Technical Specification Group Services and System Aspects ***TSGS#15(02)0088***
Meeting #15, Cheju Island, Korea, 11-14 March 2002

Source: **TSG-SA WG4**

Title: CR to TS 26.234 on " Addition of Release 5 functionality " (Release 5)

Document for: **Approval**

Agenda Item: **7.4.3**

The following CR, agreed at the TSG-SA WG4 meeting #20, is presented to TSG SA #15 for approval.

Spec	CR	Rev	Phase	Subject	Cat	Vers	WG	Meeting	S4 doc
26.234	022	2	REL-5	Addition of Release 5 functionality	B	4.2.0	S4	TSG-SA WG4#20	S4-020226

3GPP TSG-SA4 Meeting #20
Luleå, Sweden, 18-22 February 2002

Tdoc S4-020226

CR-Form-v5

CHANGE REQUEST

⌘	**26.234**	**CR**	**022**	⌘ **rev**	**2**	⌘ Current version:	**4.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE [✓ X] Radio Access Network ☐ Core Network ☐

Title:	⌘	Addition of Release 5 functionality		
Source:	⌘	TSG SA WG4		
Work item code:	⌘	PSS-E	***Date:*** ⌘	11 March 2002
Category:	⌘	**B**	***Release:*** ⌘	REL-5

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change: ⌘ Release 5 functionality added.

Summary of change: ⌘ A number a new features and functions have been included in R5, such as:

- New media types has been addressed (Vector Graphics, Synthetic Audio, Timed Text)
- Existing media types have been extended (PNG, XHTML mobile profile)
- Capability exchange have been added
- An optional video buffer model has been added
- A new module has been added to SMIL PSS Profile (Transistions Effects Module)
- Clarifications and updates to a number of protocols have been done (SDP, RTP, RTSP)
- Alignment with the upcoming ISO file format has been done.

Consequences if not approved: ⌘ Release 5 will NOT include an updated Streaming feature.

Clauses affected: ⌘ All clauses

Other specs affected: ⌘ ☐ Other core specifications ⌘
☐ Test specifications
☐ O&M Specifications

Other comments: ⌘

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

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 specification;

The 3GPP transparent end-to-end packet-switched streaming service (PSS) specification consists ~~of two 3G TSs;~~of three 3G TSs; 3GPP TS 22.233 [1], 3GPP TS 26.233 [2] and the present document. The first TS contains the service requirements for the PSS, the second TS provides an overview of the 3GPP PSS and the present document the details of protocol and codecs used by the service.

Introduction

Streaming refers to the ability of an application to play synchronised media streams like audio and video streams in a continuous way while those streams are being transmitted to the client over a data network.

Applications, which can be built on top of streaming services, can be classified into on-demand and live information delivery applications. Examples of the first category are music and news-on-demand applications. Live delivery of radio and television programs are examples of the second category.

The 3GPP PSS provides a framework for Internet Protocol (IP) based streaming applications in 3G networks.

1 Scope

The present document specifies the protocols and codecs for the PSS within the 3GPP system. Protocols for control signalling, capability exchange, scene description, media transport and media encapsulations are specified. Codecs for speech, natural and synthetic audio, video, still images, bitmap graphics, vector graphics, timed text and text are specified.

The present document is applicable to IP based packet switched networks.

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 ".~~(void)~~

[2] 3GPP TS 26.233: " Transparent end-to-end packet switched~~End-to-end transparent~~ streaming service (PSS); General description".

[3] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[4] IETF RFC 1738: "Uniform Resource Locators (URL)", Berners-Lee, Masinter & McCahill, December 1994.

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

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

[7] IETF STD 0006: "User Datagram Protocol", Postel J., August 1980.

[8] IETF STD 0007: "Transmission Control Protocol", Postel J., September 1981.

[9] IETF RFC 1889: "RTP: A Transport Protocol for Real-Time Applications", Schulzrinne H. et al., January 1996.

[10] IETF RFC 1890: "RTP Profile for Audio and Video Conferences with Minimal Control", Schulzrinne H. et al., January 1996.

[11] 3GPP TS 26.234: "Transparent end-to-end packet switched streaming service (PSS); Protocols and codecs; Annex E: RTP payload format and file storage format for AMR and AMR-WB audio".

[12] void

[13] IETF RFC 3016: "RTP Payload Format for MPEG-4 Audio/Visual Streams", Kikuchi Y. et al., November 2000.

[14] IETF RFC 2429: "RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H.263+)", Bormann C. et al., October 1998.

[15] IETF RFC 2046: "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed, N. Borenstein, November 1996.

[16] IETF RFC 3236: " The 'application/xhtml+xml' Media Type ", Baker M. and Stark P., January 2002.

[17] IETF RFC 2616: "Hypertext Transfer Protocol - HTTP/1.1", Fielding R. et al., June 1999.

[18] 3GPP TS 26.071: "Mandatory Speech Codec speech processing functions; AMR Speech Codec; General description".

[19] 3GPP TS 26.101: "Mandatory Speech Codec speech processing functions; AMR Speech Codec; Frame Structure".

[20] 3GPP TS 26.171: "AMR speech codec, wideband; General description".

[21] ISO/IEC 14496-3:2001, "Information technology -- Coding of audio-visual objects -- Part 3: Audio".

[22] ITU-T Recommendation H.263: "Video coding for low bit rate communication".

[23] ITU-T Recommendation H.263~~ (annex X)~~: "Annex X, Profiles and levels definition".

[24] ISO/IEC 14496-2:2001, "Information technology -- Coding of audio-visual objects -- Part 2: Visual".

[25] ISO/IEC 14496-2:2001/Amd 2:2002, "Streaming video profile".

[26] ITU-T Recommendation T.81 (1991) | ISO/IEC 10918-1 (1992): "Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines.

[27] "JPEG File Interchange Format", Version 1.02, September 1, 1992.

[28] W3C Recommendation: "XHTML Basic", http://www.w3.org/TR/2000/REC-xhtml-basic-20001219, December 2000

[29] ISO/IEC 10646-1 (2000): "Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane".

[30] The Unicode Consortium: "The Unicode Standard", Version 3.0 Reading, MA, Addison-Wesley Developers Press, 2000, ISBN 0-201-61633-5.

[31] W3C Recommendation: "Synchronized Multimedia Integration Language (SMIL 2.0)", http://www.w3.org/TR/2001/REC-smil20-20010807/, August 2001.

[32] CompuServe Incorporated: "GIF Graphics Interchange Format: A Standard defining a mechanism for the storage and transmission of raster-based graphics information", Columbus, OH, USA, 1987.

[33] CompuServe Incorporated: "Graphics Interchange Format: Version 89a", Columbus, OH, USA, 1990.

[34] ISO/IEC 14496-1 (2001): "Information technology - Coding of audio-visual objects - Part 1: Systems".

[35] 3GPP TS 26.234~~23.140~~: "Multimedia Messaging Service (MMS); Media formats and codecs".~~"Multimedia Messaging Service (MMS), Functional description stage 2/3".~~

[36] ISO/IEC 15444-1 (2000): "Information technology - JPEG 2000 image coding system: Core coding system; Annex I: The JPEG 2000 file format".

[37] 3GPP TS 26.201: "AMR Wideband Speech Codec; Frame Structure".

[38] IETF RFC 2083: "PNG (Portable Networks Graphics) Specification version 1.0 ", T. Boutell, et. al., March 1997.

[39] W3C Working Draft Recommendation: "CC/PP structure and vocabularies", http://www.w3.org/Mobile/CCPP/Group/Drafts/WD-CCPP-struct-vocab-20010620/, June 2001.

[40] WAP UAProf Specification, http://www1.wapforum.org/tech/documents/WAP-248-UAProf-20011020-a.pdf , October 2001.

[41] W3C Candidate Recommendation: "Resource Description Framework (RDF) Schema Specification 1.0", http://www.w3.org/TR/2000/CR-rdf-schema-20000327, March 2000.

[42] W3C Working Draft Recommendation: "Scalable Vector Graphics (SVG) 1.1 Specification", http://www.w3.org/TR/SVG11 , February 2002.

[43] W3C Working Draft Recommendation: "SVG Mobile Specification", http://www.w3.org/TR/SVGMobile/, February 2002

[44] Scalable Polyphony MIDI Specification Version 1.0, RP-34, MIDI Manufacturers Association, Los Angeles, CA, February 2002.

[45] Scalable Polyphony MIDI Device 5-to-24 Note Profile for 3GPP Version 1.0, RP-35, MIDI Manufacturers Association, Los Angeles, CA, February 2002.

[46] "Standard MIDI Files 1.0", RP-001, in "The Complete MIDI 1.0 Detailed Specification, Document Version 96.1 " The MIDI Manufacturers Association, Los Angeles, CA, USA, February 1996.

[47] WAP Forum Specification: "XHTML Mobile Profile", http://www1.wapforum.org/tech/terms.asp?doc=WAP-277-XHTMLMP-20011029-a.pdf, October 2001.

[48] "Unicode Standard Annex #13: Unicode Newline Guidelines", by Mark Davis. An integral part of The Unicode Standard, Version 3.1.

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~~,~~ I~~i~~n the present document speech, audio, ~~and ~~video and timed text

discrete media: media that itself does not contain an element of time.~~,~~ I~~i~~n the present document all media not defined as continuous media

device capability description: a description of device capabilities and/or user preferences. Contains a number of capability attributes.

device capability profile: same as device capability description

presentation description: contains information about one or more media streams within a presentation, such as the set of encodings, network addresses and information about the content

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

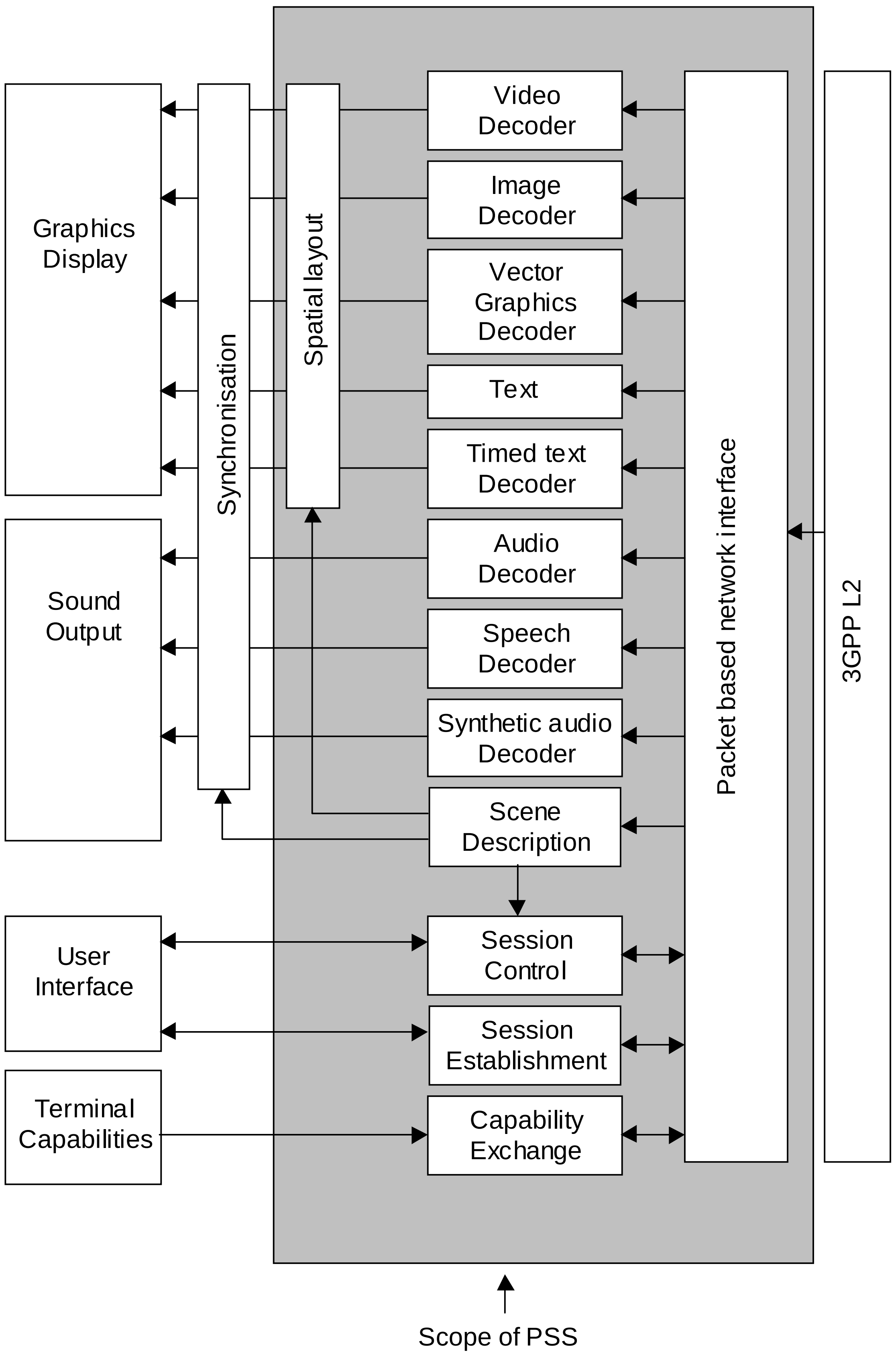
scene description: description of the spatial layout and temporal behaviour of a presentation.~~,~~ I~~i~~t can also contain hyperlinks

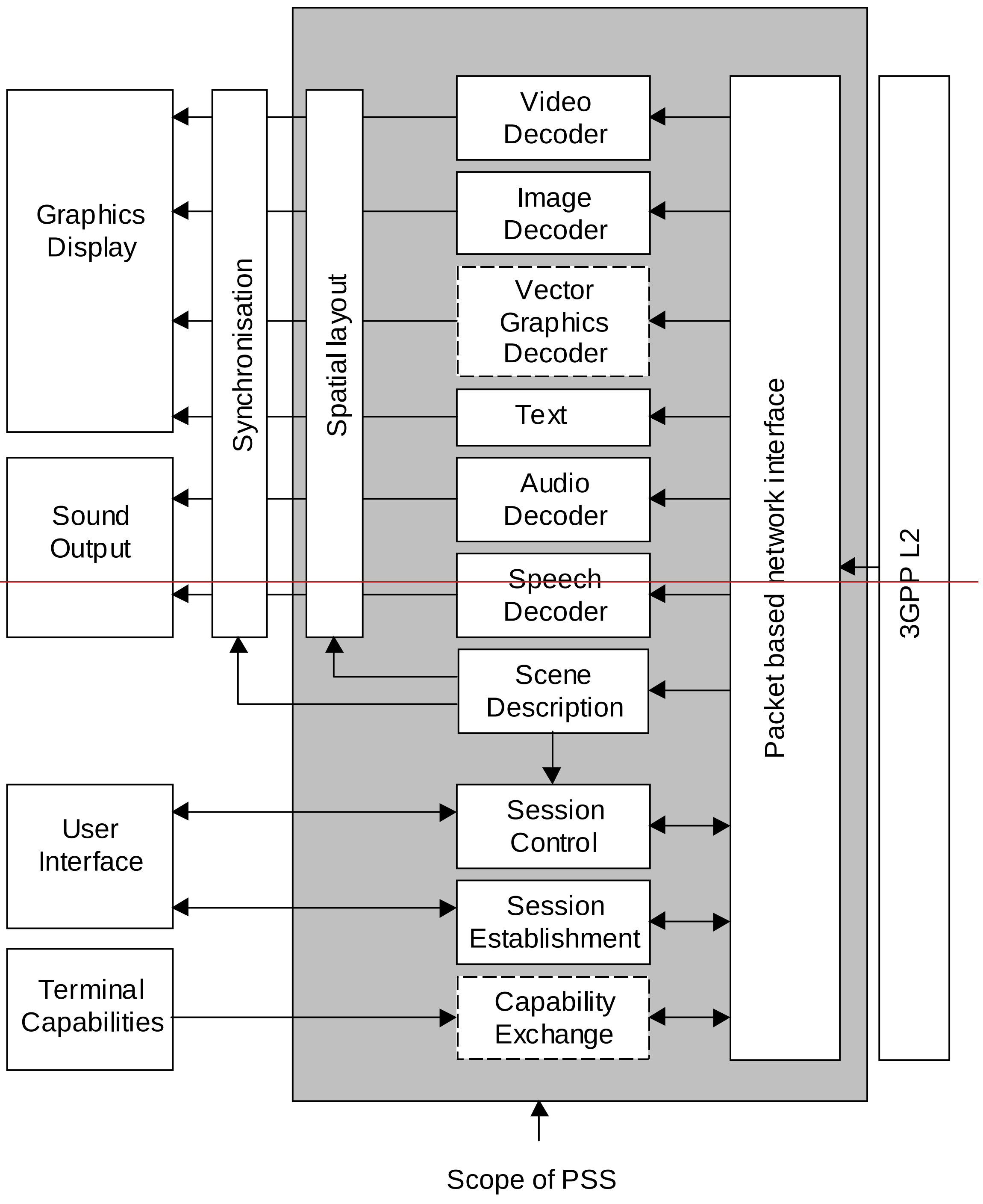
3.2 Abbreviations

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

AAC	Advanced Audio Coding
BIFS	Binary Format for Scenes ~~description~~
CC/PP	Composite Capability / Preference Profiles
DCT	Discrete Cosine Transform
GIF	Graphics Interchange Format
HTML	Hyper Text Markup Language
ITU-T	International Telecommunications Union – Telecommunications
JFIF	JPEG File Interchange Format
MIDI	Musical Instrument Digital Interface
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging Service
MP4	MPEG-4 file format
PNG	Portable Networks Graphics
PSS	Packet-switched Streaming Service
QCIF	Quarter Common Intermediate Format
RDF	Resource Description Framework
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real-Time Streaming Protocol
SDP	Session Description Protocol
SMIL	Synchronised Multimedia Integration Language
SP-MIDI	Scalable Polyphony MIDI
SVG	Scalable Vector Graphics
UAProf	User Agent Profile
UCS-2	Universal Character Set (the two octet form)
UTF-8	Unicode Transformation Format (the 8-bit form)
UTF-16	Unicode Transformation Format (the 16-bit form)
W3C	WWW Consortium
WML	Wireless Markup Language
XHTML	eXtensible Hyper Text Markup Language
XML	eXtensible Markup Language

4 System description


Graphics Display
Sound Output
User Interface
Terminal Capabilities
Synchronisation
Spatial layout
Video Decoder
Image Decoder
Vector Graphics Decoder
Text
Timed text Decoder
Audio Decoder
Speech Decoder
Synthetic audio Decoder
Scene Description
Session Control
Session Establishment
Capability Exchange
Packet based network interface
3GPP L2
Scope of PSS



~~NOTE: Dashed components are not specified for the simple PSS.~~

Figure 1: Functional components of a PSS client

Figure 1 shows the functional components of a PSS client. Figure 2 gives an overview of the protocol stack used in a PSS client and also shows a more detailed view of the packet based network interface. The functional components can be divided into control, scene description, media codecs and the transport of media and control data. ~~TS 26.233 [2] defines the simple and extended PSS. Dashed functional components in figure 1 are not specified for the simple PSS.~~

The control related elements are session establishment, capability exchange and session control (see clause 5).

- Session establishment refers to methods to invoke a PSS session from a browser or directly by entering an URL in the terminal's user interface.

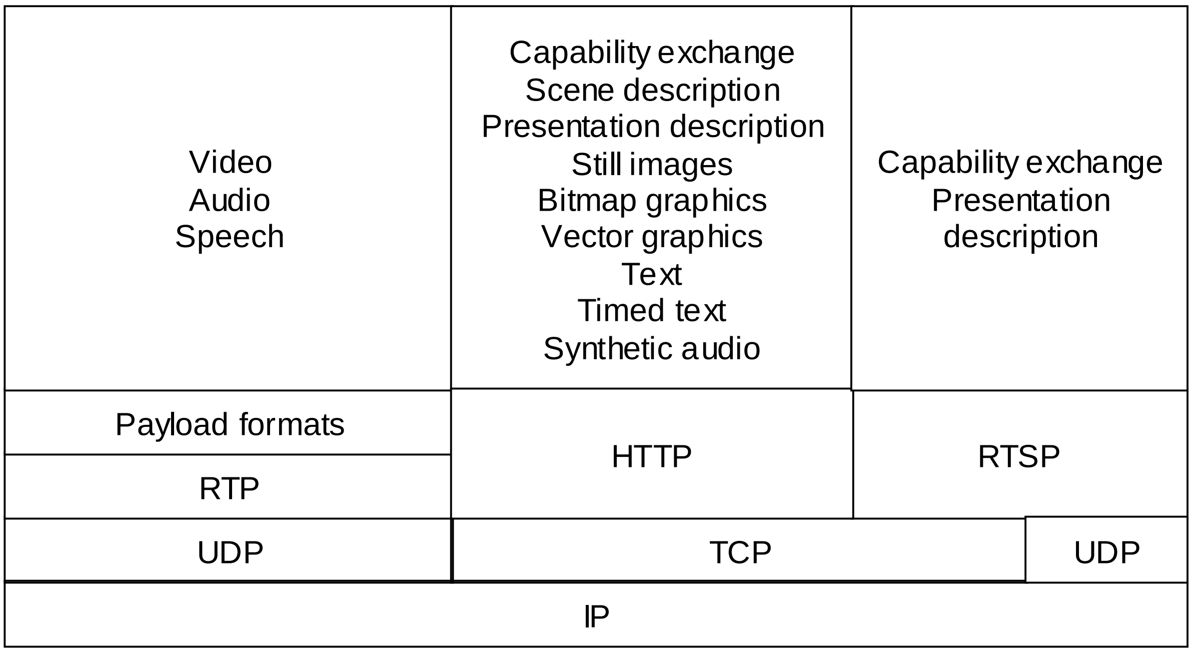
- Capability exchange enables choice or adaptation of media streams depending on different terminal capabilities.

- Session control deals with the set-up of the individual media streams between a PSS client and one or several PSS servers. It also enables control of the individual media streams by the user. It may involve VCR-like presentation control functions like start, pause, fast forward and stop of a media presentation.

