

ETSI TS 126 245 V8.0.0 (2009-01)

Technical Specification

**Universal Mobile Telecommunications System (UMTS);
LTE;
Transparent end-to-end Packet
switched Streaming Service (PSS);
Timed text format
(3GPP TS 26.245 version 8.0.0 Release 8)**



Reference

RTS/TSGS-0426245v800

Keywords

LTE, UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at

<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, please send your comment to one of the following services:

http://portal.etsi.org/chaicor/ETSI_support.asp

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2009.
All rights reserved.

DECTTM, **PLUGTESTS**TM, **UMTS**TM, **TIPHON**TM, the TIPHON logo and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.

3GPPTM is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

LTETM is a Trade Mark of ETSI currently being registered

for the benefit of its Members and of the 3GPP Organizational Partners.

GSM[®] and the GSM logo are Trade Marks registered and owned by the GSM Association.

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<http://webapp.etsi.org/IPR/home.asp>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under <http://webapp.etsi.org/key/queryform.asp>.

Contents

Intellectual Property Rights	2
Foreword.....	2
Foreword.....	4
Introduction	4
1 Scope	5
2 References	5
3 Definitions and abbreviations.....	5
3.1 Definitions	5
3.2 Abbreviations	6
4 Overview	6
5 Timed text format.....	6
5.1 Unicode Support.....	6
5.2 Bytes, Characters, and Glyphs.....	6
5.3 Character Set Support.....	7
5.4 Font Support.....	7
5.5 Fonts and Metrics	7
5.6 Colour Support	8
5.7 Text rendering position and composition	8
5.8 Marquee Scrolling.....	9
5.9 Language	10
5.10 Writing direction	10
5.11 Text wrap.....	11
5.12 Highlighting, Closed Caption, and Karaoke.....	11
5.13 Media Handler.....	11
5.14 Media Handler Header	11
5.15 Style record	11
5.16 Sample Description Format	12
5.17 Sample Format	13
5.17.1 Sample Modifier Boxes	14
5.17.1.1 Text Style	14
5.17.1.2 Highlight	14
5.17.1.3 Dynamic Highlight.....	14
5.17.1.4 Scroll Delay.....	15
5.17.1.5 HyperText	15
5.17.1.6 Textbox	16
5.17.1.7 Blink.....	16
5.17.1.8 Text Wrap Indication	16
5.18 Combinations of features.....	16
Annex A (informative): Change history	18
History	19

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
 - 1 presented to TSG for information;
 - 2 presented to TSG for approval;
 - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

The 3GPP transparent end-to-end packet-switched streaming service (PSS) specification consists of six 3GPP TSs: 3GPP TS 22.233 [1], 3GPP TS 26.233 [2], 3GPP TS 26.234 [3], 3GPP TS 26.244 [4], 3GPP TS 26.246 [5] and the present document.

The TS 22.233 contains the service requirements for the PSS. The TS 26.233 provides an overview of the PSS. The TS 26.234 provides the details of protocol and codecs used by the PSS. The TS 26.244 defines the 3GPP file format (3GP) used by the PPS and MMS services. The TS 26.246 defines the 3GPP SMIL language profile. The present document defines the Timed text format used by the PSS.

The TS 26.244, TS 26.245 (present document) and TS 26.246 start with Release 6. Earlier releases of the 3GPP file format, the Timed text format and the 3GPP SMIL language profile can be found in TS 26.234.

Introduction

Timed text is text that is rendered at the terminal, in synchronization with other timed media such as video or audio. Timed text is used for such applications as closed captioning, titling, and other visual annotation of timed media.

1 Scope

The present document defines the timed text format relative to the 3GPP file format. This specification defines the format of timed text in downloaded files.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

- [1] 3GPP TS 22.233: "Transparent End-to-End Packet-switched Streaming Service; Service aspects; Stage 1".
- [2] 3GPP TS 26.233: "Transparent end-to-end packet switched streaming service (PSS); General description".
- [3] 3GPP TS 26.234: "Transparent end-to-end packet switched streaming service (PSS); Protocols and codecs".
- [4] 3GPP TS 26.244: "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)".
- [5] 3GPP TS 26.246: "Transparent end-to-end packet switched streaming service (PSS); 3GPP SMIL Language Profile".
- [6] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".
- [7] The Unicode Consortium: "The Unicode Standard", Version 3.0 Reading, MA, Addison-Wesley Developers Press, 2000, ISBN 0-201-61633-5.
- [8] "Unicode Standard Annex #13: Unicode Newline Guidelines", by Mark Davis. An integral part of The Unicode Standard, Version 3.1.
- [9] ISO/IEC 14496-14:2003 "Information technology – Coding of audio-visual objects – Part 14: MP4 file format".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

continuous media: media with an inherent notion of time. In the present document speech, audio, video and timed text

discrete media: media that itself does not contain an element of time. In the present document all media not defined as continuous media

