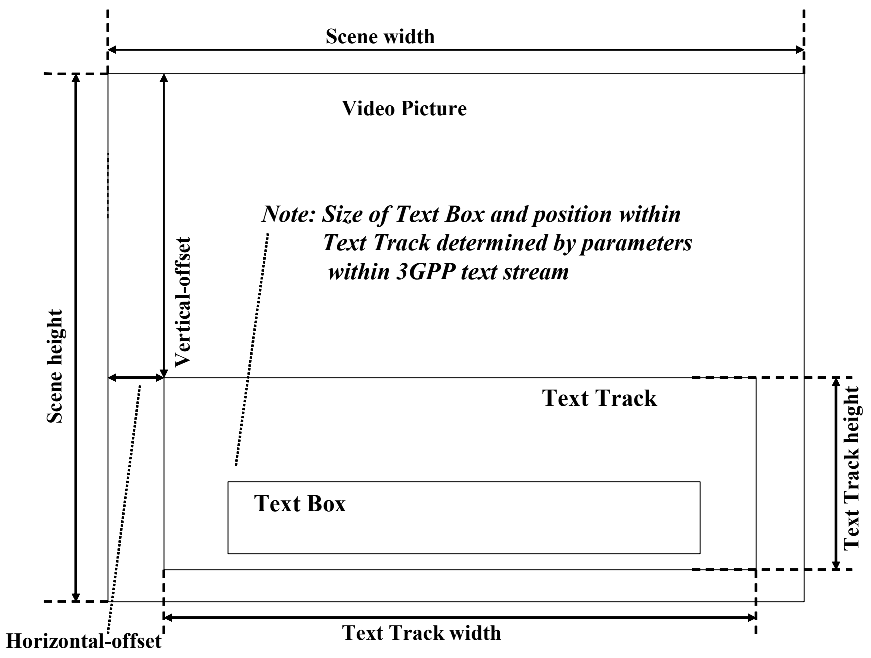

Figure 4 — Size of the Text Track and the positioning of the Text Track with respect to a video picture in the absence of an explicit scene definition

7.6 Format specific decoder configuration for 3GPP text streams

To decode a 3GPP text stream requires that the decoder has access to the format specific decoder configuration for the 3GPP text stream, as defined by formatSpecificTextConfig. The formatSpecificTextConfig provides information such as:

- the 3GPP format that the text stream is compatible with;
- the level and profile of the 3GPP text stream;
- clock units used to encode text sample durations;
- whether in-band sample descriptions are applied;
- whether out-of-band sample descriptions are applied;
- whether sample descriptions are carried in the formatSpecificTextConfig;
- optional information on the positioning of the text track relative to a scene origin or a video picture.

7.6.1 Syntax

```
formatSpecificTextConfig(){
  bit(8) 3GPPBaseFormat;
  bit(8) profileLevel;
  bit(24)durationClock;
  bit(1) contains-list-of-compatible-3GPPFormats-flag;
  bit(2) sampleDescriptionFlags;
  bit(1) SampleDescription-carriage-flag;
  bit(1) positioning-information-flag;
  bit(3) reserved;
  bit(8) layer;
  bit(16) text-track-width;
  bit(16) text-track-height;
  if (contains-list-of-compatible-3GPPFormats-flag) {
     bit(8) number-of-formats;
     bit(8) Compatible-3GPPFormat(number-of-formats);
     }
  if (SampleDescription-carriage-flag) {
     bit(8) number-of-SampleDescriptions;
     Sample_index_and_description()(number-of-SampleDescriptions);
                               //as many Sample_index_and_description()
                               //as number-of-SampleDescriptions
    }
  if (positioning-information-flag) {
     bit(16) scene-width;
     bit(16) scene-height;
     bit(16) horizontal-scene-offset;
     bit(16) vertical-scene-offset;
     }
  }

Sample_index_and_description(){
  bit(8) sample_index;
  SampleDescription();
  }
```

7.6.2 Semantics

`3GPPBaseFormat` – one byte signaling the latest 3GPP format that this text stream is compatible with, coded as specified in Table 5. If the text stream is also compatible with older versions of 3GPP specifications, then these formats can be listed in the textConfig by setting the `contains-list-of-compatible-3GPPFormats-flag`.

Table 5 — 3GPPBaseFormat

0x00 – 0x0F	Reserved
0x10	Timed Text as specified in 3GPP TS 26.245
0x11 – 0xFF	Reserved

`profileLevel` – an 8 bit unsigned integer that specifies the profile and level of the text stream encoded as specified in Table 6. See subclause 7.7 for profile and level constraints that apply to 3GPP text streams.