The scene description consists of spatial layout and a description of the temporal relation between different media that is included in the media presentation. The first gives the layout of different media components on the screen and the latter controls the synchronisation of the different media (see clause 8).

The PSS includes media codecs for video, still images, vector graphics, bitmap graphics, text, timed text, natural and synthetic audio, and speech (see clause 7).

Transport of media and control data consists of the encapsulation of the coded media and control data in a transport protocol (see clause 6). This is shown in figure 1 as the "packet based network interface" and displayed in more detail in the protocol stack of figure 2.



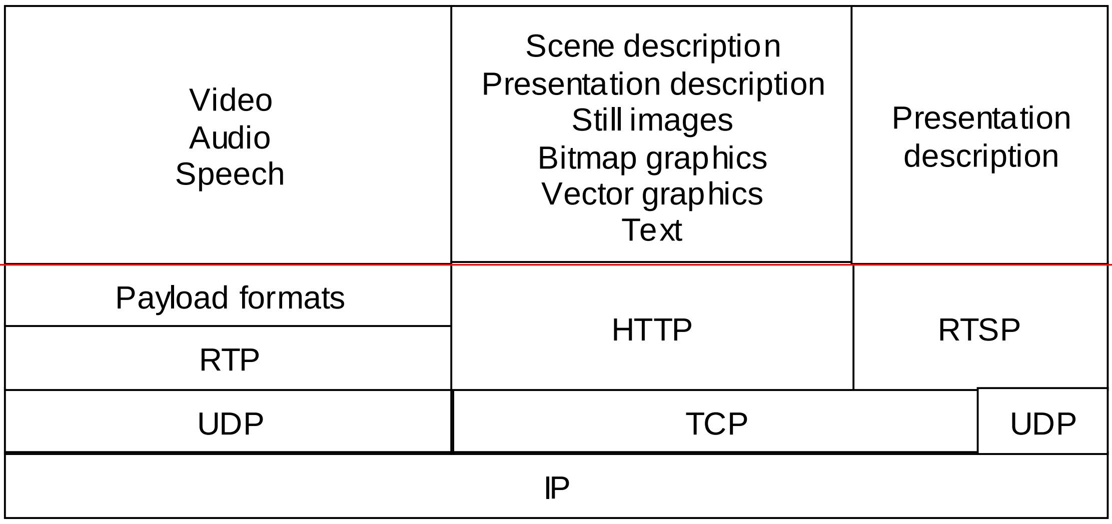


Figure 2: Overview of the protocol stack

5 Protocols

5.1 Session establishment

Session establishment refers to the method by which a PSS client obtains the initial session description. The initial session description can e.g. be a presentation description, a scene description or just an URL to the content.

A PSS client shall support initial session descriptions specified in one of the following formats: SMIL, SDP, or plain RTSP URL.

In addition to rtsp:// the PSS client shall support URLs [4] to valid initial session descriptions starting with file:// (for locally stored files) and http:// (for presentation descriptions or scene descriptions delivered via HTTP).

Examples for valid inputs to a PSS client are: file://temp/morning_news.smil, http://mediaportal/morning_news.sdp, and rtsp://mediaportal/morning_news.

URLs can be made available to a PSS client in many different ways. It is out of the scope of this recommendation to mandate any specific mechanism. However, an application using the 3GPP PSS shall at least support URLs of the above type, specified or selected by the user.

The preferred way would be to embed URLs to initial session descriptions within HTML or WML pages. Browser applications that support the HTTP protocol could then download the initial session description and pass the content to the PSS client for further processing. How exactly this is done is an implementation specific issue and out of the scope of this recommendation.

5.2 Capability exchange

5.2.1 General

Capability exchange is an important functionality in the PSS. It enables PSS servers to provide a wide range of devices with content suitable for the particular device in question. Another very important task is to provide a smooth transition between different releases of PSS. Therefore, PSS clients and servers should support capability exchange.

The specification of capability exchange for PSS is divided into two parts. The normative part contained in clause 5.2 and an informative part in clause A.4 in Annex A of the present document. The normative part gives all the necessary requirements that a client or server shall conform to when implementing capability exchange in the PSS. The informative part provides additional important information for understanding the concept and usage of the functionality. It is recommended to read clause A.4 in Annex A before continuing with clauses 5.2.2-5.2.7.

5.2.2 The device capability profile structure

A device capability profile is a RDF [41] document that follows the structure of the CC/PP framework [39] and the CC/PP application UAProf [40]. Attributes are used to specify device capabilities and preferences. A set of attribute names, permissible values and semantics constitute a CC/PP vocabulary, which is defined by a RDF schema. For PSS the UAProf vocabulary is reused and an additional PSS specific vocabulary is defined. The details can be found in clause 5.2.3. The syntax of the attributes is defined in the vocabulary schema but also, to some extent, the semantics. A PSS device capability profile is an instance of the schema (UAProf and/or the PSS specific schema) and shall follow the rules governing the formation of a profile given in the CC/PP specification [39]. The profile schema shall also be governed by the rules defined in UAProf [40] chapter 7, 7.1, 7.3 and 7.4.

5.2.3 Vocabularies for PSS

5.2.3.1 General

Clause 5.2.3 specifies the attribute vocabularies to be used by the PSS capability exchange.

PSS servers should understand the attributes in both the streaming component of the PSS base vocabulary and the recommended attributes from the UAProf vocabulary [40]. A server may additionally support other UAProf attributes.

5.2.3.2 PSS base vocabulary

The PSS base vocabulary contains one component called "Streaming". A vocabulary extension to UAProf shall be defined as a RDF schema. This schema can be found in Annex F. The schema together with the description of the attributes in the present clause, defines the vocabulary. The vocabulary is associated with an XML namespace, which combines a base URI with a local XML element name to yield a URI. Annex F provides the details.

All PSS attributes are put in a PSS specific component called “Streaming”. The list of PSS attributes is as follows:

Attribute name:	**AudioChannels**
Attribute definition:	This attribute describes the stereophonic capability of the natural audio device.
Component:	Streaming

Type: Literal

Legal values: "Mono", "Stereo"

Resolution rule: Locked

EXAMPLE 1: `<AudioChannels>Mono</AudioChannels>`

Attribute name: **MaxPolyphony**

Attribute definition: The MaxPolyphony attribute refers to the maximal polyphony that the synthetic audio device supports as defined in [44].

NOTE: MaxPolyphony attribute can be used to signal the maximum polyphony capabilities supported by the PSS client. This is a complementary mechanism for the delivery of compatible SP-MIDI content and thus the PSS client is required to support Scalable Polyphony MIDI i.e. Channel Masking defined in [44].

Component: Streaming

Type: Number

Legal values: Integer between 5 and 24

Resolution rule: Locked

EXAMPLE 2: `<MaxPolyphony>8</MaxPolyphony>`

Attribute name: **PssAccept**

Attribute definition: List of content types (MIME types) the PSS application supports. Both CcppAccept (SoftwarePlatform, UAProf) and PssAccept can be used but if PssAccept is defined it has precedence over CcppAccept.

Component: Streaming

Type: Literal (Bag)

Legal values: List of MIME types with related parameters.

Resolution rule: Append

EXAMPLE 3:
```
<PssAccept>
  <rdf:Bag>
    <rdf:li>audio/AMR-WB; octet-alignment</rdf:li>
    <rdf:li>application/smil</rdf:li>
  </rdf:Bag>
</PssAccept>
```

Attribute name: **PssAccept-Subset**

Attribute definition: List of content types for which the PSS application supports a subset. MIME-types can in most cases effectively be used to express variations in support for different media types. Many MIME-types, e.g. AMR-NB has several parameters that can be used for this purpose. There may exist content types for which the PSS application only supports a subset and this subset can not be expressed with MIME-type parameters. In these cases the attribute PssAccept-Subset is used to describe support for a subset of a specific content type. If a subset of a specific content type is declared in PssAccept-Subset, this means that PssAccept-Subset has precedence over both PssAccept and CcppAccept. PssAccept and/or CcppAccept shall always include the corresponding content types for which PSSAccept-Subset specifies subsets of.

This is to ensure compatibility with those content servers that do not understand the PssAccept-Subset attribute but do understand e.g. CcppAccept.

This is illustrated with an example. If PssAccept="audio/AMR", "image/jpeg" and PssAccept-Subset="JPEG-PSS" then "audio/AMR" and JPEG Base line are supported. "image/jpeg" in PssAccept is of no importance since it is related to "JPEG-PSS" in PssAccept-Subset. Subset identifiers and corresponding semantics shall only be defined by the TSG responsible for the present document. The following values are defined:

- "JPEG-PSS": Only the two JPEG modes described in clause 7.5 of the present document are supported.
- "SVG-Tiny"
- "SVG-Basic"

Component: Streaming

Type: Literal (Bag)

Legal values: "JPEG-PSS", "SVG-Tiny", "SVG-Basic"

Resolution rule: Append

EXAMPLE 4:
```
<PssAccept-Subset>
  <rdf:Bag>
    <rdf:li>JPEG-PSS</rdf:li>
  </rdf:Bag>
</PssAccept-Subset>
```

Attribute name: **PssVersion**

Attribute definition: PSS version supported by the client.

Component: Streaming

Type: Literal

Legal values: "3GPP-R4", "3GPP-R5" and so forth.

Resolution rule: Locked

EXAMPLE 5: `<PssVersion>3GPP-R4</PssVersion>`

Attribute name: **RenderingScreenSize**

Attribute definition: The rendering size of the device's screen in unit of pixels. The horizontal size is given followed by the vertical size.

Component: Streaming

Type: Dimension

Legal values: Two integer values equal or greater than zero. A value equal “0x0”means that there exists no possibility to render visual PSS presentations.

Resolution rule: Locked

EXAMPLE 6: `<RenderingScreenSize>70x15</RenderingScreenSize>`

Attribute name: **SmilBaseSet**

Attribute definition: Indicates a base set of SMIL 2.0 modules that the client supports.

Component: Streaming

Type: Literal

Legal values: Pre-defined identifiers. "SMIL-3GPP-R4" indicates all SMIL 2.0 modules required for scene description support according to clause 8 of Release 4 of TS 26.234. "SMIL-3GPP-R5" indicates all SMIL 2.0 modules required for scene description support according to clause 8 of the present document (Release 5 of TS 26.234).

Resolution rule: Locked

EXAMPLE 7: `<SmilBaseSet>SMIL-3GPP-R4</SmilBaseSet>`

Attribute name: **SmilModules**

Attribute definition: This attribute defines a list of SMIL 2.0 modules supported by the client. If the SmilBaseSet is used those modules do not need to be explicitly listed here. In that case only additional module support needs to be listed.

Component: Streaming

Type: Literal (Bag)

Legal values: SMIL 2.0 module names defined in the SMIL 2.0 recommendation [31], section 2.3.3, table 2.

Resolution rule: Append

EXAMPLE 8:
```
<SmilModules>
  <rdf:Bag>
    <rdf:li>BasicTransitions</rdf:li>
    <rdf:li>MulitArcTiming</rdf:li>
  </rdf:Bag>
</SmilModules>
```

Attribute name: **VideoDecodingByteRate**

Attribute definition: If Annex G is not supported, the attribute has no meaning. If Annex G is supported, this attribute defines the peak decoding byte rate the PSS client is able to support. In other words, the PSS client fulfils the requirements given in Annex G with the signalled peak decoding byte rate. The values are given in bytes per second and shall be greater than or equal to 8000. According to Annex G, 8000 is the default peak decoding byte rate for the mandatory video codec profile and level (H.263 Profile 0 Level 10).

Component: Streaming

Type: Number

Legal values: Integer value greater than or equal to 8000.

Resolution rule: Locked

EXAMPLE 9: `<VideoDecodingByteRate>16000</VideoDecodingByteRate>`

Attribute name: **VideoInitialPostDecoderBufferingPeriod**

Attribute definition: If Annex G is not supported, the attribute has no meaning. If Annex G is supported, this attribute defines the maximum initial post-decoder buffering period of video. Values are interpreted as clock ticks of a 90-kHz clock. In other words, the value is incremented by one for each 1/90 000 seconds. For example, the value 9000 corresponds to 1/10 of a second initial post-decoder buffering.

Component: Streaming

Type:	Number
Legal values:	Integer value equal to or greater than zero.
Resolution rule:	Locked
EXAMPLE 10:	`<VideoInitialPostDecoderBufferingPeriod>9000 </VideoInitialPostDecoderBufferingPeriod>`

Attribute name:	**VideoPreDecoderBufferSize**
Attribute definition:	This attribute signals if the optional video buffering requirements defined in Annex G are supported. It also defines the size of the hypothetical pre-decoder buffer defined in Annex G. A value equal to zero means that Annex G is not supported. A value equal to one means that Annex G is supported. In this case the size of the buffer is the default size defined in Annex G. A value equal to or greater than the default buffer size defined in Annex G means that Annex G is supported and sets the buffer size to the given number of octets.
Component:	Streaming
Type:	Number
Legal values:	Integer value equal to or greater than zero. Values greater than one but less than the default buffer size defined in Annex G are not allowed.
Resolution rule:	Locked
EXAMPLE 11:	`<VideoPreDecoderBufferSize>30720</VideoPreDecoderBufferSize>`

5.2.3.3 Attributes from UAProf

In the UAProf vocabulary [40] there are several attributes that are of interest for the PSS. The formal definition of these attributes is given in [40]. The following list of attributes is recommended for PSS applications:

Attribute name:	**BitsPerPixel**
Component:	HardwarePlatform
Attribute description:	The number of bits of colour or greyscale information per pixel
EXAMPLE 1:	`<BitsPerPixel>8</BitsPerPixel>`

Attribute name:	**ColorCapable**
Component:	HardwarePlatform
Attribute description:	Whether the device display supports colour or not.
EXAMPLE 2:	`<ColorCapable>Yes</ColorCapable>`

Attribute name:	**PixelAspectRatio**
Component:	HardwarePlatform
Attribute description:	Ratio of pixel width to pixel height
EXAMPLE 3:	`<PixelAspectRatio>1x2</PixelAspectRatio>`

Attribute name: **PointingResolution**

Component: HardwarePlatform

Attribute description: Type of resolution of the pointing accessory supported by the device.

EXAMPLE 4: `<PointingResolution>Pixel</PointingResolution>`

Attribute name: **Model**

Component: HardwarePlatform

Attribute description: Model number assigned to the terminal device by the vendor or manufacturer

EXAMPLE 5: `<Model>Lexus</Model>`

Attribute name: **Vendor**

Component: HardwarePlatform

Attribute description: Name of the vendor manufacturing the terminal device

EXAMPLE 6: `<Vendor>Toyota</Vendor>`

Attribute name: **CcppAccept-Charset**

Component: SoftwarePlatform

Attribute description: List of character sets the device supports

EXAMPLE 7:
```
<CcppAccept-Charset>
  <rdf:Bag>
    <rdf:li>UTF-8</rdf:li>
  </rdf:Bag>
</CcppAccept-Charset>
```

Attribute name: **CcppAccept-Encoding**

Component: SoftwarePlatform

Attribute description: List of transfer encodings the device supports

EXAMPLE 8:
```
<CcppAccept-Encoding>
  <rdf:Bag>
    <rdf:li>base64</rdf:li>
  </rdf:Bag>
</CcppAccept-Encoding>
```

Attribute name: **CcppAccept-Language**

Component: SoftwarePlatform

Attribute description: List of preferred document languages

EXAMPLE 9:
```
<CcppAccept-Language>
  <rdf:Seq>
    <rdf:li>en</rdf:li>
    <rdf:li>se</rdf:li>
  </rdf:Seq>
</CcppAccept-Language>
```

5.2.4 Extensions to the PSS schema/vocabulary

The use of RDF enables an extensibility mechanism for CC/PP-based schemas that addresses the evolution of new types of devices and applications. The PSS profile schema specification is going to provide a base vocabulary but in the future new usage scenarios might have need for expressing new attributes. If the base vocabulary is updated a new unique namespace will be assigned to the updated schema. The base vocabulary shall only be changed by the TSG responsible for the present document. All extensions to the profile schema shall be governed by the rules defined in [40] clause 7.7.

5.2.5 Signalling of profile information between client and server

When a PSS client or server support capability exchange it shall support the profile information transport over both HTTP and RTSP between client and server as defined in clause 9.1 (including its subsections) of the WAP 2.0 UAProf specification [40] with the following additions:

- The "x-wap-profile" and "x-wap-profile-diff" headers may not be present in all HTTP or RTSP request. That is, the requirement to send this header in all requests has been relaxed.

- The defined headers may be applied to both RTSP and HTTP.

- The "x-wap-profile-diff" header is only valid for the current request. The reason is that PSS does not have the WSP session concept of WAP.

- Push is not relevant for the PSS.

The following recommendations are made to how and when profile information should be sent between client and server:

- PSS content servers supporting capability exchange shall be able to receive profile information in all HTTP and RTSP requests.

- The terminal should not send the "x-wap-profile-diff" header over the air-interface since there is no compression scheme defined.

- RTSP: the client should send profile information in the DESCRIBE message. It may send it in any other request.

If the terminal has some prior knowledge about the file type it is about to retrieve, e.g. file extensions, the following apply:

- HTTP and SDP: when retrieving an SDP with HTTP the client should include profile information in the GET request. This way the HTTP server can deliver an optimised SDP to the client.

- HTTP and SMIL: When retrieving a SMIL file with HTTP the client should include profile information in the GET request. This way the HTTP server can deliver an optimised SMIL presentation to the client. A SMIL presentation can include links to static media. The server should optimise the SMIL file so that links to the referenced static media are adapted to the requesting client. When the "x-wap-profile-warning" indicates that content selection has been applied (201-203) the PSS client should assume that no more capability exchange has to be performed for the static media components. In this case it should not send any profile information when retrieving static media to be included in the SMIL presentation. This will minimise the HTTP header overhead.

5.2.6 Merging device capability profiles

Profiles need to be merged whenever the PSS server receives multiple device capability profiles. Multiple occurrences of attributes and default values make it necessary to resolve the profiles according to a resolution process.

The resolution process shall be the same as defined in UAProf [40] clause 6.4.1.

- Resolve all indirect references by retrieving URI references contained within the profile.
- Resolve each profile and profile-diff document by first applying attribute values contained in the default URI references and by second applying overriding attribute values contained within the category blocks of that profile or profile-diff.
- Determine the final value of the attributes by applying the resolved attribute values from each profile and profile-diff in order, with the attribute values determined by the resolution rules provided in the schema. Where no resolution rules are provided for a particular attribute in the schema, values provided in profiles or profile-diffs are assumed to override values provided in previous profiles or profile-diffs.

When several URLs are defined in the "x-wap-profile" header and there exists any attribute that occurs more than once in these profiles the rule is that the attribute value in the second URL overrides, or is overridden by, or is appended to the attribute value from the first URL (according to the resolution rule) and so forth. This is what is meant with "Determine the final value of the attributes by applying the resolved attribute values from each profile and profile-diff in order, with…" in the third bullet above. If the profile is completely or partly inaccessible or otherwise corrupted the server should still provide content to the client. The server is responsible for delivering content optimised for the client based on the received profile in a best effort manner.

NOTE: For the reasons explained in Annex A clause A.4.3 the usage of indirect references in profiles (using the CC/PP defaults element) is not recommended.

5.2.7 Profile transfer between the PSS server and the device profile server

The device capability profiles are stored on a device profile server and referenced with URLs. According to the profile resolution process in clause 5.2.6 of the present document, the PSS server ends up with a number of URLs referring to profiles and these shall be retrieved.

- The device profile server shall support HTTP 1.1 for the transfer of device capability profiles to the PSS server.
- If the PSS server supports capability exchange it shall support HTTP 1.1 for transfer of device capability profiles from the device profile server. A URL shall be used to identify a device capability profile.

- Normal content caching provisions as defined by HTTP apply.

~~No explicit capability exchange protocol is specified for the simple PSS.. Instead it is assumed that the user is aware of that the content he/she is about to stream fits the capabilities, e.g. screen size, of the particular device used. Protocols for capability exchange can be specified for the extended PSS.~~

5.3 Session set-up and control

5.3.1 General

Continuous media is media that has~~have~~ an intrinsic time line. Discrete media on the other hand does not it~~-~~self contain an element of time. In this specification speech, audio and video belongs to first category and still images and text to the latter one.~~ Bitmap graphics can fall into both groups, but is in this specification defined to be discrete media.~~

Streaming of continuous media using RTP/UDP/IP (see clause 6.2) requires a session control protocol to set-up and control of the individual media streams. For the transport of discrete media (images and text), vector graphics, timed text and synthetic audio this specification adopts the use of HTTP/TCP/IP (see clause 6.3). In this case there is no need for a separate session set-up and control protocol since this is built into HTTP. This clause describes session set-up and control of the continuous media speech, audio and video.

5.3.2 RTSP

RTSP [5] shall be used for session set-up and session control. PSS clients and servers shall follow the rules for minimal on-demand playback RTSP implementations in appendix D of [5]. In addition to this:

- PSS servers and clients shall implement the DESCRIBE method (see clause 10.2 in [5]);
- PSS servers and clients shall implement the Range header field (see clause 12.29 in [5]).

5.3.3 SDP

5.3.3.1 General

RTSP requires a presentation description. SDP shall be used as the format of the presentation description for both PSS clients and servers. PSS servers shall provide and clients interpret the SDP syntax according to the SDP specification [6] and appendix C of [5]. The SDP delivered to the PSS client shall declare the media types to be used in the session using a codec specific MIME media type for each media. MIME media types to be used in the SDP file are described in clause 5.4 of the present document.

The SDP [6] specification requires certain fields to always be included in an SDP file. Apart from this a PSS server shall always include the following fields in the SDP:

- "a=control:" according to clauses C.1.1, C.2 and C.3 in [5];
- "a=range:" according to clause C.1.5 in [5];
- "a=rtpmap:" according to clause 6 in [6];
- "a=fmtp:" according to clause 6 in [6].

The bandwidth field in SDP should~~can~~ be used to indicate to the PSS client the amount of bandwidth that is required for the session and the individual media in the presentation. Therefore, a PSS server should include the "b=AS:" field in the SDP (both on the session and media level) and a PSS client shall be able to interpret this field. For RTP based applications, AS gives the RTP "session bandwidth" (including UDP/IP overhead) as defined in section 6.2 of [9].