PSS client: client for the 3GPP packet switched streaming service based on the IETF RTSP/SDP and/or HTTP standards, with possible additional 3GPP requirements according to the present document

PSS server: server for the 3GPP packet switched streaming service based on the IETF RTSP/SDP and/or HTTP standards, with possible additional 3GPP requirements according to the present document

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [6] and the following apply.

3GP	3GPP file format
MMS	Multimedia Messaging Service
MP4	MPEG-4 file format
PSS	Packet-switched Streaming Service
SMIL	Synchronised Multimedia Integration Language
UTF-8	Unicode Transformation Format (the 8-bit form)
UTF-16	Unicode Transformation Format (the 16-bit form)

4 Overview

Operators may specify additional rules and restrictions when deploying terminals, in addition to this specification, and behavior that is optional here may be mandatory for particular deployments. In particular, the required character set is almost certainly dependent on the geography of the deployment.

5 Timed text format

5.1 Unicode Support

Text in this specification uses the Unicode 3.0 [7] standard. Terminals shall correctly decode both UTF-8 and UTF-16 into the required characters. If a terminal receives a Unicode code, which it cannot display, it shall display a predictable result. It shall not treat multi-byte UTF-8 characters as a series of ASCII characters, for example.

Authors should create fully-composed Unicode; terminals are not required to handle decomposed sequences for which there is a fully-composed equivalent.

Terminals shall conform to the conformance statement in Unicode 3.0 section 3.1.

Text strings for display and font names are uniformly coded in UTF-8, or start with a UTF-16 BYTE ORDER MARK (`\uFEFF`) and by that indicate that the string which starts with the byte order mark is in UTF-16. Terminals shall recognise the byte-order mark in this byte order; they are not required to recognise byte-reversed UTF-16, indicated by a byte-reversed byte-order mark.

5.2 Bytes, Characters, and Glyphs

This clause uses these terms carefully. Since multi-byte characters are permitted (i.e. 16-bit Unicode characters), the number of characters in a string may not be the number of bytes. Also, a byte-order-mark is not a character at all, though it occupies two bytes. So, for example, storage lengths are specified as byte-counts, whereas highlighting is specified using character offsets.

It should also be noted that in some writing systems the number of glyphs rendered might be different again. For example, in English, the characters "fi" are sometimes rendered as a single ligature glyph.

In this specification, the first character is at offset 0 in the string. In records specifying both a start and end offset, the end offset shall be greater than or equal to the start offset. In cases where several offset specifications occur in sequence, the start offset of an element shall be greater than or equal to the end offset of the preceding element.

5.3 Character Set Support

All terminals shall be able to render Unicode characters in these ranges:

- a) basic ASCII and Latin-1 (\u0000 to \u00FF), though not all the control characters in this range are needed;
- b) the Euro currency symbol (\u20AC)
- c) telephone and ballot symbols (\u260E through \u2612)

Support for the following characters is recommended but not required:

- a) miscellaneous technical symbols (\u2300 through \u2335)
- b) "Zapf Dingbats": locations \u2700 through \u27AF, and the locations where some symbols have been relocated (e.g. \u2605, Black star).

The private use characters \u0091 and \u0092, and the initial range of the private use area \uE000 through \uE0FF are reserved in this specification. For these Unicode values, and for control characters for which there is no defined graphical behaviour, the terminal shall not display any result: neither a glyph is shown nor is the current rendering position changed.

5.4 Font Support

Fonts are specified in this specification by name, size, and style. There are three special names which shall be recognized by the terminal: Serif, Sans-Serif, and Monospace. It is strongly recommended that these be different fonts for the required characters from ASCII and Latin-1. For many other characters, the terminal may have a limited set or only a single font. Terminals requested to render a character where the selected font does not support that character should substitute a suitable font. This ensures that languages with only one font (e.g. Asian languages) or symbols for which there is only one form are rendered.

Fonts are requested by name, in an ordered list. Authors should normally specify one of the special names last in the list.

Terminals shall support a pixel size of 12 (on a 72dpi display, this would be a point size of 12). If a size is requested other than the size(s) supported by the terminal, the next smaller supported size should be used. If the requested size is smaller than the smallest supported size, the terminal should use the smallest supported size.