Table 6 — profileLevel

0x00 – 0x0F	Reserved
0x10	Base profile, base level
0x11 – 0xFF	Reserved

`durationClock` – a 24 bit unsigned integer that specifies the frequency in Hertz of the clock used to encode the `sample-duration` field in TTU[1] and TTU[2]. A value equal to 1000 signals a 1kHz clock, and that `sample-duration` field is encoded in units of 1 msec. Transport systems should assign time stamps to text access units using the same frequency as signaled by `durationClock` or an integer multiple thereof.

`contains-list-of-compatible-3GPPFormats-flag` – one bit that signals the presence of a list of one or more 3GPP formats older than the signaled format by `3GPPBaseFormat`, that the text stream is also compatible with.

`sampleDescriptionFlags` – two bits that signal whether sample descriptions are provided in-band, out-of-band or both, as specified in Table 7. Sample descriptions provided within formatSpecificTextConfig are considered to be provided out-of-band. Note that sample descriptions may also be provided by out-of-band means beyond the scope of this Specification.

Table 7 — sampleDescriptionFlags

00	Forbidden
01	No in-band, out-of band only
10	In-band only, no out-of band
11	Both in-band and out-of band

`sampleDescription-carriage-flag` – one bit that signals the presence of a list of sample descriptions in this formatSpecificTextConfig. This bit shall be set to ‘0’, if `sampleDescriptionFlags` is coded with the value ‘10’.

`positioning-information-flag` – one bit that signals the presence of the `scene-width`, `scene-height`, `horizontal-scene-offset`, and `vertical-scene-offset` fields in this formatSpecificTextConfig.

`layer` – unsigned integer specifying the proximity of the text track to the viewer. A higher value means closer to the user. A value of ‘0’ indicates an undefined proximity. If the proximity is also provided by external means, then the externally provided information takes priority.

`text-track-width` – unsigned integer specifying the width of the text track in square pixels.

`text-track-height` – unsigned integer specifying the height of the text track in units of the same pixels as used to specify the `text-track-width` value.

`number-of-formats` – unsigned integer specifying the number of immediately following `Compatible-3GPPFormat` fields. The value ‘0’ is forbidden.

`Compatible-3GPPFormat` – one byte that signals a 3GPP formats older than the signaled format by `3GPPBaseFormat`, that the text stream is also compatible with, coded in the same way as the `3GPPBaseFormat` field, specified in Table 5.

`number-of-SampleDescriptions` – unsigned integer that specifies the number of immediately following TTU[5]s.

`TTU[5]()` – a TTU[5] as specified in subclauses 7.4.2 and 7.4.8, carrying one sample description.

`scene-width` – unsigned integer specifying the width of the scene in pixels[2]. If the scene only consists of a video picture, then `scene-width` signals the width of the video picture. If the width of the scene is also provided by external means, then the externally provided information takes priority.

`scene-height` – unsigned integer specifying the height of the scene in pixels; see footnote 2. If the scene only consists of a video picture, then `scene-height` signals the height of the video picture. If the height of the scene is also provided by external means, then the externally provided information takes priority.

2) These are the square pixels used to specify the Text Track dimensions, not scene metrics or coded video pixels.

`horizontal-scene-offset` – signed integer specifying the horizontal distance in pixels (see footnote 2) between the scene origin and the left border of the Text Track; a positive distance value indicates that the left border of the Text Track is right of the left border of the scene or the video picture. If this distance is also provided by external means, then the externally provided information takes priority.

`vertical-scene-offset` – signed integer specifying the vertical distance in pixels (see footnote 2) between the top border of the scene or the video picture and the top border of the Text Track; a positive distance value indicates that the top border of the Text Box is below the top border of the scene or the video picture. If this distance is also provided by external means, then the externally provided information takes priority.

`remaining-config-data-length` – unsigned integer, specifying the number of immediately following `reserved` bytes in this formatSpecificTextConfig.

`sample_index` – the index assigned to the sample description contained in the immediately following `sample_description()`. The `sample_index` value shall unambiguously refer to one particular sample description and shall be in the inclusive range between 128 and 254, as assigned to out-of-band provided sample descriptions in Table 2.

`sample_description()` – a sample description for a TextSample, referred to as TextDescription, as defined by 3GPP in 3GPP TS 26.245 or an extension thereof, specified in textConfig (see also subclauses 5.3 and 5.4).