NOTE: The SDP parsers and/or interpreters shall be able to accept NULL values in the 'c=' field (e.g. 0.0.0.0 in IPv4 case). This may happen when the media content does not have a fixed destination address. For more details, see Section C.1.7 of [5] and Section 6 of [6].

5.3.3.2 Additional SDP fields

The following Annex G-related media level SDP fields are defined for PSS:

- "a=X-predecbufsize:<size of the hypothetical pre-decoder buffer>"
  This gives the suggested size of the Annex G hypothetical pre-decoder buffer in bytes.
- "a=X-initpredecbufperiod:<initial pre-decoder buffering period>"
  This gives the required initial pre-decoder buffering period specified according to Annex G. Values are interpreted as clock ticks of a 90-kHz clock. That is, the value is incremented by one for each 1/90 000 seconds. For example, value 180 000 corresponds to a two second initial pre-decoder buffering.
- "a=X-initpostdecbufperiod:<initial post-decoder buffering period>"
  This gives the required initial post-decoder buffering period specified according to Annex G. Values are interpreted as clock ticks of a 90-kHz clock.
- "a=X-decbyterate:<peak decoding byte rate>"
  This gives the peak decoding byte rate that was used to verify the compatibility of the stream with Annex G. Values are given in bytes per second.

If none of the attributes "a=X-predecbufsize:", "a=X-initpredecbufperiod:", "a=X-initpostdecbufperiod:", and "a=x-decbyterate:" is present, clients should not expect a packet stream according to Annex G. If at least one of the listed attributes is present, the transmitted video packet stream shall conform to Annex G. If at least one of the listed attributes is present, but some of the listed attributes are missing in an SDP description, clients should expect a default value for the missing attributes according to Annex G.

5.4 MIME media types

For continuous media (speech, audio and video) the following MIME media types shall be used:

- AMR narrow-~~-~~band speech codec (see clause 7.2) MIME media type as defined in [11];
- AMR wide~~-~~band speech codec (see clause 7.2) MIME media type as defined in [12];
- MPEG-4 AAC audio codec (see clause 7.3) MIME media type as defined in RFC 3016 [13].
- MPEG-4 video codec (see clause 7.4) MIME media type as defined in RFC 3016 [13];
- H.263 [22] video codec (see clause 7.4) MIME media type as defined in annex C, clause C.1 of the present document.

MIME media types for JPEG, GIF, PNG, SP-MIDI, SVG, timed text and XHTML can be used both in the "Content-type" field in HTTP and in the "type" attribute in SMIL 2.0. The following MIME media types shall be used for these media:

- JPEG (see clause 7.5) MIME media type as defined in [15];
- GIF (see clause 7.6) MIME media type as defined in [15];
- PNG (see sub clause 7.6) MIME media type as defined in [38];
- SP-MIDI (see sub clause 7.3A) MIME media type as defined in clause C.2 in Annex C of the present document;
- SVG (see sub clause 7.7) MIME media type as defined in [42];
- XHTML (see clause 7.8) MIME media type as defined in [16].
- Timed text (see subclause 7.9) MIME media type as defined in clause D.9 in Annex D of the present document.

MIME media type used for SMIL files shall be according to [31] and for SDP files according to [6].

6 Data transport

6.1 Packet based network interface

PSS clients and servers shall support an IP-based network interface for the transport of session control and media data. Control and media data are sent using TCP/IP [8] and UDP/IP [7]. An overview of the protocol stack can be found in figure 2 of the present document.

6.2 RTP over UDP/IP

The IETF RTP [9] and [10] provides ~~a~~ means for sending real-time or streaming data over UDP (see [7]). The encoded media is encapsulated in the RTP packets with media specific RTP payload formats. RTP payload formats are defined by IETF. RTP also provides a protocol called RTCP (see clause 6 in [9]) for feedback about the transmission quality. For the calculation of the RTCP transmission interval Annex A.7 in [9] shall be used. Clause A.3.2.3 in Annex A of the present document provides more information about the minimum RTCP transmission interval.

RTP/UDP/IP transport of continuous media (speech ,audio and video) shall be supported.

For RTP/UDP/IP transport of continuous media the following RTP payload formats shall be used:

- AMR narrow-~~-~~band speech codec (see clause 7.2) RTP payload format according to [11]. A PSS client is not required to support multi-channel sessions;
- AMR wide~~-~~band speech codec (see clause 7.2) RTP payload format according to [11]. A PSS client is not required to support multi-channel sessions;
- MPEG-4 AAC audio codec (see clause 7.3) RTP payload format according to RFC 3016 [13];
- MPEG-4 video codec (see clause 7.4) RTP payload format according to RFC 3016 [13];
- H.263~~ [22]~~ video codec (see clause 7.4) RTP payload format according to RFC 2429 [14];

NOTE: The payload format RFC 3016 for MPEG-4 AAC specify that the audio streams shall be formatted by the LATM (Low-overhead MPEG-4 Audio Transport Multiplex) tool [21]. It should be noted that the references for the LATM format in the RFC 3016 [13] point to an older version of the LATM format than included in [21]. In [21] a corrigendum to the LATM tool is included. This corrigendum includes changes to the LATM format making implementations using the corrigendum incompatible with implementations not using it. To avoid future interoperability problems, implementations of PSS client and servers supporting AAC shall follow the changes to the LATM format included in [21].

6.3 HTTP over TCP/IP

The IETF TCP provides reliable transport of data over IP networks, but with no delay guarantees. It is the preferred way for sending the scene description, text, bitmap graphics and still images. There is also need for an application protocol to control the transfer. The IETF HTTP [17] provides this functionality.

HTTP/TCP/IP transport shall be supported for:

- still images (see clause 7.5);
- bitmap graphics (see clause 7.6);
- synthetic audio (see clause 7.3A);
- vector graphics (see clause 7.7);
- text (see clause 7.8);
- timed text (see clause 7.9);
- scene description (see clause 8);
- presentation description (see clause 5.3.3).

6.4 Transport of RTSP

Transport of RTSP shall be supported according to RFC 2326 [5].

7 Codecs

7.1 General

For PSS offering a particular media type, media decoders are specified in the following clauses.

7.2 Speech

The AMR decoder shall be supported for narrow-band speech [18]. The AMR wideband speech decoder [20] shall be supported when wideband speech working at 16 kHz sampling frequency is supported.

7.3 Audio

MPEG-4 AAC Low Complexity object type decoder [21] should be supported. The maximum sampling rate to be supported by the decoder is 48 kHz. The channel configurations to be supported are mono (1/0) and stereo (2/0). In addition, the MPEG-4 AAC Long Term Prediction object type decoder may be supported.

7.3A Synthetic audio

The Scalable Polyphony MIDI (SP-MIDI) content format defined in Scalable Polyphony MIDI Specification [44] and the device requirements defined in Scalable Polyphony MIDI Device 5-to-24 Note Profile for 3GPP [45] should be supported.

SP-MIDI content is delivered in the structure specified in Standard MIDI Files 1.0 [46], either in format 0 or format 1.

7.4 Video

ITU-T Recommendation H.263 [22] profile 0 level 10 shall be supported. This is the mandatory video decoder for the PSS. In addition, PSS should support:

- H.263 [23] Profile 3 Level 10 decoder;
- MPEG-4 Visual Simple Profile Level 0 decoder, [24] and [25].

These two video decoders are optional to implement.

An optional video buffer model is given in Annex G of the present document.

NOTE: ITU-T Recommendation H.263 [22] baseline has been mandated to ensure that video-enabled PSS support a minimum baseline video capability and interoperability can be guaranteed (an H.263 [22] baseline bitstream can be decoded by both H.263 [22] and MPEG-4 decoders). It also provides a simple upgrade path for mandating more advanced decoders in the future (from both the ITU-T and ISO MPEG).

7.5 Still images

ISO/IEC JPEG [26] together with JFIF [27] decoders shall be supported. The support for ISO/IEC JPEG only apply to the following two modes:

- baseline DCT, non-differential, Huffman coding, as defined in table B.1, symbol 'SOF0' in [26];
- progressive DCT, non-differential, Huffman coding, as defined in table B.1, symbol 'SOF2' [26].

7.6 Bitmap graphics

The following bitmap graphics decoders should be supported:

- GIF87a, [32];
- GIF89a, [33];
- PNG, [38].

7.7 Vector graphics

The SVG Tiny profile [42] [43] shall be supported. In addition SVG Basic profile [42] [43] may be supported.~~No vector graphics decoder is specified for the simple PSS. For the extended PSS mandatory and/or optional vector graphics decoders can be specified.~~

7.8 Text

The text decoder is intended to enable formatted text in a SMIL presentation. A PSS client shall support

- text formatted according to XHTML Mobile Profile [47];~~Basic [28];~~
- rendering a SMIL presentation where text is referenced with the SMIL 2.0 "text" element together with the SMIL 2.0 "src" attribute.

The following character coding formats shall be supported:

- UTF-8, [30~~29~~];
- UCS-2, [29~~30~~].

NOTE: Since both SMIL and XHTML are XML based languages it would be possible to define a SMIL plus XHTML profile. In contrast to the present defined ~~PSS4~~PSS SMIL Language Profile that only contain SMIL modules, such a profile would also contain XHTML modules. No combined SMIL and XHTML profile is specified for PSS. Rendering of such documents is out of the scope of the present document.

7.9 Timed text

PSS clients shall support timed text as defined in Annex D, clause D.8a, of this specification. There is no support for RTP transport of timed text in this release; 3GPP (MP4) files containing timed text may only be downloaded.

NOTE: When a PSS client supports timed text it needs to be able to receive and parse 3GPP (MP4) files containing the text streams. This does not imply a requirement on PSS clients to be able to render other continuous media types contained in 3GPP (MP4) files, e.g. AMR and H.263,if such media types are included in a presentation together with timed text. Audio and video are instead streamed to the client using RTSP/RTP (see clause 6.2).

8 Scene description

8.1 General

The 3GPP PSS uses a subset of SMIL 2.0 [31] as format of the scene description. PSS clients and servers with support for scene descriptions shall support the 3GPP ~~PSS4~~PSS SMIL Language Profile defined in clause 8.2 (abbreviated 3GPP PSS SMIL). This profile is a subset of the SMIL 2.0 Language Profile, but a superset of the SMIL 2.0 Basic Language Profile. The present document also includes an informative Annex B that provides guidelines for SMIL content authors.

NOTE: The interpretation of this is not that all streaming sessions are required to use SMIL. For some types of sessions, e.g. consisting of one single continuous media or two media synchronised by using RTP timestamps, SMIL may not be needed.

8.2 3GPP ~~PSS4~~PSS SMIL Language Profile

8.2.1 Introduction

3GPP ~~PSS4~~PSS SMIL is a markup language based on SMIL Basic [31] and SMIL Scalability Framework.

3GPP ~~PSS4~~PSS SMIL ~~shall consist~~consits of the modules required by SMIL Basic Profile (and SMIL 2.0 Host Language Conformance) and additional MediaAccessibility, MediaDescription, MediaClipping, MetaInformation, PrefetchControl,~~ and~~ EventTiming and BasicTransitions modules. All of~~in all~~ the following modules are included:

- SMIL 2.0 Content Control Modules --~~–~~ BasicContentControl, SkipContentControl and PrefetchControl
- SMIL 2.0 Layout Module -- BasicLayout

- SMIL 2.0 Linking Module -- BasicLinking
- SMIL 2.0 Media Object Modules – BasicMedia, MediaClipping, MediaAccessibility and MediaDescription
- SMIL 2.0 Metainformation Module -- Metainformation
- SMIL 2.0 Structure Module -- Structure
- SMIL 2.0 Timing and Synchronization Modules -- BasicInlineTiming, MinMaxTiming, BasicTimeContainers, RepeatTiming and EventTiming
- SMIL 2.0 Transition Effects Module -- BasicTransitions

8.2.2 Document Conformance

A conforming 3GPP ~~PSS4~~PSS SMIL document shall be a conforming SMIL 2.0 document.

All 3GPP ~~PSS4~~PSS SMIL documents use SMIL 2.0 namespace.

<smil xmlns="`http://www.w3.org/2001/SMIL20/Language`">

3GPP ~~PSS4~~PSS SMIL documents may declare requirements using systemRequired attribute:

EXAMPLE 1: <smil xmlns="`http://www.w3.org/2001/SMIL20/Language`"
xmlns:EventTiming="http://www.w3.org/2000/SMIL20/CR/EventTiming "
systemRequired="EventTiming">

Namespace URI http://www.3gpp.org/SMIL20/PSS5~~4~~/ identifies the version of the 3GPP ~~PSS4~~PSS SMIL profile described in the present document. Authors ~~can~~ may use this URI to indicate requirement for exact 3GPP ~~PSS4~~PSS SMIL semantics for a document or a subpart of a document:

EXAMPLE 2: <smil xmlns="`http://www.w3.org/2001/SMIL20/Language`"
xmlns:pss5~~4~~="http://www.3gpp.org/SMIL20/PSS5~~4~~/"
systemReq~~z~~uired="pss5~~4~~">

The content authors should generally not include ~~The content authors generally should choose not to include~~ the PSS requirement in the document unless the SMIL document relies on PSS specific semantics that are not part of the W3C SMIL. The reason for this is that SMIL players that are not conforming 3GPP PSS user agents may not recognize the ~~PSS4~~PSS URI and thus refuse to play the document.

8.2.3 User Agent Conformance

A conforming 3GPP ~~PSS4~~PSS SMIL user agent shall be a conforming SMIL Basic User Agent.

A conforming user agent shall implement the semantics 3GPP PSS SMIL as described in clauses 8.2.4 and 8.2.5 (including subclauses).~~of the language as described in this document.~~

A conforming user agent shall recognise

- the URIs of all included SMIL 2.0 modules;
- the URI http://www.3gpp.org/SMIL20/PSS5/ as referring to all modules and semantics of the version of the 3GPP PSS SMIL profile described in the present document;
- the URI http://www.3gpp.org/SMIL20/PSS4/ as referring to all modules and semantics of the 3GPP PSS SMIL profile defined in Release 4 of the present document.

NOTE: The difference between PSS4 and PSS5 is that the BasicTransitions module has been added in PSS5.

~~A conforming user agent shall recognize the URIs of all included SMIL 2.0 modules. It shall also recognize URI http://www.3gpp.org/SMIL20/PSS4/ as referring to all modules and semantics of 3GPP SMIL language.~~

8.2.4 3GPP PSS SMIL Language Profile definition

3GPP ~~PSS4~~PSS SMIL is based on SMIL 2.0 Basic language profile [31]. This chapter defines the content model and integration semantics of the included modules where they differ from those defined by SMIL Basic.

8.2.4.1 Content Control Modules

3GPP ~~PSS4~~PSS SMIL ~~shall~~ includes the content control functionality of the BasicContentControl, SkipContentControl and PrefetchControl modules of SMIL 2.0. PrefetchControl is not part of SMIL Basic and is an additional module in this profile.

All BasicContentControl attributes listed in the module specification shall be supported.

NOTE: The SMIL specification [31] defines that all functionality of PrefetchControl module is optional. This mean that even although~~that~~ PrefetchControl is mandatory user agents may implement semantics of PrefetchControl module only partially or not to implement them at all. ~~PrefetchControl module adds the **prefetch** element to the content model of SMIL Basic **body, switch, par** and **seq** elements.~~

PrefetchControl module adds the **prefetch** element to the content model of SMIL Basic **body, switch, par** and **seq** elements. The **prefetch** element has the attributes defined by the PrefetchControl module (**mediaSize, mediaTime and bandwidth**), the **src** attribute, the BasicContentControl attributes and the **skip-content** attribute.

8.2.4.2 Layout Module

3GPP ~~PSS4~~PSS SMIL includes~~shall use~~ the BasicLayout module of SMIL 2.0 for spatial layout. The module is part of SMIL Basic.

Default values of the width and height attributes for root-layout shall be the dimensions of the device display area.

8.2.4.3 Linking Module

3GPP ~~PSS4~~PSS SMIL includes~~shall use~~ the SMIL 2.0 BasicLinking module for providing hyperlinks between documents and document fragments. This module is from SMIL Basic.

When linking to destinations outside the current document, implementations may ignore values "play" and "pause" of the 'sourcePlaystate' attribute and values "new" and "pause" of the 'show' attribute, instead using the semantics of values "stop" and "replace" respectively. When the values of 'sourcePlaystate' and 'show' are ignored the player may also ignore the 'sourceLevel' attribute since it is of no use then

8.2.4.4 Media Object Modules

3GPP ~~PSS4~~PSS SMIL ~~shall~~ includes the media elements from the SMIL 2.0 BasicMedia module and attributes from the MediaAccessibility, MediaDescription and MediaClipping modules. MediaAccessibility, MediaDescription and MediaClipping modules are additions in this profile to the SMIL Basic.

See clause 5.4 for what are the mandatory and optional MIME types a 3GPP ~~PSS4~~PSS SMIL player needs to support.

MediaClipping module adds to the profile the ability to address sub-clips of continuous media. MediaClipping module adds '**clipBegin**' and '**clipEnd**´(and for compatibility '**clip-begin**' and '**clip-end**') attributes to all media elements.

MediaAccessibility module provides basic accessibility support for media elements. New attributes '**alt**', '**longdesc**' and '**readIndex**' are added to all media elements by this module. MediaDescription module is included by the MediaAccessibility module and adds '**abstract**', '**author**' and '**copyright**' attributes to media elements.

8.2.4.5 Metainformation Module

The MetaInformation module of SMIL 2.0 is~~shall be~~ included to the profile. This module is addition in this profile to the SMIL Basic and provides a way to include descriptive information about the document content into the document.

This module adds **meta** and **metadata** elements to the content model of SMIL Basic **head** element.

8.2.4.6 Structure Module

The Structure module defines the top-level structure of the document. It is~~'s~~ included by SMIL Basic.

8.2.4.7 Timing and Synchronization modules

The timing modules included in the 3GPP SMIL are~~shall be~~ BasicInlineTiming, MinMaxTiming, BasicTimeContainers, RepeatTiming and EventTiming. The EventTiming module is an addition in this profile to the SMIL Basic.

For 'begin' and 'end' attributes either single offset-value or single event-value shall be allowed. Offsets shall not be supported with event-values.

Event timing attributes that reference invalid IDs (for example elements that have been removed by the content control) shall be treated as being indefinite.

Supported event names and semantics shall be as defined by the SMIL 2.0 Language Profile. All user agents shall be able to raise the ~~the~~ following event types:

- activateEvent;
- beginEvent;
- endEvent.

The following~~Following~~ SMIL 2.0 Language event types should be supported:

- focusInEvent;
- focusOutEvent;
- inBoundsEvent;
- outBoundsEvent;
- repeatEvent.

User agents shall ignore unknown event types and not treat them as errors.

Events do not bubble and shall be delivered to the associated media or timed elements only.

8.2.4.8 Transition Effects Module

3GPP PSS SMIL profile includes the SMIL 2.0 BasicTransitions module to provide a framework for describing transitions between media elements.

NOTE: The SMIL specification [31] defines that all functionality of BasicTransitions module is optional: "Transitions are hints to the presentation. Implementations must be able to ignore transitions if they so desire and still play the media of the presentation". This mean that even although the BasicTransitions module is mandatory user agents may implement semantics of the BasicTransitions module only partially or not to implement them at all. Content authors should use transitions in their SMIL presentation where this appears useful. User agents that fully support the semantics of the Basic Transitions module will render the presentation with the specified transitions. All other user agents will leave out the transitions but present the media content correctly.

User agents that implement the semantics of this module should implement at least the following transition effects described in SMIL 2.0 specification [31]:

- barWipe;
- irisWipe;
- clockWipe;
- snakeWipe;
- pushWipe;

- slideWipe;
- fade;

A user agent should implement the default subtype of these transition effects.

A user agent that implements the semantics of this module shall at least support transition effects for non-animated image media elements. For purposes of the Transition Effects modules, two media elements are considered overlapping when they occupy the same region.

BasicTransitions module adds attributes 'transIn' and 'transOut' to the media elements of the Media Objects modules, and value "transition" to the set of legal values for the 'fill' attribute of the media elements. It also adds transition element to the content model of the head element.

8.2.5 Content Model

This table shows the full content model and attributes of the 3GPP ~~PSS4~~PSS SMIL profile. The attribute collections used are defined by SMIL Basic ([31], SMIL Host Language Conformance requirements, chapter 2.4). Changes to~~ the~~ SMIL Basic are shown in **bold**.

Table 1: Content model for the 3GPP PSS SMIL profile

Element	Elements	Attributes
smil	head, body	COMMON-ATTRS, CONTCTRL-ATTRS, xmlns
head	layout, switch, **meta**, **metadata, transition**	COMMON-ATTRS
body	TIMING-ELMS, MEDIA-ELMS, switch, a, **prefetch**	COMMON-ATTRS
layout	root-layout, region	COMMON-ATTRS, CONTCTRL-ATTRS, type
root-layout	EMPTY	COMMON-ATTRS, backgroundColor, height, width, skip-content
region	EMPTY	COMMON-ATTRS, backgroundColor, bottom, fit, height, left, right, showBackground, top, width, z-index, skip-content, regionName
ref, animation, audio, img, video, text, textstream	area	COMMON-ATTRS, CONTCTRL-ATTRS, TIMING-ATTRS, repeat, region, MEDIA-ATTRS, **clipBegin(clip-begin), clipEnd(clip-end), alt, longDesc, readIndex, abstract**, **author**, **copyright, transIn, transOut**
a	MEDIA-ELMS	COMMON-ATTRS, LINKING-ATTRS
area	EMPTY	COMMON-ATTRS, LINKING-ATTRS, TIMING-ATTRS, repeat, shape, coords, nohref
par, seq	TIMING-ELMS, MEDIA-ELMS, switch, a, **prefetch**	COMMON-ATTRS, CONTCTRL-ATTRS, TIMING-ATTRS, repeat
switch	TIMING-ELMS, MEDIA-ELMS, layout, a, **prefetch**	COMMON-ATTRS, CONTCTRL-ATTRS
prefetch	**EMPTY**	**COMMON-ATTRS, CONTCTRL-ATTRS, mediaSize, mediaTime, bandwidth, src, skip-content**
meta	**EMPTY**	**COMMON-ATTRS, content, name, skip-content**
metadata	**EMPTY**	**COMMON-ATTRS, skip-content**
transition	**EMPTY**	**COMMON-ATTRS, CONTCTRL-ATTRS, type, subtype, startProgress, endProgress, direction, fadeColor. skip-content**

9 Interchange format for MMS

9.1 General

The MPEG-4 file format [34] is mandated in [35] to be used for continuous media along the entire delivery chain envisaged by the MMS, independent on whether the final delivery is done by streaming or download, thus enhancing interoperability.