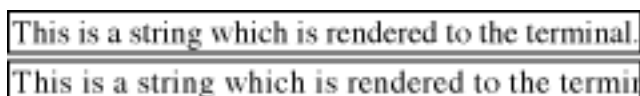
Terminals shall support unstyled text for those characters it supports. It may also support bold, italic (oblique) and bold-italic. If a style is requested which the terminal does not support, it should substitute a supported style; a character shall be rendered if the terminal has that character in any style of any font.

5.5 Fonts and Metrics

Within the sample description, a complete list of the fonts used in the samples is found. This enables the terminal to pre-load them, or to decide on font substitution.

Terminals may use varying versions of the same font. For example, here is the same text rendered on two systems; it was authored on the first, where it just fitted into the text box.

EXAMPLE:



This is a string which is rendered to the terminal.

This is a string which is rendered to the termi

Authors should be aware of this possible variation, and provide text box areas with some "slack" to allow for rendering variations.

5.6 Colour Support

The colour of both text and background are indicated in this specification using RGB values. Terminals are not required to be able to display all colours in the RGB space. Terminals with a limited colour display, with only gray-scale display, and with only black-and-white are permissible. If a terminal has a limited colour capability it should substitute a suitable colour; dithering of text may be used but is not usually appropriate as it results in 'fuzzy' display. If colour substitution is performed, the substitution shall be consistent: the same RGB colour shall result consistently in the same displayed colour. If the same colour is chosen for background and text, then the text shall be invisible (unless a style such as highlight changes its colour). If different colours are specified for the background and text, the terminal shall map these to different colours, so that the text is visible.

Colours in this specification also have an alpha or transparency value. In this specification, a transparency value of 0 indicates a fully transparent colour, and a value of 255 indicates fully opaque. Support for partial or full transparency is optional. "Keying" text (text rendered on a transparent background) is done by using a background colour which is fully transparent. "Keying" text over video or pictures, and support for transparency in general, can be complex and may require double-buffering, and its support is optional in the terminal. Content authors should beware that if they specify a colour which is not fully opaque, and the content is played on a terminal not supporting it, the affected area (the entire text box for a background colour) will be fully opaque and will obscure visual material behind it. Visual material with transparency is layered closer to the viewer than the material which it partially obscures.

5.7 Text rendering position and composition

Text is rendered within a region (a concept derived from SMIL). There is a text box set within that region. This permits the terminal to position the text within the overall presentation, and also to render the text appropriately given the writing direction. For text written left to right, for example, the first character would be rendered at, or near, the left edge of the box, and with its baseline down from the top of the box by one baseline height (a value derived from the font and font size chosen). Similar considerations apply to the other writing directions.

Within the region, text is rendered within a text box. There is a default text box set, which can be over-ridden by a sample.

Either the text box or text region is filled with the background colour; after that the text is painted in the text colour. If highlighting is requested one or both of these colours may vary.

Terminals may choose to anti-alias their text, or not.

The text region and layering are defined using structures from the ISO base media file format.

This track header box is used for text track:

```
aligned(8) class TrackHeaderBox
  extends FullBox("tkhd", version, flags){
  if (version==1) {
    unsigned int(64)    creation_time;
    unsigned int(64)    modification_time;
    unsigned int(32)    track_ID;
    const unsigned int(32) reserved = 0;
    unsigned int(64)    duration;
  } else { // version==0
    unsigned int(32)    creation_time;
    unsigned int(32)    modification_time;
    unsigned int(32)    track_ID;
    const unsigned int(32) reserved = 0;
    unsigned int(32)    duration;
  }
  const unsigned int(32)[2] reserved = 0;
  int(16) layer;
  template int(16) alternate_group = 0;
  template int(16) volume = 0;
  const unsigned int(16) reserved = 0;
  template int(32)[9] matrix=
    { 0x00010000,0,0,0,0x00010000,0,tx,ty,0x40000000 };
    // unity matrix
  unsigned int(32) width;
  unsigned int(32) height;
}
```

Visually composed tracks including video and text are layered using the "layer" value. This compares, for example, to z-index in SMIL. More negative layer values are towards the viewer. (This definition is compatible with that in ISO/MJ2).

The region is defined by the track width and height, and translation offset. This corresponds to the SMIL region. The width and height are stored in the track header fields above. The sample description sets a text box within the region, which can be over-ridden by the samples.