7.7 Hypothetical Text Decoder for 3GPP text stream decoding

This subclause defines a hypothetical decoder for decoding of 3GPP text streams, the Hypothetical Text Decoder (HTD). The HTD consists of a TTU decoder, a textSampleBuffer, a sampleDescriptionBuffer and a textDecoder; see Figure 5. Input to the HTD is a 3GPP text stream with a maximum bitrate of R kb/s. The value of R, as well as the textSampleBuffer size and the sampleDescriptionBuffer size is specified for each 3GPP text stream per profile and level, as signaled in textConfig; see subclauses 5.3 and 7.6.

At the input of the HTD, the TTU decoder processes each received byte of the 3GPP text stream instantaneously; each byte of a text sample in the 3GPP text stream is provided to the textSampleBuffer, and each byte of a sample description is provided to the sampleDescriptionBuffer. At any point in time, the total size in bytes of all valid descriptive data shall not exceed the specified size of the sampleDescriptionBuffer. 3GPP text streams shall be constructed so, that each sample description is completely present in the sampleDescriptionBuffer at the decoding time of a text sample that requires data from that sample description.

If the textSampleBuffer is not full, then the bytes of the 3GPP text stream enter the TTU decoder at a bitrate of R kb/s; if the textSampleBuffer is full, no bytes enter the TTU decoder. Bytes in the 3GPP text stream subsequent to the byte that causes the textSampleBuffer to be full do not enter the TTU decoder as long as the textSampleBuffer remains full. At the instant in time that the presentation of a text sample is to start, the text sample is instantaneously removed from the textSampleBuffer, decoded and presented. Underflow of the textSampleBuffer occurs if a text sample has not (yet) fully entered the textSampleBuffer at the instant in time that the text sample is to be presented. Text streams shall be constructed so, that in the Hypothetical Text Decoder no underflow occurs at any instant in time. The following requirements apply:

- The size of each text sample shall be smaller than or equal to the specified size of the textSampleBuffer;
- Taking the fullness of the textSampleBuffer into account, there shall be sufficient time to deliver the complete text sample to the textSampleBuffer at the specified bitrate prior to its decoding.

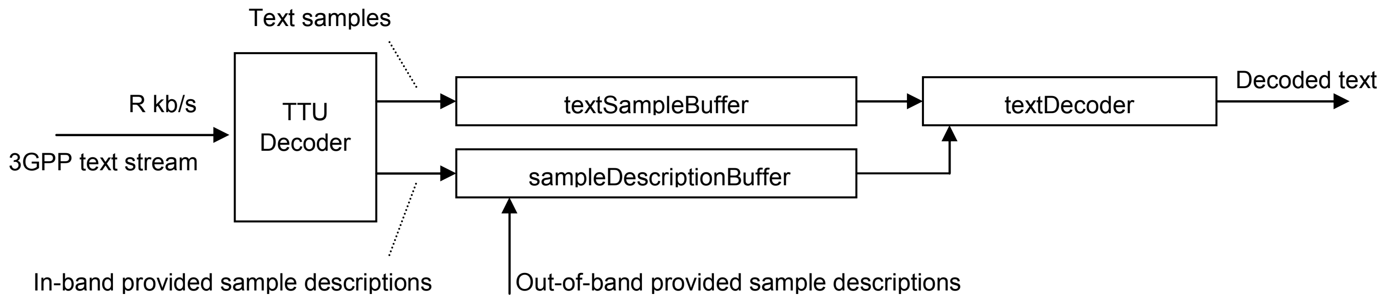

Figure 5 — Hypothetical Text Decoder for 3GPP text streams

NOTE In the extreme case that the size of the previous text sample is equal to the size of textSampleBuffer, then the next text sample is to be delivered during the duration of the previous one. To avoid underflow in this extreme case, the encoder has to ensure that the duration between the previous non-empty sample and the next non-empty sample N (note that there may be one or more "empty" samples in-between) is larger than the size of the non-empty sample N divided by the applicable bitrate R.

7.8 Profile and Level parameters for 3GPP text streams

In each 3GPP text stream that conforms to the base profile, each tool specified by 3GPP in 3GPP TS 26.245 may be used. In each 3GPP text stream that complies with the base level, the HTD parameter values defined in Table 5 shall apply. Each decoder that complies with the base profile and the base level shall be capable of decoding each text stream that conforms to the base profile and the base level.