In particular, the following stages are considered:

- upload from the originating terminal to the MMS proxy;
- file exchange between MMS servers;
- transfer of the media content to the receiving terminal, either by file download or by streaming. In the first case the self-contained file is transferred, whereas in the second case the content is extracted from the file and streamed according to open payload formats. In this case, no trace of the file format remains in the content that goes on the wire/in the air.

Additionally, the MPEG-4 file format should~~can~~ be used for the storage in the servers and the "hint track" mechanism may~~can~~ be used for the preparation for streaming.

The clause 9.2 of the present document gives the necessary requirements to follow for the MPEG-4 file format used in MMS. These requirements will guarantee PSS to interwork with MMS as well as the MPEG-4 file format to be used internally within the MMS system. For PSS servers not interworking with MMS there is no requirement to follow these guidelines.

9.2 ~~MPEG-4 file~~File format guidelines

NOTE: The file format used in this specification for timed multimedia (such as video, associated audio and timed text) is structurally based on the MP4 file format as defined in [34]. However, since non-ISO codecs are used here, it is called the 3GPP file format and has its own file extension and MIME type to distinguish these files from MPEG-4 files. When this specification refers to the MP4 file format, it is referring to its structure (ISO file format), not to its conformance definition.

9.2.1 Registration of non-ISO codecs

How to include the non-ISO code streams AMR narrow-band speech, AMR wideband speech,~~ and~~ H.263 encoded video and timed text in MP4 files is described in annex D of the present document.

9.2.2 Hint tracks

The hint tracks are a mechanism that the server implementation may choose to use in preparation for the streaming of media content contained in MP4 files. However, it should be observed that the usage of the hint tracks is an internal implementation matter for the server, and it falls outside the scope of the present document.

9.2.3 Self-contained MP4 files

All media in the MP4 file shall be self-contained, i.e. there shall not be referencing to external media data from inside the MP4 file.

9.2.4 MPEG-4 systems specific elements

Tracks relative to MPEG-4 system architectural elements (e.g. BIFS scene description tracks or OD Object descriptors) are optional and shall be ignored. The adoption of the MPEG-4 file format does not imply the usage of MPEG-4 systems architecture. The receiving terminal is not required to implement any of the specific MPEG-4 system architectural elements.

9.2.5 Interpretation of MPEG-4 file format

All index numbers used in MPEG-4 file format start with the value one rather than zero, in particular “first-chunk” in Sample to chunk atom, “sample-number” in Sync sample atom and “shadowed-sample-number”, “sync-sample-number” in Shadow sync sample atom.

Annex A (informative): Protocols

A.1 SDP

This clause gives some background information on SDP for PSS clients.

Table A.1 provides an overview of the different SDP fields that can be identified in a SDP file. The order of SDP fields is~~are~~ mandated as specified in RFC 2327 [6].

Table A.1: Overview of fields in SDP for PSS clients

Type	Description		Requirement according to [6]	Requirement according to the present document
Session Description				
V	Protocol version		R	R
O	Owner/creator and session identifier		R	R
S	Session Name		R	R
I	Session information		O	O
U	URI of description		O	O
E	Email address		O	O
P	Phone number		O	O
C	Connection Information		R	R
B	Bandwidth information	AS	O	R
One or more Time Descriptions (See below)				
Z	Time zone adjustments		O	O
K	Encryption key		O	O
A	Session attributes	control	O	R
		range	O	R
One or more Media Descriptions (See below)				
Time Description				
T	Time the session is active		R	R
R	Repeat times		O	O
Media Description				
M	Media name and transport address		R	R
I	Media title		O	O
C	Connection information		R	R
B	Bandwidth information	AS	O	R
K	Encryption Key		O	O
A	Attribute Lines	control	O	R
		range	O	R
		fmtp	O	R
		rtpmap	O	R
		X-predecbufsize	ND	O
		X-initpredecbufperiod	ND	O
		X-initpostdecbufperiod	ND	O
		X-decbyterate	ND	O

Note 1: R = Required, O = Optional, ND = Not Defined

Note 2: The "c" type is only required on the session level if not present on the media level.

Note 3: The "c" type is only required on the media level if not present on the session level.

Note 4: According to RFC 2327, either an 'e' or 'p' field must be present in the SDP description. On the other hand, both fields will be made optional in the future release of SDP. So, for the sake of robustness and maximum interoperability, either an 'e' or 'p' field shall be present during the server's SDP file creation, but the client should also be ready to receive SDP content containing~~that does not have~~ neither 'e' nor 'p' fields.

The example below shows an SDP file that could be sent to a PSS client to initiate unicast streaming of a H.263 video sequence.

EXAMPLE:

```
v=0
o=ghost 2890844526 2890842807 IN IP4 192.168.10.10
s=3GPP Unicast SDP Example
i=Example of Unicast SDP file
u=http://www.infoserver.com/ae600
e=ghost@mailserver.com
c=IN IP4 0.0.0.0
b=AS:128
t=0 0
a=range:npt=0-45.678
m=video 1024 RTP/AVP 96
b=AS:128
a=rtpmap:96 H263-2000/90000
a=fmtp:96 profile=3;level=10
a=control:rtsp:~~;~~//mediaserver.com/movie
a=recvonly
```

A.2 RTSP

A.2.1 General

Clause 5.3.2 of the present document defines the required RTSP support in PSS clients and servers by making references to Appendix D of [5]. The current clause gives an overview of the methods (see Table A.2) and headers (see Table A.3) that are specified in the referenced Appendix D. An example of an RTSP session is also given.

Table A.2: Overview of the required RTSP method support

Method	**Requirement for a minimal on-demand playback client according to [5].**	**Requirement for a PSS client according to the present document.**	**Requirement for a minimal on-demand playback server according to [5].**	**Requirement for a PSS server according to the present document.**
OPTIONS	O	O	Respond	Respond
REDIRECT	Respond	Respond	O	O
DESCRIBE	O	Generate	O	Respond
SETUP	Generate	Generate	Respond	Respond
PLAY	Generate	Generate	Respond	Respond
PAUSE	Generate	Generate	Respond	Respond
TEARDOWN	Generate	Generate	Respond	Respond
NOTE 1: O = Support is optional NOTE 2: 'Generate' means that the client/server is required to be able to generate the request. NOTE 3: 'Respond' means that the client/server is required understand and be able to properly respond to the request.				

Table A.3: Overview of the required RTSP header support

Header	**Requirement for a minimal on-demand playback client according to [5].**	**Requirement for a PSS client according to the present document.**	**Requirement for a minimal on-demand playback server according to [5].**	**Requirement for a PSS server according to the present document.**
Connection	include/understand	include/understand	include/understand	include/understand
Content-Encoding	understand	understand	include	include
Content-Language	understand	understand	include	include
Content-Length	understand	understand	include	include
Content-Type	understand	understand	include	include
CSeq	include/understand	include/understand	include/understand	include/understand
Location	understand	understand	O	O
Public	O	O	include	include
Range	O	include/understand	understand	include/understand
Require	O	O	understand	understand
RTP-Info	understand	understand	include	include
Session	include	include	understand	understand
Transport	include/understand	include/understand	include/understand	include/understand
NOTE 1: O = Support is optional NOTE 2: 'include' means that the client/server is required to be able to include the header in a request or response. NOTE 3: 'understand' means that the client/server is required to be able to understand the header and respond properly if the header is received in a request or response.				

The example below is intended to give some more understanding of how RTSP and SDP are used within the 3GPP PSS. The example assumes that the streaming client has the RTSP URL to a presentation consisting of an H.263 video sequence and AMR speech. RTSP messages sent from the client to the server are in **bold** and messages from the server to the client in *italic*. In the example the server provides aggregate control of the two streams.

EXAMPLE:

DESCRIBE rtsp://mediaserver.com/movie.test RTSP/1.0
CSeq: 1

RTSP/1.0 200 OK
CSeq: 1
Content-Type: application/sdp
Content-Length: 435

v=0
o=- 950814089 950814089 IN IP4 144.132.134.67
s=Example of aggregate control of AMR speech and H.263 video
e=foo@bar.com
c=IN IP4 0.0.0.0

b=AS:77
t=0 0
a=range:npt=0-59.3478
a=control:*
m=audio 0 RTP/AVP 97
b=AS:13
a=rtpmap:97 AMR/8000
a=fmtp:97
a=maxptime:200
a=control:streamID=0
m=video 0 RTP/AVP 98
b=AS:64
a=rtpmap:98 H263-2000/90000
a=fmtp:98 profile=3;level=10
a=control: streamID=1

~~c=IN IP4 192.168.30.29~~
~~b=AS:77~~
~~t=0 0~~
~~a=range:npt=0-59.3478~~
~~a=control:*~~
~~m=audio 0 RTP/AVP 97~~
~~b=AS:13~~
~~a=rtpmap:97 AMR/8000~~
~~a=fmtp:97 mode-set=0,2,5,7; maxptime=200~~
~~a=control:streamID=0~~
~~m=video 0 RTP/AVP 98~~
~~b=AS:64~~
~~a=rtpmap:98 H263-2000/90000~~
~~a=fmtp:98 profile=3;level=10~~
~~a=control: streamID=1~~

SETUP rtsp://mediaserver.com/movie.test/streamID=0 RTSP/1.0
CSeq: 2
Transport: RTP/AVP/UDP;unicast;client_port=3456-3457

RTSP/1.0 200 OK
CSeq: 2
Transport: RTP/AVP/UDP;unicast;client_port=3456-3457; server_port=5678-5679
Session: dfhyrio90llk

SETUP rtsp://mediaserver.com/movie.test/streamID=1 RTSP/1.0
CSeq: 3
Transport: RTP/AVP/UDP;unicast;client_port=3458-3459
Session: dfhyrio90llk

RTSP/1.0 200 OK
CSeq: 3
Transport: RTP/AVP/UDP;unicast;client_port=3458-3459; server_port=5680-5681
Session: dfhyrio90llk

PLAY rtsp://mediaserver.com/movie.test RTSP/1.0
CSeq: 4
Session: dfhyrio90llk

RTSP/1.0 200 OK
CSeq: 4
Session: dfhyrio90llk
Range: npt=0-
RTP-Info: url= rtsp://mediaserver.com/movie.test/streamID=0; seq=9900~~093~~;rtptime=4470048,
url= rtsp://mediaserver.com/movie.test/streamID=1; seq=1004~~096~~;rtptime=1070549

NOTE: Headers can be folded onto multiple lines if the continuation line begins with a space or horizontal tab. For more information, see RFC2616 [17].

The user watches the movie for 20 seconds and then decides to fast forward to 10 seconds before the end…

PAUSE rtsp://mediaserver.com/movie.test RTSP/1.0
CSeq: 5
Session: dfhyrio90llk

PLAY rtsp://mediaserver.com/movie.test RTSP/1.0
CSeq: 6
Range: npt=50-59.3478
Session: dfhyrio90llk

RTSP/1.0 200 OK
CSeq: 5
Session: dfhyrio90llk

RTSP/1.0 200 OK
CSeq: 6
Session: dfhyrio90llk
Range: npt=50-59.3478
RTP-Info: url= rtsp://mediaserver.com/movie.test/streamID=0;
seq=39900~~043~~;rtptime=44470648,
url= rtsp://mediaserver.com/movie.test/streamID=1;
seq=31004~~046~~;rtptime=41090349

After the movie is over the client issues a TEARDOWN to end the session…

TEARDOWN rtsp://mediaserver.com/movie.test RTSP/1.0
CSeq: 7
Session: dfhyrio90llk

RTSP/1.0 200 OK
Cseq: 7
Session: dfhyrio90llk
Connection: close

A.2.2 Implementation guidelines

A.2.2.1 Usage of persistent TCP

Considering the potentially long round-trip-delays in a packet switched streaming service over UMTS it is important to keep the number of messages exchanged between a server and a client low. The number of requests and responses exchanged is one of the factors that will determine how long it takes from the time that a user initiates PSS until the streams starts playing in a client.

RTSP methods are sent over either TCP or UDP for IP. Both client and server shall~~must~~ support RTSP over TCP whereas RTSP over UDP is optional. For TCP the connection can be persistent or non-persistent. A persistent connection is used for several RTSP request/response pairs whereas one connection is used per RTSP request/response pair for the non-persistent connection. In the non-persistent case each connection will start with the three-way handshake (SYN, ACK, SYN) before the RTSP request can be sent. This will increase the time for the message to be sent by one round trip delay.

For these reasons it is recommended that 3GPP PSS clients should use a persistent TCP connection, at least for the initial RTSP methods until media starts streaming.

A.2.2.2 Detecting link aliveness

In the wireless environment, connection may be lost due to fading, shadowing, loss of battery power, or turning off the terminal even though the PSS session is active. In order for the server to be able to detect the client's aliveness, the PSS client should send "wellness" information to the PSS server for a defined interval as described in the RFC2326. There are several ways for detecting link aliveness described in the RFC2326, however, the client should be careful about issuing "PLAY method without Range header field" too close to the end of the streams, because it may conflict with pipelined PLAY requests. Below is the list of recommended "wellness" information for the PSS clients and servers in a prioritised order.

1. RTCP
2. OPTIONS method with Session header field

NOTE: Both servers and clients can initiate this OPTIONS method.

A.3 RTP

A.3.1 General

Void.

A.3.2 Implementation guidelines

A.3.2.1 Maximum RTP packet size

The RFC 1889 (RTP) [9] does not impose a maximum size on RTP packets. However, when RTP packets are sent over the radio link of a 3GPP PSS system there is an advantage in limiting the maximum size of RTP packets.

Two types of bearers can be envisioned for streaming using either acknowledged mode (AM) or unacknowledged mode (UM) RLC. The AM uses retransmissions over the radio link whereas the UM does not. In UM mode large RTP packets are more susceptible to losses over the radio link compared to small RTP packets since the loss of a segment may result in the loss of the whole packet. On the other hand in AM mode large RTP packets will result in larger delay jitter compared to small packets as there is a larger chance that more segments have to be retransmitted.

For these reasons it is recommended that the maximum size of RTP packets should be limited in~~s~~ size taking into account the wireless link. This will decrease the RTP packet loss rate particularly for RLC in UM. For RLC in AM the delay jitter will be reduced permitting the client to use a smaller receiving buffer. It should also be noted that too small RTP packets could result in too much overhead if IP/UDP/RTP header compression is not applied or unnecessary load at the streaming server.

In the case of transporting video in the payload of RTP packets it may be that a video frame is split into more than one RTP packet in order not to produce too large RTP packets. Then, to be able to decode packets following a lost packet in the same video frame, it is recommended that synchronisation information be inserted at the start of such RTP packets. For H.263 this implies the use of GOBs with non-empty GOB headers and in the case of MPEG-4 video the use of video packets (resynchronisation markers). If the optional Slice Structured mode (Annex K) of H.263 is in use, GOBs are replaced by slices.

A.3.2.2 Sequence number and timestamp in the presence of NPT jump

The description below is intended to give more understanding of how RTP sequence number and timestamp are specified within the 3GPP PSS in the presence of NPT jumps. The jump happens when a client sends a PLAY request to skip media.

The RFC 2326 (RTSP) [5] specifies that both RTP sequence numbers and RTP timestamps must be continuous and monotonic across jumps of NPT. Thus when a server receives a request for a skip of the media that causes a jump of NPT, it shall specify RTP sequence numbers and RTP timestamps continuously and monotonically across the skip of the media to conform to the RTSP specification. Also, the server may respond with "seq" in the RTP-Info field if this parameter is known at the time of issuing the response.

A.3.2.3 RTCP transmission interval

In RTP [9], Section 6.2, rules for the calculation of the interval between the sending of two consecutive RTCP packets, i.e. the RTCP transmission interval, are defined. These rules consist of two steps:

- Step 1: an algorithm that calculates a transmission interval from parameters such as the session bit rate and the average RTCP packet size. This algorithm is described in [9], annex A.7.

- Step 2: Taking the maximum of the transmission interval computed in step 1 and a mandatory fixed minimum RTCP transmission interval of 5 seconds.

Implementations conforming to this TS shall perform step 1 and may perform step 2. All other algorithms and rules of [9] stay valid and shall be followed

Following these recommendations results in regular sending of RTCP messages, where the interval between those is depending on the session bandwidth and the RTCP packet size.

A.4 Capability exchange

A.4.1 Overview

Clause A.4 provides detailed information about the structure and exchange of device capability descriptions for the PSS. It complements the normative part contained in clause 5.2 of the present document.

The functionality is sometimes referred to as capability exchange. Capability exchange in PSS uses the CC/PP [39] framework and reuse parts of the CC/PP application UAProf [40].

To facilitate server-side content negotiation for streaming, the PSS server needs to have access to a description of the specific capabilities of the mobile terminal, i.e. the device capability description. The device capability description contains a number of attributes. During the set-up of a streaming session the PSS server can use the description to provide the mobile terminal with the correct type of multimedia content. Concretely, it is envisaged that servers use information about the capabilities of the mobile terminal to decide which stream(s) to provision to the connecting terminal. For instance, the server could compare the requirements on the mobile terminal for multiple available variants of a stream with the actual capabilities of the connecting terminal to determine the best-suited stream(s) for that particular terminal. A similar mechanism could also be used for other types of content.

A device capability description contains a number of device capability attributes. In the present document they are referred to as just attributes. The current version of PSS does not include a definition of any specific user preference attributes. Therefore we use the term device capability description. However, it should be noted that even though no specific user preference attributes are included, simple tailoring to the preferences of the user could be achieved by temporarily overrides of the available attributes. E.g. if the user for a particular session only would like to receive mono sound even though the terminal is capable of stereo, this can be accomplished by providing an override for the "AudioChannels" attribute. It should also be noted that the extension mechanism defined would enable an easy introduction of specific user preference attributes in the device capability description if needed.

The term device capability profile or profile is sometimes used instead of device capability description to describe a description of device capabilities and/or user preferences. The three terms are used interchangeably in the present document.

Figure A.1 illustrates how capability exchange in PSS is performed. In the simplest case the mobile terminal informs the PSS server(s) about its identity so that the latter can retrieve the correct device capability profile(s) from the device profile server(s). For this purpose, the mobile terminal adds one or several URLs to RTSP and/or HTTP protocol data units that it sends to the PSS server(s). These URLs point to locations on one or several device profile servers from where the PSS server should retrieve the device capability profiles. This list of URLs is encapsulated in RTSP and HTTP protocol data units using additional header field(s). The list of URLs is denoted URLdesc. The mobile terminal may supplementthe URLdesc with extra attributes or overrides for attributes already defined in the profile(s) located at URLdesc. This information is denoted Profdiff. As URLdesc, Profdiff is encapsulated in RTSP and HTTP protocol data units using additional header field(s).

The device profile server in Figure A.1 is the logical entity that stores the device capability profiles. The profile needed for a certain request from a mobile terminal may be stored on one or several such servers. A terminal manufacturer or a software vendor could maintain a device profile server to provide device capability profiles for its products. It would also be possible for an operator to manage a device profile server for its subscribers and then e.g. enable the subscriber to make user specific updates to the profiles. The device profile server provides device capability profiles to the PSS server on request.

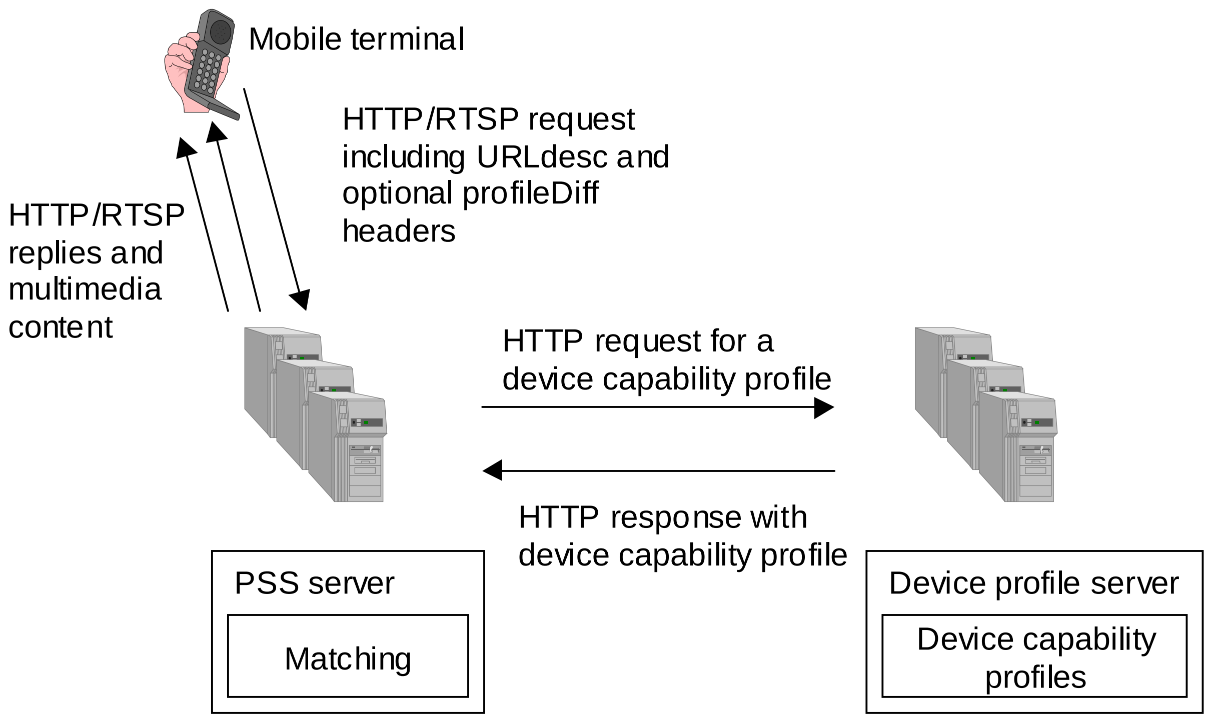

Mobile terminal
HTTP/RTSP request including URLdesc and optional profileDiff headers
HTTP/RTSP replies and multimedia content
HTTP request for a device capability profile
HTTP response with device capability profile
PSS server
Matching
Device profile server
Device capability profiles

Figure A.1: Functional components in PSS capability exchange

The PSS server is the logical entity that provides multimedia streams and other, static content (e.g. SMIL documents, images, and graphics) to the mobile terminal (see Figure A.1). A PSS application might involve multiple PSS servers, e.g. separate servers for multimedia streams and for static content. A PSS server handles the matching process. Matching is a process that takes place in the PSS servers (see Figure A.1). The device capability profile is compared with the content descriptions at the server and the best fit is delivered to the client.