The translation values are stored in the track header matrix in the following positions:

```
{ 0x00010000,0,0, 0,0x00010000,0, tx, ty, 0x40000000 }
```

These values are fixed-point 16.16 values, here restricted to be integers (the lower 16 bits of each value shall be zero). The X axis increases from left to right; the Y axis from top to bottom. (This use of the matrix is conformant with ISO/MJ2.)

So, for example, a centered region of size 200x20, positioned below a video of size 320x240, would have track_width set to 200 (width= 0x00c80000), track_height set to 20 (height= 0x00140000), and tx = (320-200)/2 = 60, and ty=240.

Since matrices are not used on the video tracks, all video tracks are set at the coordinate origin. Figure 5.1 provides an overview:

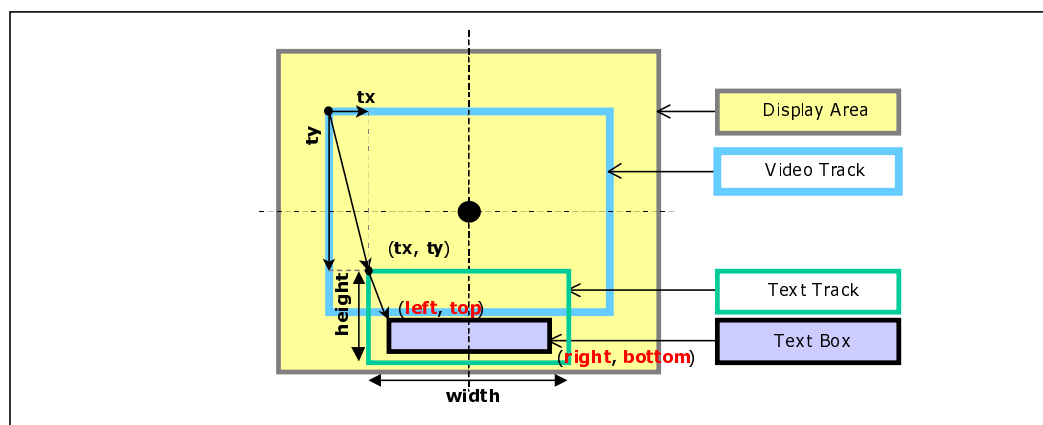


Figure 5.1: Illustration of text rendering position and composition

The top and left positions of the text track is determined by the tx and ty, which are the translation values from the coordinate origin (since the video track is at the origin, this is also the offset from the video track). The default text box set in the sample description sets the rendering area unless over-ridden by a 'tbox' in the text sample. The box values are defined as the relative values from the top and left positions of the text track.

It should be noted that this only specifies the relationship of the tracks within a single 3GP file. If a SMIL presentation lays up multiple files, their relative position is set by the SMIL regions. Each file is assigned to a region, and then within those regions the spatial relationship of the tracks is defined.

5.8 Marquee Scrolling

Text can be "marquee" scrolled in this specification (compare this to Internet Explorer's marquee construction). When scrolling is performed, the terminal first calculates the position in which the text would be displayed with no scrolling requested. Then:

- If scroll-in is requested, the text is initially invisible, just outside the text box, and enters the box in the indicated direction, scrolling until it is in the normal position;
- If scroll-out is requested, the text scrolls from the normal position, in the indicated direction, until it is completely outside the text box.

The rendered text is clipped to the text box in each display position, as always. This means that it is possible to scroll a string which is longer than can fit into the text box, progressively disclosing it (for example, like a ticker-tape). Note

that both scroll in and scroll out may be specified; the text scrolls continuously from its invisible initial position, through the normal position, and out to its final position.

If a scroll-delay is specified, the text stays steady in its normal position (not initial position) for the duration of the delay; so the delay is after a scroll-in but before a scroll-out. This means that the scrolling is not continuous if both are specified. So without a delay, the text is in motion for the duration of the sample. For a scroll in, it reaches its normal position at the end of the sample duration; with a delay, it reaches its normal position before the end of the sample duration, and remains in its normal position for the delay duration, which ends at the end of the sample duration. Similarly for a scroll out, the delay happens in its normal position before scrolling starts. If both scroll in, and scroll out are specified, with a delay, the text scrolls in, stays stationary at the normal position for the delay period, and then scrolls out – all within the sample duration.

The speed of scrolling is calculated so that the complete operation takes place within the duration of the sample. Therefore the scrolling has to occur within the time left after scroll-delay has been subtracted from the sample duration. Note that the time it takes to scroll a string may depend on the rendered length of the actual text string. Authors should consider whether the scrolling speed that results will be exceed that at which text on a wireless terminal could be readable.