Table 8 — HTD parameter values for the Base Level

HTD Parameter	**Value**
Bitrate R	10 kb/s
Size of textSampleBuffer	8192 Byte
Size of in-band-sampleDescriptionBuffer	4096 Byte
Size of out-of-band-sampleDescriptionBuffer	4096 Byte

Annex A
(normative)

Font Referencing

A.1 Font references in text streams

Text streams may reference fonts in the receiver. These fonts may be residently available in the receiver, or downloaded or streamed to the receiver. To ensure that the desired fonts in the receiver are referenced correctly, the font name identification in textData() has to uniquely reference the name of the desired font in the receiver.

In an MPEG-4 systems context, when a font stream is used to supply fonts for a text stream, that text stream shall indicate that dependency by using the dependsonES_ID field in the ES_Descriptor of the text stream, containing the ES_ID of the font stream. Other systems contexts should indicate that dependency in an appropriate way.

In text streams which have an associated font stream, when a specific font instance in the font stream is to be used for a text access unit, the font name shall use the appropriate naming structure, which for MPEG-4 font streams is defined in subclause 5.5, ISO/IEC 14496-18.

Annex B
(informative)

Transport of Text Streams

B.1 Transport tools for 3GPP Timed Text

In this Specification a flexible and transport generic framing structure is defined that can be conveniently adapted for transport of text streams accross various transport layers, such as MPEG-2 Systems [1] for use in media such as broadcast and optical discs and RTP [2] for transport over IP. For transport over IP, IETF defined an RTP payload format for transport of generic MPEG-4 content [3] that is also suitable to transport ISO/IEC 14496-17 text streams. On the other hand, in IETF also an RTP payload format for 3GPP Timed Text [4] has been defined that directly packetizes 3GPP Timed Text data, without first constructing a ISO/IEC 14496-17 text stream; see also Figure B.1.

To ensure maximum interoperability, the RTP payload format for 3GPP Timed Text [4] has been developed in close cooperation with the authors of this Specification; as a result, this Specifications uses exactly the same structure for transport of 3GPP timed text data as RFC xxxx, thereby allowing the RTP packets payloads of RFC 3640 and RFC xxxx to be fully identical, in which case only the associated signaling differs. However, the folllowing should be carfully noted. In RFC xxxx some packetization rules for 3GPP timed text data are specified; to achieve fully identical RFC xxxx and RFC 3640 payloads, the same packetization rules should be used for RFC 3640.

The intention is to maintain the use of exactly the same transport structure for 3GPP timed text data in RFC xxxx and ISO/IEC 14496-17. This would ensure that the option of identical payload formats for [2] and [3] is maintained in future. However, it should be noted that these formats will diverge in case ISO/IEC 14496-17 and RFC xxxx diverge in future.

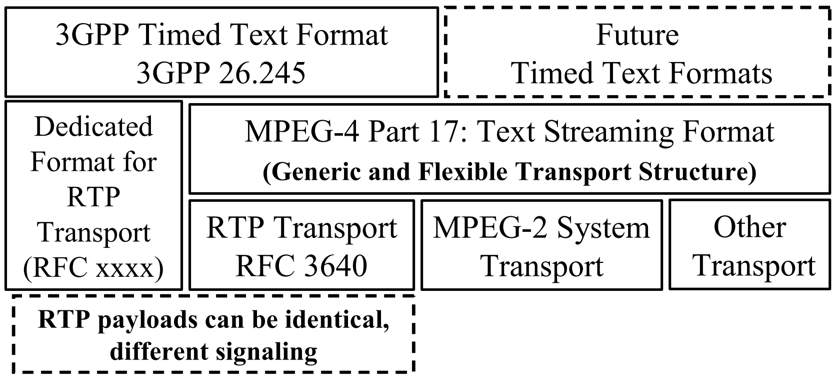
3GPP Timed Text Format
3GPP 26.245
Future
Timed Text Formats
Dedicated Format for RTP Transport (RFC xxxx)
MPEG-4 Part 17: Text Streaming Format
(Generic and Flexible Transport Structure)
RTP Transport RFC 3640
MPEG-2 System Transport
Other Transport
RTP payloads can be identical, different signaling

Figure B.1 — ISO/IEC 14496-17 and transport of 3GPP Timed Text

The RFC xxxx and RFC 3640 payloads can be identical, but only if the same packetization rules are used for RFC 3640 as for RFC xxxx.

Bibliography

[1] MPEG-2 Systems: ITU-T Rec. H.262 | ISO/IEC 13818-1, *Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems*

[2] IETF RFC 3550, *RTP, A Transport Protocol for Real Time Applications*

[3] IETF RFC 3640, *RTP payload for transport of generic MPEG-4 content*

[4] IETF RFC xxxx, *RTP payload for 3GPP Timed Text**

* To be published, number to be assigned.

ICS 35.040

Price based on 21 pages