A.4.2 Scope of the specification

The following bullet list describes what is considered to be within the scope of the specification for capability exchange in PSS.

- Definition of the structure for the device capability profiles, see clause A.4.3.

- Definition of the CC/PP vocabularies, see clause A.4.4.
    - Reference to a set of device capability attributes for multimedia content retrieval applications that have already been defined by UAProf [40]. The purpose of this reference is to point out which attributes are useful for the PSS application.
    - Definition of a set of device capability attributes specifically for PSS applications that are missing in UAProf.

- It is important to define an extension mechanism to easily add attributes since it is not possible to cover all attributes from the beginning. The extension mechanism is described in clause A.4.5.
- The structure of URLdesc, Profdiff and their interchange is described in clause A.4.6.
- Protocols for the interchange of device capability profiles between the PSS server and the device profile server is defined in clause 5.2.7.

The specification does not include:

- rules for the matching process on the PSS server. These mechanisms should be left to the implementations. For interoperability, only the format of the device capability description and its interchange is relevant.
- definition of specific user preference attributes. It is very difficult to standardise such attributes since they are dependent on the type of personalised services one would like to offer the user. The extensible descriptions format and exchange mechanism proposed in this document provide the means to create and exchange such attributes if needed in the future. However, as explained in clause A.4.1 limited tailoring to the preferences of the user could be achieved by temporarily overridingavailable attributes in the vocabularies already defined for PSS. The vocabulary also includes some very basic user preference attributes. For example, the profile includes a list of preferred languages. Also the list of MIME types can be interpreted as user preference, e.g. leaving out audio MIME's could mean that user does not want to receive any audio content. The available attributes are described in clause 5.2.3 of the present document.
- requirements for caching of device capability profiles on the PSS server. In UAProf, a content server can cache the current device capability profile for a given WSP session. This feature relies on the presence of WSP sessions. Caching significantly increases the complexity of both the implementations of the mobile terminal and the server. However, HTTP is used between the PSS server and the device profile server. For this exchange, normal content caching provisions as defined by HTTP apply and the PSS server may utilise this to speed up the session set-up (see clause 5.2.7)
- intermediate proxies. This feature is considered not relevant in the context of PSS applications.

A.4.3 The device capability profile structure

A device capability profile is a description of the capabilities of the device and possibly also the preferences of the user of that device. It can be used to guide the adaptation of content presented to the device. A device capability profile for PSS is a RDF [41] document that follows the structure of the CC/PP framework [39] and the CC/PP application UAProf [40]. The terminology of CC/PP is used in this text and therefore briefly described here.

Attributes are used for specifying the device capabilities and user preferences. A set of attribute names, permissible values and semantics constitute a CC/PP vocabulary. A RDF schema defines a vocabulary. The syntax of the attributes is defined in the schema but also, to some extent, the semantics. A profile is an instance of a schema and contains one or more attributes from the vocabulary. Attributes in a schema are divided into components distinguished by attribute characteristics. In the CC/PP specification it is anticipated that different applications will use different vocabularies. According to the CC/PP framework a hypothetical profile might look like Figure A.2. A further illustration of how a profile might look like is given in the example in clause A.4.7.

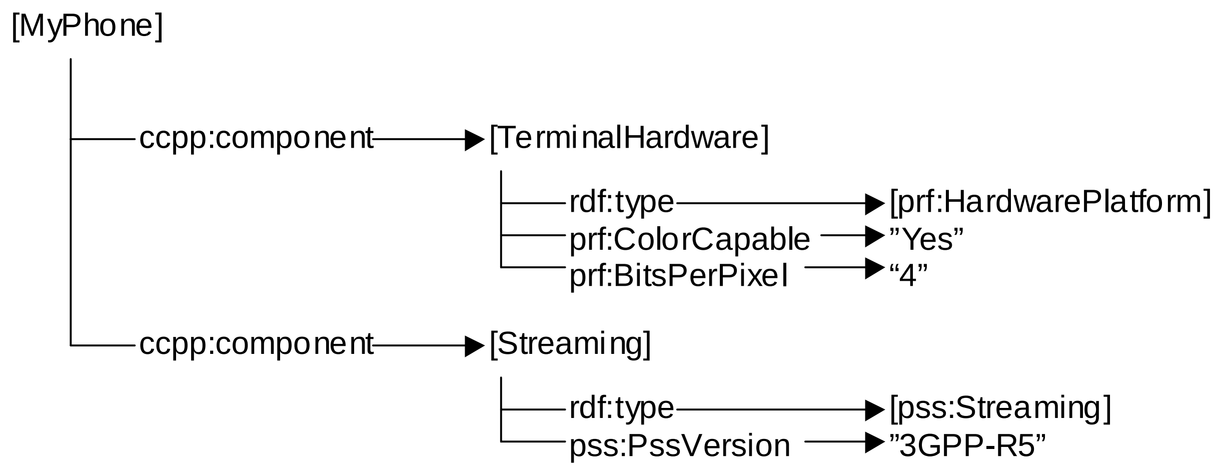


Figure A.2: Illustration of the profile structure

A CC/PP schema is extended through the introduction of new attribute vocabularies and a device capability profile can use attributes drawn from an arbitrary number of different vocabularies. Each vocabulary is associated with a unique XML namespace. This mechanism makes it possible to reuse attributes from other vocabularies. It should be mentioned that the prefix **ccpp** identifies elements of the CCPP namespace (URI http://www.w3.org/1999/02/22-rdf-syntax-ns), **prf** identifies elements of the UAProf namespace (URI http://www.wapforum.org/profiles/UAPROF/ccppschema-20010330) , **rdf** identifies elements of the RDF namespace (URI http://www.w3.org/1999/02/22-rdf-syntax-ns) and **pss** identifies elements of the Streaming namespace. (URI http://www.3gpp.org/profiles/PSS/ccppschema-PSS5).

Attributes of a component can be included directly or may be specified by a reference to a CC/PP default profile. Resolving a profile that includes a reference to a default profile is time-consuming. When the PSS server receives the profile from a device profile server the final attribute values can not be determined until the default profile has been requested and received. Support for defaults is required by the CC/PP specification [39]. Due to these problems, there is a recommendation made in clause 5.2.6 to not use the CC/PP defaults element in PSS device capability profile documents.

A.4.4 CC/PP Vocabularies

A CC/PP vocabulary shall according to CC/PP and UAProf include:

- A RDF schema for the vocabulary based on the CC/PP schema.
- A description of the semantics/type/resolution rules/sample values for each attribute.
- A unique namespace shall be assigned to each version of the profile schema.

Additional information that could be included in the profile schema:

- A description about the profile schema, i.e. the purpose of the profile, how to use it, when to use it etc.
- A description of extensibility,i.e.how to handle future extensions of the profile schema.

A device capability profile can use an arbitrary number of vocabularies and thus it is possible to reuse attributes from other vocabularies by simply referencing the corresponding namespaces. The focus of the PSS vocabulary is content formatting which overlaps the focus of the UAProf vocabulary. UAProf is specified by WAP Forum and is an architecture and vocabulary/schema for capability exchange in the WAP environment. Since there are attributes in the UAProf vocabulary suitable for streaming applications these are reused and combined with a PSS application specific streaming component. This makes the PSS vocabulary an extension vocabulary to UAProf. The CC/PP specification encourages reuse of attributes from other vocabularies. To avoid confusion, the same attribute name should not be used in different vocabularies. In clause 5.2.3.3 a number of attributes from UAProf [40] are recommended for PSS. The PSS base vocabulary is defined in clause 5.2.3.2.

A profile is allowed to instantiate a subset of the attributes in the vocabularies and no specific attributes are required but insufficient description may lead to content unable to be shown by the client.

A.4.5 Principles of extending a schema/vocabulary

The use of RDF enables an extensibility mechanism for CC/PP-based schemas that addresses the evolution of new types of devices and applications. The PSS profile schema specification is going to provide a base vocabulary but in the future new usage scenarios might have need for expressing new attributes. This is the reason why there is a need to specify how extensions of the schema will be handled. If the TSG responsible for the present document updates the base vocabulary schema a new unique namespace will be assigned to the updated schema. In another scenario the TSG may decide to add a new component containing specific user related attributes. This new component will be assigned a new namespace and it will not influence the base vocabulary in any way. If other organisations or companies make extensions this can be either as a new component or as attributes added to the existing base vocabulary component where the new attributes uses a new namespace. This ensures that third parties can define and maintain their own vocabularies independently from the PSS base vocabulary.

A.4.6 Signalling of profile information between client and server

URLdesc and Profdiff were introduced in clause A.4.1. The URLdesc is a list of URLs that point to locations on device profile servers from where the PSS server retrieves suitable device capability profiles. The Profdiff contains additional capability description information; e.g. overrides for certain attribute values. Both URLdesc and Profdiff are encapsulated in RTSP and HTTP messages using additional header fields. This can be seen in Figure A.1. In clause 9.1 of [40] three new HTTP headers are defined that can be used to implement the desired functionality: "x-wap-profile", "x-wap-profile-diff" and "x-wap-profile-warning". These headers are reused in PSS for both HTTP and RTSP.

- The "x-wap-profile" is a request header that contains a list of absolute URLs to device capability descriptions and profile diff names. The profile diff names correspond to additional profile information in the "x-wap-profile-diff" header.
- The "x-wap-profile-diff" is a request header that contains a subset of a device capability profile.
- The "x-wap-profile-warning" is a response header that contains error codes explaining to what extent the server has been able to match the terminal request.

Clause 5.2.5 of the present document defines this exchange mechanism.

It is left to the mobile terminal to decide when to send x-wap-profile headers. The mobile terminal could send the "x-wap-profile" and "x-wap-profile-diff" headers with each RTSP DESCRIBE and/or with each RTSP SETUP request. Sending them in the RTSP DESCRIBE request is useful for the PSS server to be able to make a better decision which presentation description to provision to the client. Sending the "x-wap-profile" and "x-wap-profile-diff" headers with an HTTP request is useful whenever the mobile terminal requests some multimedia content that will be used in the PSS application. For example it can be sent with the request for a SMIL file and the PSS server can see to it that the mobile terminal receives a SMIL file which is optimised for the particular terminal. Clause 5.2.5 of the present document gives recommendations for when profile information should be sent.

It is up to the PSS server to retrieve the device capability profiles using the URLs in the "x-wap-profile" header. The PSS server is also responsible to merge the profiles then received. If the "x-wap-profile-diff" header is present it must also merge that information with the retrieved profiles. This functionality is defined in clause 5.2.6.

It should be noted that it is up the implementation of the mobile terminal what URLs to send in the "x-wap-profile" header. For instance, a terminal could just send one URL that points to a complete description of its capabilities. Another terminal might provide one URL that points to a description of the terminal hardware. A second URL that points to a description of a particular software version of the streaming application, and a third URL that points to the description of a hardware or software plug-in that is currently added to the standard configuration of that terminal. From this example it becomes clear that sending URLs from the mobile terminal to the server is good enough not only for static profiles but that it can also handle re-configurations of the mobile terminal such as software version changes, software plug-ins, hardware upgrades, etc.

As described above the list of URLs in the x-wap-profile header is a powerful tool to handle dynamic changes of the mobile terminal. The "x-wap-profile-diff" header could also be used to facilitate the same functionality. To use the "x-wap-profile-diff" header to e.g. send a complete profile (no URL present at all in the "x-wap-profile header") or updates as a result of e.g. a hardware plug-in is not recommended unless some compression scheme is applied over the air-interface. The reason is of course that the size of a profile may be large.

A.4.7 Example of a PSS device capability description

The following is an example of a device capability profile as it could be available from a device profile server. The XML document includes the description of the imaginary "Phone007" phone.

Instead of a single XML document the description could also be spread over several files. The PSS server would need to retrieve these profiles separately in this case and would need to merge them. For instance, this would be useful when device capabilities of this phone that are related to streaming would differ among different versions of the phone. In this case the part of the profile for streaming would be separated from the rest into its own profile document. This separation allows describing the difference in streaming capabilities by providing multiple versions of the profile document for the streaming capabilities.

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns"
         xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp"
         xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010330"
         xmlns:pss5="http://www.3gpp.org/profiles/PSS/ccppschema-PSS5">

  <rdf:Description rdf:about="http://www.bar.com/Phones/Phone007">

    <ccpp:component>
      <rdf:Description ID="HardwarePlatform">
      <rdf:type rdf:resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-
20010330#HardwarePlatform" />
        <prf:BitsPerPixel>4</prf:BitsPerPixel>
        <prf:ColorCapable>Yes</prf:ColorCapable>
        <prf:PixelAspectRatio>1x2</prf:PixelAspectRatio>
        <prf:PointingResolution>Pixel</prf:PointingResolution>

        <prf:Model>Phone007</prf:Model>
        <prf:Vendor>Ericsson</prf:Vendor>
      </rdf:Description>
    </ccpp:component>

    <ccpp:component>
      <rdf:Description ID="SoftwarePlatform">
      <rdf:type rdf:resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-
20010330#SoftwarePlatform" />
        <prf:CcppAccept-Charset>
          <rdf:Bag>
            <rdf:li>UTF-8</rdf:li>
            <rdf:li>ISO-10646-UCS-2</rdf:li>
          </rdf:Bag>
        </prf:CcppAccept-Charset>
        <prf:CcppAccept-Encoding>
          <rdf:Bag>
            <rdf:li>base64</rdf:li>
            <rdf:li>quoted-printable</rdf:li>
          </rdf:Bag>
        </prf:CcppAccept-Encoding>
        <prf:CcppAccept-Language>
          <rdf:Seq>
            <rdf:li>en</rdf:li>
        <rdf:li>se</rdf:li>
          </rdf:Seq>
        </prf:CcppAccept-Language>
      </rdf:Description>
    </ccpp:component>

    <ccpp:component>
      <rdf:Description ID="Streaming">
      <rdf:type rdf:resource=" http://www.3gpp.org/profiles/PSS/ccppschema-PSS5#Streaming" />
        <pss5:AudioChannels>Stereo</pss5:AudioChannels>
        <pss5:VideoPreDecoderBufferSize>30720</pss5:VideoPreDecoderBufferSize>
        <pss5:VideoInitialPostDecoderBufferingPeriod>0</pss5:VideoInitialPostDecoderBufferingPeriod>
        <pss5:VideoDecodingByteRate>16000</pss5:VideoDecodingByteRate>
        <pss5:RenderingScreenSize>73x50</pss5:RenderingScreenSize>
    <pss5:PssAccept>
```

```
          <rdf:Bag>
            <rdf:li>audio/AMR-WB;octet-alignment</rdf:li>
            <rdf:li>video/MP4V-ES</rdf:li>
          </rdf:Bag>
        </pss5:PssAccept>
        <pss5:PssAccept-Subset>
          <rdf:Bag>
            <rdf:li>JPEG-PSS</rdf:li>
          </rdf:Bag>
        </pss5:PssAccept-Subset>
        <pss5:PssVersion>3GPP-R5</pss5:PssVersion>
        <pss5:RenderingScreenSize>70x40</pss5:RenderingScreenSize>
        <pss5:SmilBaseSet>SMIL-3GPP-R4</pss5:SmilBaseSet>
        <pss5:SmilModules>
          <rdf:Bag>
            <rdf:li>BasicTransitions</rdf:li>
            <rdf:li>MulitArcTiming</rdf:li>
          </rdf:Bag>
        </pss5:SmilModules>
      </rdf:Description>
    </ccpp:component>