Terminals may use simple algorithms to determine the actual scroll speed. For example, the speed may be determined by moving the text an integer number of pixels in every update cycle. Terminals should choose a scroll speed which is as fast or faster than needed so that the scroll operation completes within the sample duration.

Terminals are not required to handle dynamic or stylistic effects such as highlight, dynamic highlight, or href links on scrolled text.

The scrolling direction is set by a two-bit field, with the following possible values:

- 00b – text is vertically scrolled up ("credits style"), entering from the bottom of the bottom and leaving towards the top.
- 01b – text is horizontally scrolled ("marquee style"), entering from the right and leaving towards the left.
- 10b – text is vertically scrolled down, entering from the top and leaving towards the bottom.
- 11b – text is horizontally scrolled, entering from the left and leaving towards the right.

5.9 Language

The human language used in this stream is declared by the language field of the media-header box in this track. It is an ISO 639/T 3-letter code. The knowledge of the language used might assist searching, or speaking the text. Rendering is language neutral. Note that the values "und" (undetermined) and "mul" (multiple languages) might occur.

5.10 Writing direction

Writing direction specifies the way in which the character position changes after each character is rendered. It also will imply a start-point for the rendering within the box.

Terminals shall support the determination of writing direction, for those characters they support, according to the Unicode 3.0 specification. Note that the only required characters can all be rendered using left-right behaviour. A terminal which supports characters with right-left writing direction shall support the right-left composition rules specified in Unicode.

Terminals may also set, or allow the user to set, an overall writing direction, either explicitly or implicitly (e.g. by the language selection). This affects layout. For example, if upper-case letters are left-right, and lower-case right-left, and the Unicode string ABCdefGHI shall be rendered, it would appear as ABCfedGHI on a terminal with overall left-right writing (English, for example) and GHIfedABC on a system with overall right-left (Hebrew, for example).

Terminals are not required to support the bi-directional ordering codes (`\u200E`, `\u200F` and `\u202A` through `\u202E`).

If vertical text is requested by the content author, characters are laid out vertically from top to bottom. The terminal may choose to render different glyphs for this writing direction (e.g. a horizontal parenthesis), but in general the glyphs should not be rotated. The direction in which lines advance (left-right, as used for European languages, or right-left, as used for Asian languages) is set by the terminal, possibly by a direct or indirect user preference (e.g. a language setting).

Terminals shall support vertical writing of the required character set. It is recommended that terminals support vertical writing of text in those languages commonly written vertically (e.g. Asian languages). If vertical text is requested for characters which the terminal cannot render vertically, the terminal may behave as if the characters were not available.

5.11 Text wrap

Automatic wrapping of text from line to line is complex, and can require hyphenation rules and other complex language-specific criteria. For these reasons, soft text wrap is optional in this specification. Text wrap behavior may be specified using a TextWrapBox, and a terminal that does not support this feature shall not perform soft text wrapping. When text wrap is not used and a string is too long to be drawn within the box, it is clipped. The terminal may choose whether to clip at the pixel boundary, or to render only whole glyphs.

There may be multiple lines of text in a sample (hard wrap). Terminals shall start a new line for the Unicode characters line separator (\u2028), paragraph separator (\u2029) and line feed (\u000A). It is recommended that terminals follow Unicode Technical Report 13 [8]. Terminals should treat carriage return (\u000D), next line (\u0085) and CR+LF (\u000D\u000A) as new line.

5.12 Highlighting, Closed Caption, and Karaoke

Text may be highlighted for emphasis. Since this is a non-interactive system, solely for text display, the utility of this function may be limited.

Dynamic highlighting used for Closed Caption and Karaoke highlighting, is an extension of highlighting. Successive contiguous sub-strings of the text sample are highlighted at the specified times.

5.13 Media Handler

A text stream is its own unique stream type. For the 3GPP file format, the handler-type within the "hdlr" box shall be "text".

5.14 Media Handler Header

The 3G text track uses an empty null media header ("nmhd"), called Mpeg4MediaHeaderBox in the MP4 specification [9], in common with other MPEG streams.

```
aligned(8) class Mpeg4MediaHeaderBox
    extends FullBox("nmhd", version = 0, flags) {
}
```

5.15 Style record

Both the sample format and the sample description contain style records, and so it is defined once here for compactness.

```
aligned(8) class StyleRecord {
    unsigned int(16) startChar;
    unsigned int(16) endChar;
    unsigned int(16) font-ID;
    unsigned int(8) face-style-flags;
    unsigned int(8) font-size;
    unsigned int(8) text-color-rgba[4];
}
```