  </rdf:Description>
</rdf:RDF>
```

Annex B (informative): SMIL authoring guidelines

B.1 General

This is an informative annex for SMIL presentation authors. Authors can expect that PSS clients can handle the SMIL module collection defined in clause 8.2, with the restrictions defined in this Annex. When creating SMIL documents the author is recommended to consider that terminals may have small displays and simple input devices. The media types and their encoding included in the presentation should be restricted to what is described in clause 7 of the present document. Considering that many mobile devices may have limited software and hardware capabilities, the number of media to be played simultaneous should be limited. For example, many devices will not be able to handle more than one video sequence at the time.

B.2 BasicLinking

The Linking Modules define elements and attributes for navigational hyperlinking, either through user interaction or through temporal events. The BasicLinking module defines the "a" and "area" elements for basic linking:

a	Similar to the "a" element in HTML it provides a link from a media object through the href attribute (which contains the URI of the link's destination). The "a" element includes a number of attributes for defining the behaviour of the presentation when the link is followed.
area	Whereas the a element only allows a link to be associated with a complete media object, the area element allows links to be associated with spatial and/or temporal portions of a media object.

The area element may be useful for enabling services that rely on interactivity where the display size is not big enough to allow the display of links alongside a media (e.g. QCIF video) window. Instead, the user could, for example, click on a watermark logo displayed in the video window to visit the company website.

Even if the area element may be useful some mobile terminals will not be able to handle area elements that include multiple selectable regions within an area element. One reason for this could be that the terminals do not have the appropriate user interface. Such area elements should therefore be avoided. Instead it is recommended that the "a" element be used. If the "area" element is used, the SMIL presentation should also include alternative links to navigate through the presentation; i.e. the author should not create presentations that rely on that the player can handle "area" elements.

B.3 BasicLayout

The "fit" attribute defines how different media should be fitted into their respective display regions.

The rendering and layout of some objects on a small display might be difficult and all mobile devices may not support features such as scroll bars; in addition, the root-layout window may represent the full screen of the display. Therefore "fit=scroll" should not be used.

Due to hardware restrictions in mobile devices, operations such that scaling of a video sequence, or even images, may be very difficult to achieve. According to the SMIL 2.0 specification SMIL players may in these situations clip the content instead. To be sure of that the presentation is displayed as the author intended, content should be encoded in a size suitable for the targeted terminals ~~intended~~ and it is recommended to use "fit=hidden".

B.4 EventTiming

The two attributes "endEvent" and "repeatEvent" in the EventTiming module may cause problems for a mobile SMIL player. The end of a media element triggers the "endEvent". In the same way the "repeatEvent" occurs when the second and subsequent iterations of a repeated element begin playback. Both these events rely on that the SMIL player receives information about that the media element has ended. One example could be when the end of a video sequence initiates the event. If the player has not received explicit information about the duration of the video sequence, e.g. by the "dur" attribute in SMIL or by some external source as the "a=range" field in SDP. The player will have to rely on the RTCP BYE message to decide when the video sequence ends. If the RTCP BYE message is lost, the player will have problems initiate the event. For these reasons is recommended that the "endEvent" and "repeatEvent" attributes are used with care, and if used the player should be provided with some additional information about the duration of the media element that triggers the event. This additional information could e.g. be the "dur" attribute in SMIL or the "a=range" field in SDP.

The "inBoundsEvent" and "outOfBoundsEvent" attributes assume that the terminal has a pointer device for moving the focus to within a window (i.e. clicking within a window). Not all terminals will support this functionality since they do not have the appropriate user interface. Hence care should be taken in using these particular event triggers.

B.5 MetaInformation

Authors are encouraged to make use of meta data whenever providing such information to the mobile terminal appears to be useful. However, they should keep in mind that some mobile terminals will parse but not process the meta data.

Furthermore, authors should keep in mind that excessive use of meta data will substantially increase the file size of the SMIL presentation that needs to be transferred to the mobile terminal. This may result in longer set-up times.

B.6 XML entities

Entities are a mechanism to insert XML fragments inside an XML document. Entities can be internal, essentially a macro expansion, or external. Use of XML entities in SMIL presentations is not recommended, as many current XML parsers do not fully support them.

B.7 XHTML Mobile Profile~~XHTML Basic~~

When rendering texts in a SMIL presentation, authors are able to use XHTML Mobile Profile [47] that contains thirteen modules. However, some of the modules include non-text information. When referring to an XHTML Mobile Profile document from a SMIL document, authors should use only the required XHTML Host Language modules : Structure Module, Text Module, Hypertext Module and List Module. The use of the Image Module, in particular, should not be used. Images and other non-text contents should be included in the SMIL document.

NOTE: An XHTML file including a module which is not part of the XHTML Host Language modules may not be shown as intended. Also, an XHTML file which uses elements or attributes from the required XHTML Host Language modules and which uses elements or attributes that are not included in XHTML Basic Profile [28], may not render correctly on legacy handsets which implement only XHTML Basic. These are:

- The start attribute on the 'ol' element in the List module
- The value attribute on the 'li' element in the List module
- The 'b' element in the Presentation module
- The 'big' element in the Presentation module
- The 'hr' element in the Presentation module

- The 'i' element in the Presentation module
- The 'small' element in the Presentation module

~~When rendering texts in a SMIL presentation, authors are able to use XHTML Basic that contains eleven modules. However, some of the modules include non-text information. When referring to an XHTML Basic document from a SMIL document, authors should use only~~ ~~*the required XHTML Host Language modules*~~ ~~: Structure Module, Text Module, Hypertext Module and List Module. The use of the Image Module, in particular, should not be used. Images and other non-text contents should be included in the SMIL document.~~

~~Note: An XHTML file Including a module which is not part of the XHTML Host Language modules may not be shown as intended.~~

Annex C (normative): MIME media types

C.1 MIME media type H263-2000

MIME media type name: video
MIME subtype name: H263-2000

Required parameters: None

Optional parameters:
profile: H.263 profile number, in the range 0 through 8, specifying the supported H.263 annexes/subparts.
level: Level of bitstream operation, in the range 0 through 99, specifying the level of computational complexity of the decoding process. When no profile and level parameters are specified, Baseline Profile (Profile 0) level 10 are the default values.

The profile and level specifications can be found in [23]. Note that the RTP payload format for H263-2000 is the same as for H263-1998 and is defined in [14], but additional annexes/subparts are specified along with the profiles and levels.

NOTE: The above text will be replaced with a reference to the RFC describing the H263-2000 MIME media type as soon as this becomes available.

C.2 MIME media type sp-midi

MIME media type name: audio
MIME subtype name: sp-midi

Required parameters: none

Optional parameters: none

NOTE: The above text will be replaced with a reference to the RFC describing the sp-midi MIME media type as soon as this becomes available.

Annex D (normative): Support for non-ISO code streams in MP4 files

D.1 General

The purpose of this annex is to define the necessary structure for integration of the H.263, AMR and AMR-WB media specific information in an MP4 file. Clauses D.2 to D.4 give some background information about the Sample Description atom, VisualSampleEntry atom and the AudioSampleEntry atom in the MPEG-4 file format. Then, the definitions of the SampleEntry atoms for AMR, AMR-WB and H.263 are given in clauses D.5 to D.8.

AMR and AMR-WB data is stored in the stream according to the AMR and AMR-WB storage format for single channel header of [11],without the AMR magic numbers.

D.2 Sample Description atom

In an MP4 file, Sample Description Atom gives detailed information about the coding type used, and any initialisation information needed for that coding. The Sample Description Atom can be found in the MP4 Atom Structure Hierarchy shown in figure D.1.

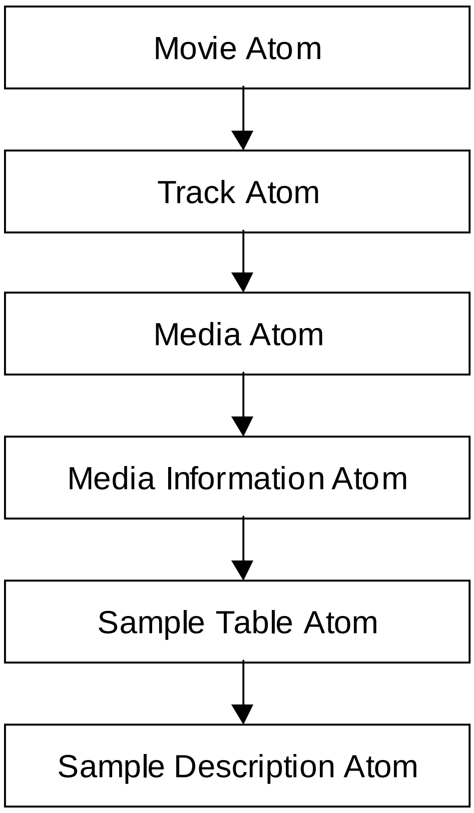


Figure D.1: MP4 Atom Structure Hierarchy

The Sample Description Atom can have one or more SampleDescriptionEntry fields. Valid Sample Description Entry atoms already defined for MP4 are AudioSampleEntry, VisualSampleEntry, HintSampleEntry and MPEGSampleEntry

Atoms. The Sample-DescriptionEntry Atoms for AMR and AMR-WB shall be AMRSampleEntry, and for H.263 shall be H263SampleEntry, respectively.

The format of SampleDescriptionEntry and its fields are explained as follows:

SampleDescriptionEntry ::= VisualSampleEntry |

AudioSampleEntry |

HintSampleEntry |

MpegSampleEntry

H263SampleEntry |

AMRSampleEntry

Table D.1: SampleDescriptionEntry fields

Field	Type	Details	Value
VisualSampleEntry		Entry type for visual samples defined in the MPEG-4 specification.	
AudioSampleEntry		Entry type for audio samples defined in the MPEG-4 specification.	
HintSampleEntry		Entry type for hint track samples defined in the MPEG-4 specification.	
MpegSampleEntry		Entry type for MPEG related stream samples defined in the MPEG-4 specification.	
H263SampleEntry		Entry type for H.263 visual samples defined in clause D.6 of the present document.	
AMRSampleEntry		Entry type for AMR and AMR-WB speech samples defined in clause D.5 of the present document.	

From the above 5 atoms, only the VisualSampleEntry, AudioSampleEntry, H263SampleEntry and AMRSampleEntry atoms are taken into consideration, since MPEG specific streams and hint tracks are out of the scope of the present document.

D.3 VisualSampleEntry atom

The VisualSampleEntry Atom is defined as follows:

VisualSampleEntry ::= AtomHeader

Reserved_6

Data-reference-index

Reserved_16

Width

Height

Reserved_4

Reserved_4

Reserved_4

Reserved_2

Reserved_32

Reserved_2

Reserved_2

ESDAtom

Table D.2: VisualSampleEntry fields

Field	Type	Details	Value
AtomHeader.Size	Unsigned int(32)		
AtomHeader.Type	Unsigned int(32)		'mp4v'
Reserved_6	Unsigned int(8) [6]		0
Data-reference-index	Unsigned int(16)	Index to a data reference that to use to retrieve the sample data. Data references are stored in data reference Atoms.	
Reserved_16	Const unsigned int(32) [4]		0
Width	Unsigned int(16)	Maximum width, in pixels of the stream	
Height	Unsigned int(16)	Maximum height, in pixels of the stream	
Reserved_4	Const unsigned int(32)		0x00480000
Reserved_4	Const unsigned int(32)		0x00480000
Reserved_4	Const unsigned int(32)		0
Reserved_2	Const unsigned int(16)		1
Reserved_32	Const unsigned int(8) [32]		0
Reserved_2	Const unsigned int(16)		24
Reserved_2	Const int(16)		-1
ESDAtom		Atom containing an elementary stream descriptor for this stream.	

The stream type specific information is in the ESDAtom structure, which will be explained later.

This version of the VisualSampleEntry, with explicit width and height, shall be used for MPEG-4 video streams conformant to this specification.

NOTE: width and height parameters together may be used to allocate the necessary memory in the playback device without need to analyse the video stream.

D.4 AudioSampleEntry atom

AudioSampleEntryAtom is defined as follows:

AudioSampleEntry ::= AtomHeader

Reserved_6

Data-reference-index

Reserved_8

Reserved_2

Reserved_2

Reserved_4

TimeScale

Reserved_2

ESDAtom

Table D.3: AudioSampleEntry fields

Field	Type	Details	Value
AtomHeader.Size	Unsigned int(32)		
AtomHeader.Type	Unsigned int(32)		'mp4a'
Reserved_6	Unsigned int(8) [6]		0
Data-reference-index	Unsigned int(16)	Index to a data reference that to use to retrieve the sample data. Data references are stored in data reference Atoms.	
Reserved_8	Const unsigned int(32) [2]		0
Reserved_2	Const unsigned int(16)		2
Reserved_2	Const unsigned int(16)		16
Reserved_4	Const unsigned int(32)		0
TimeScale	Unsigned int(16)	Copied from track	
Reserved_2	Const unsigned int(16)		0
ESDAtom		Atom containing an elementary stream descriptor for this stream.	

The stream type specific information is in the ESDAtom structure, which will be explained later.

D.5 AMRSampleEntry atom

For narrow-band AMR, the atom type of the AMRSampleEntry Atom shall be 'samr'. For AMR wide-band (AMR-WB), the atom type of the AMRSampleEntry Atom shall be 'sawb'.

The AMRSampleEntry Atom is defined as follows:

AMRSampleEntry ::= AtomHeader

Reserved_6

Data-reference-index

Reserved_8

Reserved_2

Reserved_2

Reserved_4

TimeScale

Reserved_2

AMRSpecificAtom

Table D.4: AMRSampleEntry fields

Field	Type	Details	Value
AtomHeader.Size	Unsigned int(32)		
AtomHeader.Type	Unsigned int(32)		'samr' or ‘sawb’
Reserved_6	Unsigned int(8) [6]		0
Data-reference-index	Unsigned int(16)	Index to a data reference that to use to retrieve the sample data. Data references are stored in data reference Atoms.	
Reserved_8	Const unsigned int(32) [2]		0
Reserved_2	Const unsigned int(16)		2
Reserved_2	Const unsigned int(16)		16
Reserved_4	Const unsigned int(32)		0
TimeScale	Unsigned int(16)	Copied from media header atom of this media	
Reserved_2	Const unsigned int(16)		0
AMRSpecificAtom		Information specific to the decoder.	

If one compares the AudioSampleEntry Atom - AMRSampleEntry Atom the main difference is in the replacement of the ESDAtom, which is specific to MPEG-4 systems, with an atom suitable for AMR and AMR-WB. The **AMRSpecificAtom** field structure is described in clause D.7.

D.6 H263SampleEntry atom

The atom type of the H263SampleEntry Atom shall be 's263'.

The H263SampleEntry Atom is defined as follows:

H263SampleEntry ::= AtomHeader

Reserved_6

Data-reference-index

Reserved_16

Width

Height

Reserved_4

Reserved_4

Reserved_4

Reserved_2

Reserved_32

Reserved_2

Reserved_2

H263SpecificAtom

Table D.5: H263SampleEntry fields

Field	Type	Details	Value
AtomHeader.Size	Unsigned int(32)		
AtomHeader.Type	Unsigned int(32)		's263'
Reserved_6	Unsigned int(8) [6]		0
Data-reference-index	Unsigned int(16)	Index to a data reference that to use to retrieve the sample data. Data references are stored in data reference Atoms.	
Reserved_16	Const unsigned int(32) [4]		0
Width	Unsigned int(16)	Maximum width, in pixels of the stream	
Height	Unsigned int(16)	Maximum height, in pixels of the stream	
Reserved_4	Const unsigned int(32)		0x00480000
Reserved_4	Const unsigned int(32)		0x00480000
Reserved_4	Const unsigned int(32)		0
Reserved_2	Const unsigned int(16)		1
Reserved_32	Const unsigned int(8) [32]		0
Reserved_2	Const unsigned int(16)		24
Reserved_2	Const int(16)		-1
H263SpecificAtom		Information specific to the H.263 decoder.	

If one compares the VisualSampleEntry – H263SampleEntry Atom the main difference is in the replacement of the ESDAtom, which is specific to MPEG-4 systems, with an atom suitable for H.263. The **H263SpecificAtom** field structure for H.263 is described in clause D.8.

D.7 AMRSpecificAtom field for AMRSampleEntry atom

The AMRSpecificAtom fields for AMR and AMR-WB shall be as defined in table D.6. The AMRSpecificAtom for the AMRSampleEntry Atom shall always be included if the MP4 file contains AMR or AMR-WB media.

Table D.6: The AMRSpecificAtom fields for AMRSampleEntry

Field	Type	Details	Value
AtomHeader.Size	Unsigned int(32)		
AtomHeader.Type	Unsigned int(32)		‘damr’
DecSpecificInfo	AMRDecSpecStruc	Structure which holds the AMR and AMR-WB Specific information	

AtomHeader Size and Type: indicate the size and type of the AMR decoder-specific atom. The type must be ‘damr’.

DecSpecificInfo: the structure where the AMR and AMR-WB stream specific information resides.

The AMRDecSpecStruc is defined as follows:

struct **AMRDecSpecStruc**{

Unsigned int (32)	**vendor**
Unsigned int (8)	**decoder_version**
Unsigned int (16)	**mode_set**
Unsigned int (8)	**mode_change_period**
Unsigned int (8)	**frames_per_sample**

}

The definitions of AMRDecSpecStruc members are as follows:

vendor: four character code of the manufacturer of the codec, e.g. 'VXYZ'. The vendor field gives information about the vendor whose codec is used to create the encoded data. It is an informative field which may be used by the decoding end. If a manufacturer already has a four character code, it is recommended that it uses the same code in this field. Else, it is recommended that the manufacturer creates a four character code which best addresses the manufacturer's name. It can be safely ignored.

decoder_version: version of the vendor's decoder which can decode the encoded stream in the best (i.e. optimal) way. This field is closely tied to the vendor field. It may give advantage to the vendor which has optimal encoder-decoder version pairs. The value is set to 0 if decoder version has no importance for the vendor. It can be safely ignored.

mode_set: the active codec modes. Each bit of the mode_set parameter corresponds to one mode. The bit index of the mode is calculated according to the 4 bit FT field of the AMR or AMR-WB frame structure. The mode_set bit structure is as follows: (B15xxxxxxB8B7xxxxxxB0) where B0 (Least Significant Bit) corresponds to Mode 0, and B8 corresponds to Mode 8.

The mapping of existing AMR modes to FT is given in table 1.a in [19]. A value of 0x81FF means all modes and comfort noise frames are possibly present in an AMR stream.

The mapping of existing AMR-WB modes to FT is given in Table 1.a in TS 26.201 [37]. A value of 0x83FF means all modes and comfort noise frames are possibly present in an AMR-WB stream.

As an example, if mode_set = 0000000110010101b, only Modes 0, 2, 4, 7 and 8 are present in the stream.

mode_change_period: defines a number N, which restricts the mode changes only at a multiple of N frames. If no restriction is applied, this value should be set to 0. If mode_change_period is not 0, the following restrictions apply to it according to the frames_per_sample field:

```
if (mode_change_period < frames_per_sample)
    frames_per_sample = k x (mode_change_period)
else if (mode_change_period > frames_per_sample)
    mode_change_period = k x (frames_per_sample)
where k : integer [2, …]
```

If mode_change_period is equal to frames_per_sample, then the mode is the same for all frames inside one sample.

frames_per_sample: defines the number of frames to be considered as 'one sample' inside the MP4 file. This number shall be greater than 0 and less than 16. A value of 1 means each frame is treated as one sample. A value of 10 means that 10 frames (of duration 20 msec each) are put together and treated as one sample. It must be noted that, in this case, one sample duration is 20 (msec/frame) x 10 (frame) = 200 msec. For the last sample of the stream, the number of frames can be smaller than frames_per_sample, if the number of remaining frames is smaller than frames_per_sample.

NOTE: The "hinter", for the creation of the hint tracks, can use the information given by the AMRDecSpecStruc members.

D.8 H263SpecificAtom field for H263SampleEntry atom

The H263SpecificAtom fields for H. 263 shall be as defined in table D.7. The H263SpecificAtom for the H263SampleEntry Atom shall always be included if the MP4 file contains H.263 media.

The H263SpecificAtom for H263 is composed of the following fields.

Table D.7: The H263SpecificAtom fields H263SampleEntry

Field	Type	Details	Value
AtomHeader.Size	Unsigned int(32)		
AtomHeader.Type	Unsigned int(32)		'd263'
DecSpecificInfo	H263DecSpecStruc	Structure which holds the H.263 Specific information	

AtomHeader Size and Type: indicate the size and type of the H.263 decoder-specific atom. The type must be 'd263'.

DecSpecificInfo: This is the structure where the H263 stream specific information resides.

H263DecSpecStruc is defined as follows:

struct **H263DecSpecStruc**{

Unsigned int (32)	**vendor**
Unsigned int (8)	**decoder_version**
Unsigned int (8)	**H263_Level**
Unsigned int (8)	**H263_Profile**

}

The definitions of H263DecSpecStruc members are as follows:

vendor: four character code of the manufacturer of the codec, e.g. 'VXYZ'. The vendor field gives information about the vendor whose codec is used to create the encoded data. It is an informative field which may be used by the decoding end. If a manufacturer already has a four character code, it is recommended that it uses the same code in this field. Else, it is recommended that the manufacturer creates a four character code which best addresses the manufacturer's name. It can be safely ignored.

decoder_version: version of the vendor's decoder which can decode the encoded stream in the best (i.e. optimal) way. This field is closely tied to the vendor field. It may give advantage to the vendor which has optimal encoder-decoder version pairs. . The value is set to 0 if decoder version has no importance for the vendor. It can be safely ignored.

H263_Level and H263_Profile: These two parameters define which H263 profile and level is used. These parameters are based on the MIME media type video/H263-2000. The profile and level specifications can be found in [23].

EXAMPLE 1: H.263 Baseline = {H263_Level = 10, H263_Profile = 0}

EXAMPLE 2: H.263 Profile 3 @ Level 10 = {H263_Level = 10 , H263_Profile = 3}

NOTE: The "hinter", for the creation of the hint tracks, can use the information given by the H263DecSpecStruc members.

D.8a Timed Text Format

This clause defines the format of timed text in downloaded files. In this release, timed text is downloaded, not streamed.

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.

D.8a.1 Unicode Support

Text in this specification uses the Unicode 3.0 [30] 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.

D.8a.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.

D.8a.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.

D.8a.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.

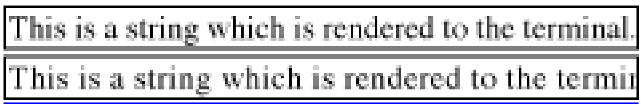
Terminals shall support unstyled text for those characters it supports. It may also support bold, italic (oblique) and bold-italic. If a style is requested which the terminal does not support, it should substitute a supported style; a character shall be rendered if the terminal has that character in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

D.8a.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.

D.8a.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.

D.8a.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.

The text box 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:

```
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;
    template int(16) layer = 0;
    template int(16) alternate_group = 0;
    template int(16)    volume = {if track_is_audio 0x0100 else 0};
    const unsigned int(16)  reserved = 0;
    template int(32)[9]     matrix=
        { 0x00010000,0,0,0,0x00010000,0,0,0,0x40000000 };
        // unity matrix
    template unsigned int(32) width =
        {if track_is_visual 0x01400000 else 0};
    template unsigned int(32) height =
        {if track_is_visual 0x00F00000 else 0};
}
```

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, track_height set to 20, 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 D.2 provides an overview:

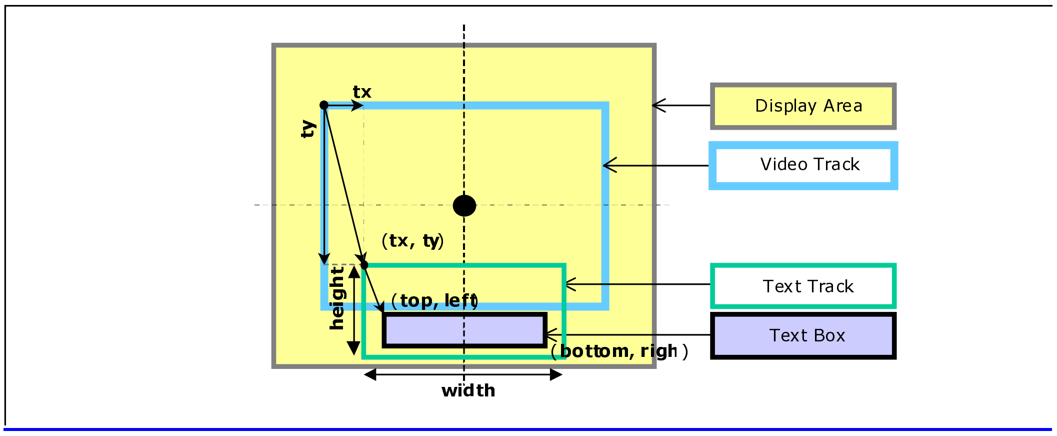


Figure D.2: Illustration of text rendering position and composition

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 (MP4) 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.

D.8a.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:

a) 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;

b) 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:

0x00 –	text is vertically scrolled up (‘credits style’), entering from the bottom of the bottom and leaving towards the top.
0x01 –	text is horizontally scrolled (‘marquee style’), entering from the right and leaving towards the left.
0x10 –	text is vertically scrolled down, entering from the top and leaving towards the bottom.
0x11 –	text is horizontally scrolled, entering from the left and leaving towards the right.

D.8a.9 Language

The human language used in this stream is declared by the language field of the media-header atom 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.

D.8a.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 GHIdefABC 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.

D.8a.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, text is not wrapped in this specification. If 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 [48]. Terminals should treat carriage return (\u000D), next line (\u0085) and CR+LF (\u000D\u000A) as new line.

D.8a.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.

D.8a.13 Media Handler

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

D.8a.14 Media Handler Header

The 3G text track uses an empty null media header (‘nmhd’), called Mpeg4MediaHeaderAtom in the MP4 specification, in common with other MPEG streams.

```
aligned(8) class  Mpeg4MediaHeaderAtom
    extends FullAtom(’nmhd’, version = 0, flags) {
 }
```

D.8a.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.
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.

D.8a.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	

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 high is {0, 0, 50, 100}.

A font table shall follow these fields, to define the complete set of fonts used. The font table is an atom 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 series of font names, 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.

D.8a.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. 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.

D.8a.17.1 Sample Modifier Boxes

D.8a.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];
}
```

D.8a.17.1.2 Highlight

'hlit' - Specifies highlighted text: the atom 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.

D.8a.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 atom 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 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.

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

At most one dynamic highlight ('krok') atom 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;
}
```

D.8a.17.1.4 Scroll Delay

'dlay' - 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 ('dlay') {
    unsigned int(32)    scroll-delay;
}
```

D.8a.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];
}
```

D.8a.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;
}
```

D.8a.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;
}
```

D.8a.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).

Table D.9 details the effects of multiple options:

Table D.9: Combinations of features

			First sample modifier atom				
		Sample description style record	styl	hlit	krok	href	blnk
Second sample modifier atom	styl	1	3				
	hlit			3			
	krok			4	3		
	href	2	2		5	3	
	blnk		6	6	6	6	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. The characters specified by the highlight record are highlighted all the time; the Karaoke highlighting is applied at the selected times. This may be visually confusing and is not recommended.
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.

D.9 File Identification

3GPP multimedia files can be identified using several mechanisms. When stored in traditional computer file systems, these files should be given the file extension ".3gp" (readers should allow mixed case for the alphabetic characters). The MIME types "video/3gpp" (for visual~~video~~ or audio/visual~~video~~ content, where visual includes both video and timed text) and "audio/3gpp" (for purely audio content) are expected to be registered and used.

A file-type atom, as defined in the JPEG 2000 specification [36] shall be present in conforming files. The file type box 'ftyp' shall occur before any variable-length box (e.g. movie, free space, media data). Only a fixed-size box such as a file signature, if required, may precede it.

The brand identifier for this specification is '3gp5'. This brand identifier must occur in the compatible brands list, and may also be the primary brand. If the file is also conformant to release 4 of this specification, it is recommended that the Release 4 brand '3gp4' also occur in the compatible brands list; if 3gp4 is not in the compatible brand list the file will not be processed by a Release 4 reader. Readers should check the compatible brands list for the identifiers they recognize, and not rely on the file having a particular primary brand, for maximum compatibility. Files may be compatible with more than one brand, and have a 'best use' other than this specification, yet still be compatible with this specification.~~The brand identifier for this specification is '3gp4'. This brand identifier must occur in the compatible brands list, and may also be the primary brand. Readers should check the compatible brands list for this identifier, and not rely on the file having a primary brand of '3gp4', for maximum compatibility. Files may be compatible with more than one brand, and have a 'best use' other than this specification, yet still be compatible with this specification.~~

Table D.8: The File-Type atom

Field	Type	Details	Value
AtomHeader.Size	Unsigned int(32)		
AtomHeader.Type	Unsigned int(32)		'ftyp'
Brand	Unsigned int(32)	The major or 'best use' of this file	
MinorVersion	Unsigned int(32)		
CompatibleBrands	Unsigned int(32)	A list of brands, to end of the atom	

Brand: Identifies the 'best use' of this file. The brand should match the file extension. For files with extension '.3gp' and conforming to this specification, the brand shall be '3gp5~~4~~'.

MinorVersion: This identifies the minor version of the brand. For files with brand '3gpZ', where Z is a digit, and conforming to release Z.x.y, this field takes the value x*256 + y. ~~This identifies the minor version of the brand. For files with brand '3gp4', and conforming to release 4.x.y, this field takes the value x*256 + y.~~

CompatibleBrands: a list of brand identifiers (to the end of the atom). '3gp5~~4~~' shall be a member of this list.

Annex E (normative): RTP payload format and file storage format for AMR and AMR-WB audio

This section specifies the AMR and AMR-WB speech codec RTP payload, storage format and MIME type registration. It is identical to "draft-ietf-avt-rtp-amr-12.txt". All references in the text in this Annex refer to the reference list in the end of the Annex.

NOTE: The intention is to replace this normative annex with the IETF RFC defining the AMR and AMR-WB RTP payload and MIME media type registration when the RFC is available.

E.1. Introduction

This document specifies the payload format for packetization of AMR and AMR-WB encoded speech signals into the Real-time Transport Protocol (RTP) [8]. The payload format supports transmission of multiple channels, multiple frames per payload, the use of fast codec mode adaptation, robustness against packet loss and bit errors, and interoperation with existing AMR and AMR-WB transport formats on non-IP networks, as described in Section E.3.

The payload format itself is specified in Section E.4. A related file format is specified in Section E.5 for transport of AMR and AMR-WB speech data in storage mode applications such as email. In Section E.8, two separate MIME type registrations are provided, one for AMR and one for AMR-WB.

Even though this RTP payload format definition supports the transport of both AMR and AMR-WB speech, it is important to remember that AMR and AMR-WB are two different codecs and they are always handled as different payload types in RTP.

E.2. Conventions and Acronyms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [5].

The following acronyms are used in this document:

3GPP	- the Third Generation Partnership Project
AMR	- Adaptive Multi-Rate Codec
AMR-WB	- Adaptive Multi-Rate Wideband Codec
CMR	- Codec Mode Request
CN	- Comfort Noise
DTX	- Discontinuous Transmission
ETSI	- European Telecommunications Standards Institute
FEC	- Forward Error Correction
SCR	- Source Controlled Rate Operation
SID	- Silence Indicator (the frames containing only CN parameters)
VAD	- Voice Activity Detection
UED	- Unequal Error Detection
UEP	- Unequal Error Protection

The term "frame-block" is used in this document to describe the time-synchronized set of speech frames in a multi-channel AMR or AMR-WB session. In particular, in an N-channel session, a frame-block will contain N speech frames, one from each of the channels, and all N speech frames represents exactly the same time period.

E.3. Background on AMR/AMR-WB and Design Principles

AMR and AMR-WB were originally designed for circuit-switched mobile radio systems. Due to their flexibility and robustness, they are also suitable for other real-time speech communication services over packet-switched networks such as the Internet.

Because of the flexibility of these codecs, the behavior in a particular application is controlled by several parameters that select options or specify the acceptable values for a variable. These options and variables are described in general terms at appropriate points in the text of this specification as parameters to be established through out-of-band means. In Section E.8, all of the parameters are specified in the form of MIME subtype registrations for the AMR and AMR-WB encodings. The method used to signal these parameters at session setup or to arrange prior agreement of the participants is beyond the scope of this document; however, Section E.8.3 provides a mapping of the parameters into the Session Description Protocol (SDP) [11] for those applications that use SDP.

E.3.1. The Adaptive Multi-Rate (AMR) Speech Codec

The AMR codecs was originally developed and standardized by the European Telecommunications Standards Institute (ETSI) for GSM cellular systems. It is now chosen by the Third Generation Partnership Project (3GPP) as the mandatory codec for third generation (3G) cellular systems [1].

The AMR codec is a multi-mode codec that supports 8 narrow band speech encoding modes with bit rates between 4.75 and 12.2 kbps. The sampling frequency used in AMR is 8000 Hz and the speech encoding is performed on 20 ms speech frames. Therefore, each encoded AMR speech frame represents 160 samples of the original speech.

Among the 8 AMR encoding modes, three are already separately adopted as standards of their own. Particularly, the 6.7 kbps mode is adopted as PDC-EFR [14], the 7.4 kbps mode as IS-641 codec in TDMA [13], and the 12.2 kbps mode as GSM-EFR [12].

E.3.2. The Adaptive Multi-Rate Wideband (AMR-WB) Speech Codec

The Adaptive Multi-Rate Wideband (AMR-WB) speech codec [3] was originally developed by 3GPP to be used in GSM and 3G cellular systems.

Similar to AMR, the AMR-WB codec is also a multi-mode speech codec. AMR-WB supports 9 wide band speech coding modes with respective bit rates ranging from 6.6 to 23.85 kbps. The sampling frequency used in AMR-WB is 16000 Hz and the speech processing is performed on 20 ms frames. This means that each AMR-WB encoded frame represents 320 speech samples.

E.3.3. Multi-rate Encoding and Mode Adaptation

The multi-rate encoding (i.e., multi-mode) capability of AMR and AMR-WB is designed for preserving high speech quality under a wide range of transmission conditions.

With AMR or AMR-WB, mobile radio systems are able to use available bandwidth as effectively as possible. E.g. in GSM it is possible to dynamically adjust the speech encoding rate during a session so as to continuously adapt to the varying transmission conditions by dividing the fixed overall bandwidth between speech data and error protective coding to enable best possible trade-off between speech compression rate and error tolerance. To perform mode adaptation, the decoder (speech receiver) needs to signal the encoder (speech sender) the new mode it prefers. This mode change signal is called Codec Mode Request or CMR.

Since in most sessions speech is sent in both directions between the two ends, the mode requests from the decoder at one end to the encoder at the other end are piggy-backed over the speech frames in the reverse direction. In other words, there is no out-of-band signaling needed for sending CMRs.

Every AMR or AMR-WB codec implementation is required to support all the respective speech coding modes defined by the codec and must be able to handle mode switching to any of the modes at any time. However, some transport systems may impose limitations in the number of modes supported and how often the mode can change due to bandwidth limitations or other constraints. For this reason, the decoder is allowed to indicate its acceptance of a particular mode or a subset of the defined modes for the session using out-of-band means.

For example, the GSM radio link can only use a subset of at most four different modes in a given session. This subset can be any combination of the 8 AMR modes for an AMR session or any combination of the 9 AMR-WB modes for an AMR-WB session.

Moreover, for better interoperability with GSM through a gateway, the decoder is allowed to use out-of-band means to set the minimum number of frames between two mode changes and to limit the mode change among neighboring modes only.

Section E.8 specifies a set of MIME parameters that may be used to signal these mode adaptation controls at session setup.

E.3.4. Voice Activity Detection and Discontinuous Transmission

Both codecs support voice activity detection (VAD) and generation of comfort noise (CN) parameters during silence periods. Hence, the codecs have the option to reduce the number of transmitted bits and packets during silence periods to a minimum. The operation of sending CN parameters at regular intervals during silence periods is usually called discontinuous transmission (DTX) or source controlled rate (SCR) operation. The AMR or AMR-WB frames containing CN parameters are called Silence Indicator (SID) frames. See more details about VAD and DTX functionality in [9] and [10].

E.3.5. Support for Multi-Channel Session

Both the RTP payload format and the storage format defined in this document support multi-channel audio content (e.g., a stereophonic speech session).

Although AMR and AMR-WB codecs themselves do not support encoding of multi-channel audio content into a single bit stream, they can be used to separately encode and decode each of the individual channels.

To transport (or store) the separately encoded multi-channel content, the speech frames for all channels that are framed and encoded for the same 20 ms periods are logically collected in a frame-block.

At the session setup, out-of-band signaling, e.g., using the rtpmap attribute in SDP, must be used to indicate the number of channels in the session and the order of the speech frames from different channels in each frame-block.

When using SDP for signaling, the number and order of channels carried in each frame-block are specified in Section 4.1 in [24].

E.3.6. Unequal Bit-error Detection and Protection

The speech bits encoded in each AMR or AMR-WB frame have different perceptual sensitivity to bit errors. This property has been exploited in cellular systems to achieve better voice quality by using unequal error protection and detection (UEP and UED) mechanisms.

The UEP/UED mechanisms focus the protection and detection of corrupted bits to the perceptually most sensitive bits in an AMR or AMR-WB frame. In particular, speech bits in an AMR or AMR-WB frame are divided into class A, B, and C, where bits in class A are most sensitive and bits in class C least sensitive (see Table E.1 below for AMR and [4] for AMR-WB). A frame is only declared damaged if there are bit errors found in the most sensitive bits, i.e., the class A bits. On the other hand, it is acceptable to have some bit errors in the other bits, i.e., class B and C bits.

```
                    Class A   total speech

   Index   Mode       bits       bits
```

0	AMR 4.75	42	95
1	AMR 5.15	49	103
2	AMR 5.9	55	118
3	AMR 6.7	58	134
4	AMR 7.4	61	148
5	AMR 7.95	75	159
6	AMR 10.2	65	204
7	AMR 12.2	81	244
8	AMR SID	39	39

Table E.1. The number of class A bits for the AMR codec.

Moreover, a damaged frame is still useful for error concealment at the decoder since some of the less sensitive bits can still be used. This approach can improve the speech quality compared to discarding the damaged frame.

E.3.6.1. Applying UEP and UED in an IP Network

To take full advantage of the bit-error robustness of the AMR and AMR-WB codec, the RTP payload format is designed to facilitate UEP/UED in an IP network. It should be noted however that the utilization of UEP and UED discussed below is OPTIONAL.

UEP/UED in an IP network can be achieved by detecting bit errors in class A bits and tolerating bit errors in class B/C bits of the AMR or AMR-WB frame(s) in each RTP payload.

Today there exist some link layers that do not discard packets with bit errors, e.g. SLIP and some wireless links. With the Internet traffic pattern shifting towards a more multimedia-centric one, more link layers of such nature may emerge in the future. With transport layer support for partial checksums, for example those supported by UDP-Lite [15] (work in progress), bit error tolerant AMR and AMR-WB traffic could achieve better performance over these types of links.

There are at least two basic approaches for carrying AMR and AMR-WB traffic over bit error tolerant IP networks:

1) Utilizing a partial checksum to cover headers and the most important speech bits of the payload. It is recommended that at least all class A bits are covered by the checksum.

2) Utilizing a partial checksum to only cover headers, but a frame CRC to cover the class A bits of each speech frame in the RTP payload.

In either approach, at least part of the class B/C bits are left without error-check and thus bit error tolerance is achieved.

Note, it is still important that the network designer pay attention to the class B and C residual bit error rate. Though less sensitive to errors than class A bits, class B and C bits are not insignificant and undetected errors in these bits cause degradation in speech quality. An example of residual error rates considered acceptable for AMR in UMTS can be found in [20] and for AMR-WB in [21].

The application interface to the UEP/UED transport protocol (e.g., UDP-Lite) may not provide any control over the link error rate, especially in a gateway scenario. Therefore, it is incumbent upon the designer of a node with a link interface of this type to choose a residual bit error rate that is low enough to support applications such as AMR encoding when transmitting packets of a UEP/UED transport protocol.

Approach 1 is a bit efficient, flexible and simple way, but comes with two disadvantages, namely, a) bit errors in protected speech bits will cause the payload to be discarded, and b) when transporting multiple frames in a payload there is the possibility that a single bit error in protected bits will cause all the frames to be discarded.

These disadvantages can be avoided, if needed, with some overhead in the form of a frame-wise CRC (Approach 2). In problem a), the CRC makes it possible to detect bit errors in class A bits and use the frame for error concealment, which gives a small improvement in speech quality. For b), when transporting multiple frames in a payload, the CRCs remove the possibility that a single bit error in a class A bit will cause all the frames to be discarded. Avoiding that gives an improvement in speech quality when transporting multiple frames over links subject to bit errors.

The choice between the above two approaches must be made based on the available bandwidth, and desired tolerance to bit errors. Neither solution is appropriate to all cases. Section E.8 defines parameters that may be used at session setup to select between these approaches.

E.3.7. Robustness against Packet Loss

The payload format supports several means, including forward error correction (FEC) and frame interleaving, to increase robustness against packet loss.

E.3.7.1. Use of Forward Error Correction (FEC)

The simple scheme of repetition of previously sent data is one way of achieving FEC. Another possible scheme which is more bandwidth efficient is to use payload external FEC, e.g., RFC2733 [19], which generates extra packets containing repair data. The whole payload can also be sorted in sensitivity order to support external FEC schemes using UEP. There is also a work in progress on a generic version of such a scheme [18] that can be applied to AMR or AMR-WB payload transport.

With AMR or AMR-WB, it is possible to use the multi-rate capability of the codec to send redundant copies of the same mode or of another mode, e.g. one with lower-bandwidth. We describe such a scheme next.

This involves the simple retransmission of previously transmitted frame-blocks together with the current frame-block(s). This is done by using a sliding window to group the speech frame-blocks to send in each payload. Figure E.1 below shows us an example.

```
--+--------+-------+--------+-------+-------+-------+--------+--

  | f(n-2) | f(n-1) |  f(n)  | f(n+1) | f(n+2) | f(n+3) | f(n+4) |

--+--------+-------+--------+-------+-------+-------+--------+--


  <---- p(n-2) ---->

           <---- p(n-1) ---->

                    <----- p(n) ----->

                              <---- p(n+1) ---->

                                       <---- p(n+2) ---->

                                                <---- p(n+3) ---->
```

Figure E.1: An example of redundant transmission.

In this example each frame-block is retransmitted one time in the following RTP payload packet. Here, f(n-2)..f(n+4) denotes a sequence of speech frame-blocks and p(n-2)..p(n+3) a sequence of payload packets.

The use of this approach does not require signaling at the session setup. In other words, the speech sender can choose to use this scheme without consulting the receiver. This is because a packet containing redundant frames will not look different from a packet with only new frames. The receiver may receive multiple copies or versions (encoded with different modes) of a frame for a certain timestamp if no packet is lost. If multiple versions of the same speech frame are received, it is recommended that the mode with the highest rate be used by the speech decoder.

This redundancy scheme provides the same functionality as the one described in RFC 2198 "RTP Payload for Redundant Audio Data" [24]. In most cases the mechanism in this payload format is more efficient and simpler than requiring both endpoints to support RFC 2198 in addition. There are two situations in which use of RFC 2198 is indicated: if the spread in time required between the primary and redundant encodings is larger than 5 frame times, the bandwidth overhead of RFC 2198 will be lower; or, if a non-AMR codec is desired for the redundant encoding, the AMR payload format won't be able to carry it.

The sender is responsible for selecting an appropriate amount of redundancy based on feedback about the channel, e.g. in RTCP receiver reports. A sender should not base selection of FEC on the CMR, as this parameter most probably was set based on none-IP information, e.g. radio link performance measures. The sender is also responsible for avoiding congestion, which may be exacerbated by redundancy (see Section E.6 for more details).

E.3.7.2. Use of Frame Interleaving

To decrease protocol overhead, the payload design allows several speech frame-blocks be encapsulated into a single RTP packet. One of the drawbacks of such approach is that in case of packet loss this means loss of several consecutive speech frame-blocks, which usually causes clearly audible distortion in the reconstructed speech. Interleaving of frame-blocks can improve the speech quality in such cases by distributing the consecutive losses into a series of single frame-block losses. However, interleaving and bundling several frame-blocks per payload will also increase end-to-end delay and is therefore not appropriate for all types of applications. Streaming applications will most likely be able to exploit interleaving to improve speech quality in lossy transmission conditions.

This payload design supports the use of frame interleaving as an option. For the encoder (speech sender) to use frame interleaving in its outbound RTP packets for a given session, the decoder (speech receiver) needs to indicate its support via out-of-band means (see Section E.8).

E.3.8. Bandwidth Efficient or Octet-aligned Mode

For a given session, the payload format can be either bandwidth efficient or octet aligned, depending on the mode of operation that is established for the session via out-of-band means.

In the octet-aligned format, all the fields in a payload, including payload header, table of contents entries, and speech frames themselves, are individually aligned to octet boundaries to make implementations efficient. In the bandwidth efficient format only the full payload is octet aligned, so fewer padding bits are added.

Note, octet alignment of a field or payload means that the last octet is padded with zeroes in the least significant bits to fill the octet. Also note that this padding is separate from padding indicated by the P bit in the RTP header.

Between the two operation modes, only the octet-aligned mode has the capability to use the robust sorting, interleaving, and frame CRC to make the speech transport robust to packet loss and bit errors.

E.3.9. AMR or AMR-WB Speech over IP scenarios

The primary scenario for this payload format is IP end-to-end between two terminals, as shown in Figure E.2. This payload format is expected to be useful for both conversational and streaming services.

```
+----------+                          +----------+
|          |    IP/UDP/RTP/AMR or     |          |
| TERMINAL |<------------------------>| TERMINAL |
|          |    IP/UDP/RTP/AMR-WB     |          |
+----------+                          +----------+
```

Figure E.2: IP terminal to IP terminal scenario

A conversational service puts requirements on the payload format. Low delay is one very important factor, i.e. few speech frame-blocks per payload packet. Low overhead is also required when the payload format traverses low bandwidth links, especially as the frequency of packets will be high. For low bandwidth links it also an advantage to support UED which allows a link provider to reduce delay and packet loss or to reduce the utilization of link resources.

Streaming service has less strict real-time requirements and therefore can use a larger number of frame-blocks per packet than conversational service. This reduces the overhead from IP, UDP, and RTP headers. However, including several frame-blocks per packet makes the transmission more vulnerable to packet loss, so interleaving may be used to reduce the effect packet loss will have on speech quality. A streaming server handling a large number of clients also needs a payload format that requires as few resources as possible when doing packetization. The octet-aligned and interleaving modes require the least amount of resources, while CRC, robust sorting, and bandwidth efficient modes have higher demands.

Another scenario occurs when AMR or AMR-WB encoded speech will be transmitted from a non-IP system (e.g., a GSM or 3GPP network) to an IP/UDP/RTP VoIP terminal, and/or vice versa, as depicted in Figure E.3.

```
AMR or AMR-WB
over
I.366.{2,3} or +------+                        +----------+
3G Iu or       |      |   IP/UDP/RTP/AMR or    |          |
<------------->|  GW  |<---------------------->| TERMINAL |
GSM Abis       |      |   IP/UDP/RTP/AMR-WB    |          |
etc.           +------+                        +----------+
                   |
 GSM/3GPP network  |          IP network
                   |
```

Figure E.3: GW to VoIP terminal scenario

In such a case, it is likely that the AMR or AMR-WB frame is packetized in a different way in the non-IP network and will need to be re-packetized into RTP at the gateway. Also, speech frames from the non-IP network may come with some UEP/UED information (e.g., a frame quality indicator) that will need to be preserved and forwarded on to the decoder along with the speech bits. This is specified in Section E.4.3.2.

AMR's capability to do fast mode switching is exploited in some non-IP networks to optimize speech quality. To preserve this functionality in scenarios including a gateway to an IP network, a codec mode request (CMR) field is needed. The gateway will be responsible for forwarding the CMR between the non-IP and IP parts in both directions. The IP terminal should follow the CMR forwarded by the gateway to optimize speech quality going to the non-IP decoder. The mode control algorithm in the gateway must accommodate the delay imposed by the IP network on the response to CMR by the IP terminal.

The IP terminal should not set the CMR (see Section E.4.3.1), but the gateway can set the CMR value on frames going toward the encoder in the non-IP part to optimize speech quality from that encoder to the gateway. The gateway can alternatively set a lower CMR value, if desired, as one means to control congestion on the IP network.

A third likely scenario is that IP/UDP/RTP is used as transport between two non-IP systems, i.e., IP is originated and terminated in gateways on both sides of the IP transport, as illustrated in Figure E.4 below.