- startChar:** character offset of the beginning of this style run (always 0 in a sample description)
- endChar:** first character offset to which this style does not apply (always 0 in a sample description); shall be greater than or equal to startChar. All characters, including line-break characters and any other non-printing characters, are included in the character counts.
- font-ID:** font identifier from the font table; in a sample description, this is the default font
- face style flags:** in the absence of any bits set, the text is plain

- 1 bold
- 2 italic
- 4 underline

font-size: font size (nominal pixel size, in essentially the same units as the width and height)

text-color-rgba: rgb colour, 8 bits each of red, green, blue, and an alpha (transparency) value

Terminals shall support plain text, and underlined horizontal text, and may support bold, italic and bold-italic depending on their capabilities and the font selected. If a style is not supported, the text shall still be rendered in the closest style available.

5.16 Sample Description Format

The sample table box ('stbl') contains sample descriptions for the text track. Each entry is a sample entry box of type "tx3g". This name defines the format both of the sample description and the samples associated with that sample description. Terminals shall not attempt to decode or display sample descriptions with unrecognised names, nor the samples attached to those sample descriptions.

It starts with the standard fields (the reserved bytes and the data reference index), and then some text-specific fields. Some fields can be overridden or supplemented by additional boxes within the text sample itself. These are discussed below.

There can be multiple text sample descriptions in the sample table. If the overall text characteristics do not change from one sample to the next, the same sample description is used. Otherwise, a new sample description is added to the table. Not all changes to text characteristics require a new sample description, however. Some characteristics, such as font size, can be overridden on a character-by-character basis. Some, such as dynamic highlighting, are not part of the text sample description and can be changed dynamically.

The TextDescription extends the regular sample entry with the following fields.

```
class FontRecord {
    unsigned int(16) font-ID;
    unsigned int(8) font-name-length;
    unsigned int(8) font[font-name-length];
}

class FontTableBox() extends Box("ftab") {
    unsigned int(16) entry-count;
    FontRecord font-entry[entry-count];
}

class BoxRecord {
    signed int(16) top;
    signed int(16) left;
    signed int(16) bottom;
    signed int(16) right;
}

class TextSampleEntry() extends SampleEntry ("tx3g") {
    unsigned int(32) displayFlags;
    signed int(8) horizontal-justification;
    signed int(8) vertical-justification;
    unsigned int(8) background-color-rgba[4];
    BoxRecord default-text-box;
    StyleRecord default-style;
    FontTableBox font-table;
}
```

displayFlags:

scroll In	0x00000020	
scroll Out	0x00000040	
scroll direction	0x00000180	/ see above for values
continuous karaoke	0x00000800	
write text vertically	0x00020000	
fill text region	0x00040000	

horizontal and vertical justification: / two eight-bit values from the following list:

left, top 0
centered 1
bottom, right -1

background-color-rgba:

rgb color, 8 bits each of red, green, blue, and an alpha (transparency) value

Default text box: the default text box is set by four values, relative to the text region; it may be over-ridden in samples;

style record of default style: startChar and endChar shall be zero in a sample description

The text box is inset within the region defined by the track translation offset, width, and height. The values in the box are relative to the track region, and are uniformly coded with respect to the pixel grid. So, for example, the default text box for a track at the top left of the track region and 50 pixels high and 100 pixels wide is {0, 0, 50, 100}.

If the "fill text region" flag is 0 (the default value, and the value from previous releases), then the background fill is applied to the text box only. If this flag is 1, then the author is requesting that the background fill be applied to the entire text region, if possible. Note that this flag was not defined in previous releases and will not therefore always be interpreted. Implementation of this flag is recommended but not required for compliance.

A font table shall follow these fields, to define the complete set of fonts used. The font table is an box of type "ftab". Every font used in the samples is defined here by name. Each entry consists of a 16-bit local font identifier, and a font name, expressed as a string, preceded by an 8-bit field giving the length of the string in bytes. The name is expressed in UTF-8 characters, unless preceded by a UTF-16 byte-order-mark, whereupon the rest of the string is in 16-bit Unicode characters. The string should be a comma separated list of font names to be used as alternative font, in preference order. The special names 'Serif', 'Sans-serif' and 'Monospace' may be used. The terminal should use the first font in the list which it can support; if it cannot support any for a given character, but it has a font which can, it should use that font. Note that this substitution is technically character by character, but terminals are encouraged to keep runs of characters in a consistent font where possible.

5.17 Sample Format

Each sample in the media data consists of a string of text, optionally followed by sample modifier boxes.

For example, if one word in the sample has a different size than the others, a 'styl' box is appended to that sample, specifying a new text style for those characters, and for the remaining characters in the sample. This overrides the style in the sample description. These boxes are present only if they are needed. If all text conforms to the sample description, and no characteristics are applied that the sample description does not cover, no boxes are inserted into the sample data.

```
class TextSampleModifierBox(type) extends Box(type) {
}

class TextSample {
    unsigned int(16)      text-length;
    unsigned int(8)      text[text-length];
    TextSampleModifierBox text-modifier[]; // to end of the sample
}
```

The initial string is preceded by a 16-bit count of the number of bytes in the string. There is no need for null termination of the text string. The sample size table provides the complete byte-count of each sample, including the trailing modifier boxes; by comparing the string length and the sample size, you can determine how much space, if any, is left for modifier boxes.

Authors should limit the string in each text sample to not more than 2048 bytes, for maximum terminal interoperability.

Any unrecognised box found in the text sample should be skipped and ignored, and processing continue as if it were not there.

5.17.1 Sample Modifier Boxes

5.17.1.1 Text Style

'styl'

This specifies the style of the text. It consists of a series of style records as defined above, preceded by a 16-bit count of the number of style records. Each record specifies the starting and ending character positions of the text to which it applies. The styles shall be ordered by starting character offset, and the starting offset of one style record shall be greater than or equal to the ending character offset of the preceding record; styles records shall not overlap their character ranges.

```
class TextStyleBox() extends TextSampleModifierBox ("styl") {
    unsigned int(16)    entry-count;
    StyleRecord        text-styles[entry-count];
}
```

5.17.1.2 Highlight

'hlit' - Specifies highlighted text: the box contains two 16-bit integers, the starting character to highlight, and the first character with no highlighting (e.g. values 4, 6 would highlight the two characters 4 and 5). The second value may be the number of characters in the text plus one, to indicate that the last character is highlighted.

```
class TextHighlightBox() extends TextSampleModifierBox ("hlit") {
    unsigned int(16)    startcharoffset;
    unsigned int(16)    endcharoffset;
}
class TextHilightColorBox() extends TextSampleModifierBox ('hclr') {
    unsigned int(8)     highlight_color_rgba[4];
}
```

highlight_color_rgb:

rgb color, 8 bits each of red, green, blue, and an alpha (transparency) value

The TextHilightColor Box may be present when the TextHighlightBox or TextKaraokeBox is present in a text sample. It is recommended that terminals use the following rules to determine the displayed effect when highlight is requested:

- a) if a highlight colour is not specified, then the text is highlighted using a suitable technique such as inverse video: both the text colour and the background colour change.
- b) if a highlight colour is specified, the background colour is set to the highlight colour for the highlighted characters; the text colour does not change.

Terminals do not need to handle text that is both scrolled and either statically or dynamically highlighted. Content authors should avoid specifying both scroll and highlight for the same sample.

5.17.1.3 Dynamic Highlight

'krok' – Karaoke, closed caption, or dynamic highlighting. The number of highlight events is specified, and each event is specified by a starting and ending character offset and an end time for the event. The start time is either the sample start time or the end time of the previous event. The specified characters are highlighted from the previous end-time (initially the beginning of this sample's time), to the end time. The times are all specified relative to the sample's time; that is, a time of 0 represents the beginning of the sample time. The times are measured in the timescale of the track.

The box starts with the start-time offset of the first highlight event, a 16-bit count of the event count, and then that number of 8-byte records. Each record contains the end-time offset as a 32-bit number, and the text start and end values, each as a 16-bit number. These values are specified as in the highlight record – the offset of the first character to highlight, and the offset of the first character not highlighted. The special case, where the startcharoffset equals to the endcharoffset, can be used to pause during or at the beginning of dynamic highlighting. The records shall be ordered and not overlap, as in the highlight record. The time in each record is the end time of this highlight event; the first

highlight event starts at the indicated start-time offset from the start time of the sample. The time values are in the units expressed by the timescale of the track. The time values shall not exceed the duration of the sample.

The continuouskaraoke flag controls whether to highlight only those characters (continuouskaraoke = 0) selected by a karaoke entry, or the entire string from the beginning up to the characters highlighted (continuouskaraoke = 1) at any given time. In other words, the flag specifies whether karaoke should ignore the starting offset and highlight all text from the beginning of the sample to the ending offset.

Karaoke highlighting is usually achieved by using the highlight colour as the text colour, without changing the background.

At most one dynamic highlight ("krok") box may occur in a sample.

```
class TextKaraokeBox() extends TextSampleModifierBox ("krok") {
    unsigned int(32)    highlight-start-time;
    unsigned int(16)    entry-count;
    for (i=1; i<=entry-count; i++) {
        unsigned int(32)    highlight-end-time;
        unsigned int(16)    startcharoffset;
        unsigned int(16)    endcharoffset;
    }
}
```

5.17.1.4 Scroll Delay

'delay' - Specifies a delay after a Scroll In and/or before Scroll Out. A 32-bit integer specifying the delay, in the units of the timescale of the track. The default delay, in the absence of this box, is 0.

```
class TextScrollDelayBox() extends TextSampleModifierBox ("delay") {
    unsigned int(32)    scroll-delay;
}
```

5.17.1.5 HyperText

'href' – HyperText link. The existence of the hypertext link is visually indicated in a suitable style (e.g. underlined blue text).

This box contains these values:

- startCharOffset: – the start offset of the text to be linked
- endCharOffset: – the end offset of the text (start offset + number of characters)
- URLLength:– the number of bytes in the following URL
- URL: UTF-8 characters – the linked-to URL
- altLength:– the number of bytes in the following 'alt' string
- altstring: UTF-8 characters – an 'alt' string for user display

The URL should be an absolute URL, as the context for a relative URL may not always be clear.

The 'alt' string may be used as a tool-tip or other visual clue, as a substitute for the URL, if desired by the terminal, to display to the user as a hint on where the link refers.

Hypertext-linked text should not be scrolled; not all terminals can display this or manage the user interaction to determine whether user has interacted with moving text. It is also hard for the user to interact with scrolling text.

```
class TextHyperTextBox() extends TextSampleModifierBox ("href") {
    unsigned int(16)    startcharoffset;
    unsigned int(16)    endcharoffset;
    unsigned int(8)    URLLength;
    unsigned int(8)    URL[URLLength];
    unsigned int(8)    altLength;
    unsigned int(8)    altstring[altLength];
}
```


5.17.1.6 Textbox

"tbox" – text box over-ride. This over-rides the default text box set in the sample description.

```
class TextboxBox() extends TextSampleModifierBox ('tbox') {
    BoxRecord    text-box;
}
```

5.17.1.7 Blink

"blnk" – Blinking text. This requests blinking text for the indicated character range. Terminals are not required to support blinking text, and the precise way in which blinking is achieved, and its rate, is terminal-dependent.

```
class BlinkBox() extends TextSampleModifierBox ('blnk') {
    unsigned int(16)    startcharoffset;
    unsigned int(16)    endcharoffset;
}
```

5.17.1.8 Text Wrap Indication

'twrp' - Specifies text wrap behavior: the Box contains one 8-bit integer as a wrap mode flag.

```
class TextWrapBox() extends TextSampleModifierBox ("twrp") {
    unsigned int(8) wrap_flag;
}
```

wrap_flag: a value from table 5.1

Table 5.1: Wrap Flag Values

Value	Description
0x00	No wrap
0x01	Automatic "soft" wrap enabled
0x02-0xFF	Reserved

5.18 Combinations of features

Two modifier boxes of the same type shall not be applied to the same character (e.g. it is not permitted to have two href links from the same text). As the "hclr", "dlay" and "tbox" are globally applied to the whole text in a sample, each sample shall contain at most one "hclr", at most one "dlay", and at most one "tbox" modifier.

Table 5.2 details the effects of multiple options:

Table 5.2: Combinations of features

		Sample description style record	First sample modifier box				
			styl	hlit	krok	href	blnk
Second sample modifier box	styl	1	3				
	hlit			3			
	krok			4	3		
	href	2			5	3	
	blnk						6

1. The sample description provides the default style; the style records over-ride this for the selected characters.
2. The terminal over-rides the chosen style for HREF links.
3. Two records of the same type cannot be applied to the same character.
4. Dynamic and static highlighting must not be applied to the same text.
5. Dynamic highlighting and linking must not be applied to the same text.
6. Blinking text is optional, particularly when requested in combination with other features.

Annex A (informative): Change history

Change history							
Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	Old	New
2004-06	24	SP-040344			Approved at 3GPP TSG SA#24	2.0.0	6.0.0
2004-12	26	SP-040839	001	1	Removal of incorrect statement in Scope section of Rel-6 Timed Text	6.0.0	6.1.0
2007-06	36				Version for Release 7	6.1.0	7.0.0
2008-12	42				Version for Release 8	7.0.0	8.0.0

History

Document history		
V8.0.0	January 2009	Publication