```
AMR or AMR-WB                                                  AMR or AMR-WB
over                                                           over
```

```
I.366.{2,3} or +------+                     +------+ I.366.{2,3} or
3G Iu or       |      |  IP/UDP/RTP/AMR or  |      | 3G Iu or
<------------->|  GW  |<------------------->|  GW  |<------------->
GSM Abis       |      |  IP/UDP/RTP/AMR-WB  |      | GSM Abis
etc.           +------+                     +------+ etc.
                   |                           |
 GSM/3GPP network  |          IP network       |  GSM/3GPP network
                   |                           |
```

Figure E.4: GW to GW scenario

This scenario requires the same mechanisms for preserving UED/UEP and CMR information as in the single gateway scenario. In addition, the CMR value may be set in packets received by the gateways on the IP network side. The gateway should forward to the non-IP side a CMR value that is the minimum of three values:

- the CMR value it receives on the IP side;

- the CMR value it calculates based on its reception quality on the non-IP side; and

- a CMR value it may choose for congestion control of transmission on the IP side.

The details of the control algorithm are left to the implementation.

E.4. AMR and AMR-WB RTP Payload Formats

The AMR and AMR-WB payload formats have identical structure, so they are specified together. The only differences are in the types of codec frames contained in the payload. The payload format consists of the RTP header, payload header and payload data.

E.4.1. RTP Header Usage

The format of the RTP header is specified in [8]. This payload format uses the fields of the header in a manner consistent with that specification.

The RTP timestamp corresponds to the sampling instant of the first sample encoded for the first frame-block in the packet. The timestamp clock frequency is the same as the sampling frequency, so the timestamp unit is in samples.

The duration of one speech frame-block is 20 ms for both AMR and AMR-WB. For AMR, the sampling frequency is 8 kHz, corresponding to 160 encoded speech samples per frame from each channel. For AMR-WB, the sampling frequency is 16 kHz, corresponding to 320 samples per frame from each channel. Thus, the timestamp is increased by 160 for AMR and 320 for AMR-WB for each consecutive frame-block.

A packet may contain multiple frame-blocks of encoded speech or comfort noise parameters. If interleaving is employed, the frame-blocks encapsulated into a payload are picked according to the interleaving rules as defined in Section E.4.4.1. Otherwise, each packet covers a period of one or more contiguous 20 ms frame-block intervals. In case the data from all the channels for a particular frame-block in the period is missing, for example at a gateway from some other transport format, it is possible to indicate that no data is present for that frame-block rather than breaking a multi-frame-block packet into two, as explained in Section E.4.3.2.

To allow for error resiliency through redundant transmission, the periods covered by multiple packets MAY overlap in time. A receiver MUST be prepared to receive any speech frame multiple times, either in exact duplicates, or in different AMR rate modes, or with data present in one packet and not present in another. If multiple versions of the same speech frame are received, it is RECOMMENDED that the mode with the highest rate be used by the speech decoder. A given frame MUST NOT be encoded as speech in one packet and comfort noise parameters in another.

The payload is always made an integral number of octets long by padding with zero bits if necessary. If additional padding is required to bring the payload length to a larger multiple of octets or for some other purpose, then the P bit in the RTP in the header may be set and padding appended as specified in [8]. The RTP header marker bit (M) SHALL be set to 1 if the first frame-block carried in the packet contains a speech frame which is the first in a talkspurt. For all other packets the marker bit SHALL be set to zero (M=0).

The assignment of an RTP payload type for this new packet format is outside the scope of this document, and will not be specified here. It is expected that the RTP profile under which this payload format is being used will assign a payload type for this encoding or specify that the payload type is to be bound dynamically.

E.4.2. Payload Structure

The complete payload consists of a payload header, a payload table of contents, and speech data representing one or more speech frame-blocks. The following diagram shows the general payload format layout:

```
+----------------+-------------------+----------------
| payload header | table of contents | speech data ...
+----------------+-------------------+----------------
```

Payloads containing more than one speech frame-block are called compound payloads.

The following sections describe the variations taken by the payload format depending on whether the AMR session is set up to use the bandwidth-efficient mode or octet-aligned mode and any of the OPTIONAL functions for robust sorting, interleaving, and frame CRCs. Implementations SHOULD support both bandwidth-efficient and octet-aligned operation to increase interoperability.

E.4.3. Bandwidth-Efficient Mode

E.4.3.1. The Payload Header

In bandwidth-efficient mode, the payload header simply consists of a 4 bit codec mode request:

```
 0 1 2 3
+-+-+-+-+
|  CMR  |
+-+-+-+-+
```

CMR (4 bits): Indicates a codec mode request sent to the speech encoder at the site of the receiver of this payload. The value of the CMR field is set to the frame type index of the corresponding speech mode being requested. The frame type index may be 0-7 for AMR, as defined in Table 1a in [2], or 0-8 for AMR-WB, as defined in Table 1a in [4]. CMR value 15 indicates that no mode request is present, and other values are for future use.

The mode request received in the CMR field is valid until the next CMR is received, i.e. a newly received CMR value overrides the previous one. Therefore, if a terminal continuously wishes to receive frames in the same mode X, it needs to set CMR=X for all its outbound payloads, and if a terminal has no preference in which mode to receive, it SHOULD set CMR=15 in all its outbound payloads.

If receiving a payload with a CMR value which is not a speech mode or NO_DATA, the CMR MUST be ignored by the receiver.

In a multi-channel session, CMR SHOULD be interpreted by the receiver of the payload as the desired encoding mode for all the channels in the session.

An IP end-point SHOULD NOT set the CMR based on packet losses or other congestion indications, for several reasons:

- The other end of the IP path may be a gateway to a non-IP network (such as a radio link) that needs to set the CMR field to optimize performance on that network.

- Congestion on the IP network is managed by the IP sender, in this case at the other end of the IP path. Feedback about congestion SHOULD be provided to that IP sender through RTCP or other means, and then the sender can choose to avoid congestion using the most appropriate mechanism. That may include adjusting the codec mode, but also includes adjusting the level of redundancy or number of frames per packet.

The encoder SHOULD follow a received mode request, but MAY change to a lower-numbered mode if it so chooses, for example to control congestion.

The CMR field MUST be set to 15 for packets sent to a multicast group. The encoder in the speech sender SHOULD ignore mode requests when sending speech to a multicast session but MAY use RTCP feedback information as a hint that a mode change is needed.

The codec mode selection MAY be restricted by a session parameter to a subset of the available modes. If so, the requested mode MUST be among the signalled subset (see Section E.8).

E.4.3.2. The Payload Table of Contents

The table of contents (ToC) consists of a list of ToC entries, each representing a speech frame.

In bandwidth-efficient mode, a ToC entry takes the following format:

```
  0 1 2 3 4 5

 +-+-+-+-+-+-+

 |F|  FT   |Q|

 +-+-+-+-+-+-+
```

F (1 bit): If set to 1, indicates that this frame is followed by another speech frame in this payload; if set to 0, indicates that this frame is the last frame in this payload.

FT (4 bits): Frame type index, indicating either the AMR or AMR-WB speech coding mode or comfort noise (SID) mode of the corresponding frame carried in this payload.

The value of FT is defined in Table 1a in [2] for AMR and in Table 1a in [4] for AMR-WB. FT=14 (SPEECH_LOST, only available for AMR-WB) and FT=15 (NO_DATA) are used to indicate frames that are either lost or not being transmitted in this payload, respectively.

NO_DATA (FT=15) frame could mean either that there is no data produced by the speech encoder for that frame or that no data for that frame is transmitted in the current payload (i.e., valid data for that frame could be sent in either an earlier or later packet).

If receiving a ToC entry with a FT value in the range 9-14 for AMR or 10-13 for AMR-WB the whole packet SHOULD be discarded. This is to avoid the loss of data synchronization in the depacketization process, which can result in a huge degradation in speech quality.

Note that packets containing only NO_DATA frames SHOULD NOT be transmitted. Also, frame-blocks containing only NO_DATA frames at the end of a packet SHOULD NOT be transmitted, except in the case of interleaving. The AMR SCR/DTX is described in [6] and AMR-WB SCR/DTX in [7].

The extra comfort noise frame types specified in table 1a in [2] (i.e., GSM-EFR CN, IS-641 CN, and PDC-EFR CN) MUST NOT be used in this payload format because the standardized AMR codec is only required to implement the general AMR SID frame type and not those that are native to the incorporated encodings.

Q (1 bit): Frame quality indicator. If set to 0, indicates the corresponding frame is severely damaged and the receiver should set the RX_TYPE (see [6]) to either SPEECH_BAD or SID_BAD depending on the frame type (FT).

The frame quality indicator is included for interoperability with the ATM payload format described in ITU-T I.366.2, the UMTS Iu interface [16], as well as other transport formats. The frame quality indicator enables damaged frames to be forwarded to the speech decoder for error concealment. This can improve the speech quality comparing to dropping the damaged frames. See Section E.4.4.2.1 for more details.

For multi-channel sessions, the ToC entries of all frames from a frame-block are placed in the ToC in consecutive order as defined in Section 4.1 in [24]. When multiple frame-blocks are present in a packet in bandwidth-efficient mode, they will be placed in the packet in order of their creation time.

Therefore, with N channels and K speech frame-blocks in a packet, there MUST be N*K entries in the ToC, and the first N entries will be from the first frame-block, the second N entries will be from the second frame-block, and so on.

The following figure shows an example of a ToC of three entries in a single channel session using bandwidth efficient mode.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|  FT   |Q|1|  FT   |Q|0|  FT   |Q|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Below is an example of how the ToC entries will appear in the ToC of a packet carrying 3 consecutive frame-blocks in a session with two channels (L and R).

```
+----+----+----+----+----+----+
| 1L | 1R | 2L | 2R | 3L | 3R |
+----+----+----+----+----+----+
|<------->|<------->|<------->|
  Frame-    Frame-    Frame-
  Block 1   Block 2   Block 3
```

E.4.3.3 Speech Data

Speech data of a payload contains one or more speech frames or comfort noise frames, as described in the ToC of the payload.

Note, for ToC entries with FT=14 or 15, there will be no corresponding speech frame present in the speech data.

Each speech frame represents 20 ms of speech encoded with the mode indicated in the FT field of the corresponding ToC entry. The length of the speech frame is implicitly defined by the mode indicated in the FT field. The order and numbering notation of the bits are as specified for Interface Format 1 (IF1) in [2] for AMR and [4] for AMR-WB. As specified there, the bits of speech frames have been rearranged in order of decreasing sensitivity, while the bits of comfort noise frames are in the order produced by the encoder. The resulting bit sequence for a frame of length K bits is denoted d(0), d(1), ..., d(K-1).

E.4.3.4. Algorithm for Forming the Payload

The complete RTP payload in bandwidth-efficient mode is formed by packing bits from the payload header, table of contents, and speech frames, in order as defined by their corresponding ToC entries in the ToC list, contiguously into octets beginning with the most significant bits of the fields and the octets.

To be precise, the four-bit payload header is packed into the first octet of the payload with bit 0 of the payload header in the most significant bit of the octet. The four most significant bits (numbered 0-3) of the first ToC entry are packed into the least significant bits of the octet, ending with bit 3 in the least significant bit. Packing continues in the second octet with bit 4 of the first ToC entry in the most significant bit of the octet. If more than one frame is contained in the payload, then packing continues with the second and successive ToC entries. Bit 0 of the first data frame follows immediately after the last ToC bit, proceeding through all the bits of the frame in numerical order.

Bits from any successive frames follow contiguously in numerical order for each frame and in consecutive order of the frames.

If speech data is missing for one or more speech frame within the sequence, because of, for example, DTX, a ToC entry with FT set to NO_DATA SHALL be included in the ToC for each of the missing frames, but no data bits are included in the payload for the missing frame (see Section E.4.3.5.2 for an example).

E.4.3.5. Payload Examples

E.4.3.5.1. Single Channel Payload Carrying a Single Frame

The following diagram shows a bandwidth-efficient AMR payload from a single channel session carrying a single speech frame-block.

In the payload, no specific mode is requested (CMR=15), the speech frame is not damaged at the IP origin (Q=1), and the coding mode is AMR 7.4 kbps (FT=4). The encoded speech bits, d(0) to d(147), are arranged in descending sensitivity order according to [2]. Finally, two zero bits are added to the end as padding to make the payload octet aligned.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | CMR=15|0| FT=4  |1|d(0)                                          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                     d(147)|P|P|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E.4.3.5.2. Single Channel Payload Carrying Multiple Frames

The following diagram shows a single channel, bandwidth efficient compound AMR-WB payload that contains four frames, of which one has no speech data. The first frame is a speech frame at 6.6 kbps mode (FT=0) that is composed of speech bits d(0) to d(131). The second frame is an AMR-WB SID frame (FT=9), consisting of bits g(0) to g(39). The third frame is NO_DATA frame and does not carry any speech information, it is represented in the payload by its ToC entry. The fourth frame in the payload is a speech frame at 8.85 kpbs mode (FT=1), it consists of speech bits h(0) to h(176).

As shown below, the payload carries a mode request for the encoder on the receiver's side to change its future coding mode to AMR-WB 8.85 kbps (CMR=1). None of the frames is damaged at IP origin (Q=1). The encoded speech and SID bits, d(0) to d(131), g(0) to g(39) and h(0) to h(176), are arranged in the payload in descending sensitivity order according to [4]. (Note, no speech bits are present for the third frame). Finally, seven 0s are padded to the end to make the payload octet aligned.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | CMR=1 |1| FT=0  |1|1| FT=9  |1|1| FT=15 |1|0| FT=1  |1|d(0)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                         d(131)|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |g(0)                                                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          g(39)|h(0)                                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                           h(176)|P|P|P|P|P|P|P|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E.4.3.5.3. Multi-Channel Payload Carrying Multiple Frames

The following diagram shows a two channel payload carrying 3 frame-blocks, i.e. the payload will contain 6 speech frames.

In the payload all speech frames contain the same mode 7.4 kbit/s (FT=4) and are not damaged at IP origin. The CMR is set to 15, i.e., no specific mode is requested. The two channels are defined as left (L) and right (R) in that order. The encoded speech bits is designated dXY(0).. dXY(K-1), where X = block number, Y = channel, and K is the number of speech bits for that mode. Exemplifying this, for frame-block 1 of the left channel the encoded bits are designated as d1L(0) to d1L(147).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| CMR=15|1|1L FT=4|1|1|1R FT=4|1|1|2L FT=4|1|1|2R FT=4|1|1|3L FT|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|4|1|0|3R FT=4|1|d1L(0)                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                               d1L(147)|d1R(0) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
: ...                                                           :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       d1R(147)|d2L(0)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
: ...                                                           :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|d2L(147|d2R(0)                                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
: ...                                                           :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                       d2R(147)|d3L(0)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
: ...                                                           :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               d3L(147)|d3R(0)                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
: ...                                                           :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                       d3R(147)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E.4.4. Octet-aligned Mode

E.4.4.1. The Payload Header

In octet-aligned mode, the payload header consists of a 4 bit CMR, 4 reserved bits, and optionally, an 8 bit interleaving header, as shown below:

```
    0                   1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+- - - - - - - -
   |  CMR  |R|R|R|R|  ILL  |  ILP  |
   +-+-+-+-+-+-+-+-+- - - - - - - -
```

CMR (4 bits): same as defined in section E.4.3.1.

R: is a reserved bit that MUST be set to zero. All R bits MUST be ignored by the receiver.

ILL (4 bits, unsigned integer): This is an OPTIONAL field that is present only if interleaving is signalled out-of-band for the session. ILL=L indicates to the receiver that the interleaving length is L+1, in number of frame-blocks.

ILP (4 bits, unsigned integer): This is an OPTIONAL field that is present only if interleaving is signalled. ILP MUST take a value between 0 and ILL, inclusive, indicating the interleaving index for frame-blocks in this payload in the interleave group. If the value of ILP is found greater than ILL, the payload SHOULD be discarded.

ILL and ILP fields MUST be present in each packet in a session if interleaving is signalled for the session. Interleaving MUST be performed on a frame-block basis (i.e., NOT on a frame basis) in a multi-channel session.

The following example illustrates the arrangement of speech frame-blocks in an interleave group during an interleave session. Here we assume ILL=L for the interleave group that starts at speech frame-block n. We also assume that the first payload packet of the interleave group is s and the number of speech frame-blocks carried in each payload is N. Then we will have:

Payload s (the first packet of this interleave group):

ILL=L, ILP=0, Carry frame-blocks: n, n+(L+1), n+2*(L+1), ..., n+(N-1)*(L+1)

Payload s+1 (the second packet of this interleave group):

ILL=L, ILP=1, frame-blocks: n+1, n+1+(L+1), n+1+2*(L+1), ..., n+1+(N-1)*(L+1)

...

Payload s+L (the last packet of this interleave group):

ILL=L, ILP=L, frame-blocks: n+L, n+L+(L+1), n+L+2*(L+1), ..., n+L+(N-1)*(L+1)

The next interleave group will start at frame-block n+N*(L+1).

There will be no interleaving effect unless the number of frame-blocks per packet (N) is at least 2. Moreover, the number of frame-blocks per payload (N) and the value of ILL MUST NOT be changed inside an interleave group. In other words, all payloads in an interleave group MUST have the same ILL and MUST contain the same number of speech frame-blocks.

The sender of the payload MUST only apply interleaving if the receiver has signalled its use through out-of-band means. Since interleaving will increase buffering requirements at the receiver, the receiver uses MIME parameter "interleaving=I" to set the maximum number of frame-blocks allowed in an interleaving group to I.

When performing interleaving the sender MUST use a proper number of frame-blocks per payload (N) and ILL so that the resulting size of an interleave group is less or equal to I, i.e., N*(L+1)<=I.

E.4.4.2. The Payload Table of Contents and Frame CRCs

The table of contents (ToC) in octet-aligned mode consists of a list of ToC entries where each entry corresponds to a speech frame carried in the payload and, optionally, a list of speech frame CRCs, i.e.,

```
+---------------------+
| list of ToC entries |
+---------------------+
| list of frame CRCs  | (optional)
 - - - - - - - - - - -
```

Note, for ToC entries with FT=14 or 15, there will be no corresponding speech frame or frame CRC present in the payload.

The list of ToC entries is organized in the same way as described for bandwidth-efficient mode in E.4.3.2, with the following exception; when interleaving is used the frame-blocks in the ToC will almost never be placed consecutive in time. Instead, the presence and order of the frame-blocks in a packet will follow the pattern described in E.4.4.1.

The following example shows the ToC of three consecutive packets, each carrying 3 frame-blocks, in an interleaved two-channel session. Here, the two channels are left (L) and right (R) with L coming before R, and the interleaving length is 3 (i.e., ILL=2). This makes the interleave group 9 frame-blocks large.

```
Packet #1
---------

ILL=2, ILP=0:
+----+----+----+----+----+----+
| 1L | 1R | 4L | 4R | 7L | 7R |
+----+----+----+----+----+----+
|<------->|<------->|<------->|
  Frame-    Frame-    Frame-
  Block 1   Block 4   Block 7

Packet #2
---------

ILL=2, ILP=1:
+----+----+----+----+----+----+
| 2L | 2R | 5L | 5R | 8L | 8R |
+----+----+----+----+----+----+
|<------->|<------->|<------->|
  Frame-    Frame-    Frame-
  Block 2   Block 5   Block 8


Packet #3
---------


ILL=2, ILP=2:
+----+----+----+----+----+----+
| 3L | 3R | 6L | 6R | 9L | 9R |
+----+----+----+----+----+----+
|<------->|<------->|<------->|
  Frame-    Frame-    Frame-
  Block 3   Block 6   Block 9
```

A ToC entry takes the following format in octet-aligned mode:

```
    0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+
   |F|  FT   |Q|P|P|
   +-+-+-+-+-+-+-+-+
```

F (1 bit): see definition in Section E.4.3.2.

FT (4 bits unsigned integer): see definition in Section E.4.3.2.

Q (1 bit): see definition in Section E.4.3.2.

P bits: padding bits, MUST be set to zero.

The list of CRCs is OPTIONAL. It only exists if the use of CRC is signalled out-of-band for the session. When present, each CRC in the list is 8 bit long and corresponds to a speech frame (NOT a frame-block) carried in the payload. Calculation and use of the CRC is specified in the next section.

E.4.4.2.1. Use of Frame CRC for UED over IP

The general concept of UED/UEP over IP is discussed in Section E.3.6. This section provides more details on how to use the frame CRC in the octet-aligned payload header together with a partial transport layer checksum to achieve UED.

To achieve UED, one SHOULD use a transport layer checksum, for example, the one defined in UDP-Lite [15], to protect the RTP header, payload header, and table of contents bits in a payload. The frame CRC, when used, MUST be calculated only over all class A bits in the frame. Class B and C bits in the frame MUST NOT be included in the CRC calculation and SHOULD NOT be covered by the transport checksum.

Note, the number of class A bits for various coding modes in AMR codec is specified as informative in [2] and is therefore copied into Table E.1 in Section E.3.6 to make it normative for this payload format. The number of class A bits for various coding modes in AMR-WB codec is specified as normative in table 2 in [4], and the SID frame (FT=9) has 40 class A bits. These definitions of class A bits MUST be used for this payload format.

Packets SHOULD be discarded if the transport layer checksum detects errors.

The receiver of the payload SHOULD examine the data integrity of the received class A bits by re-calculating the CRC over the received class A bits and comparing the result to the value found in the received payload header. If the two values mismatch, the receiver SHALL consider the class A bits in the receiver frame damaged and MUST clear the Q flag of the frame (i.e., set it to 0). This will subsequently cause the frame to be marked as SPEECH_BAD, if the FT of the frame is 0..7 for AMR or 0..8 for AMR-WB, or SID_BAD if the FT of the frame is 8 for AMR or 9 for AMR-WB, before it is passed to the speech decoder. See [6] and [7] more details.

The following example shows an octet-aligned ToC with a CRC list for a payload containing 3 speech frames from a single channel session (assuming none of tthe FTs is equal to 14 or 15)

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1|  FT#1 |Q|P|P|1|  FT#2 |Q|P|P|0|  FT#3 |Q|P|P|     CRC#1     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     CRC#2     |     CRC#3     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Each of the CRC's takes 8 bits

```
     0   1   2   3   4   5   6   7
   +---+---+---+---+---+---+---+---+
   | c0| c1| c2| c3| c4| c5| c6| c7|
   +---+---+---+---+---+---+---+---+
```

and is calculated by the cyclic generator polynomial,

```
     C(x) = 1 + x^2 + x^3 + x^4 + x^8
```

where ^ is the exponentiation operator.

In binary form the polynomial has the following form: 101110001 (MSB..LSB).

The actual calculation of the CRC is made as follows:

First, an 8-bit CRC register is reset to zero: 00000000. For each bit over which the CRC shall be calculated, an XOR operation is made between the rightmost bit of the CRC register and the bit. The CRC register is then right shifted one step (inputting a "0" as the leftmost bit). If the result of the XOR operation mentioned above is a "1" "10111000" is then bit-wise XOR-ed into the CRC register. This operation is repeated for each bit that the CRC should cover. In this case, the first bit would be d(0) for the speech frame for which the CRC should cover. When the last bit (e.g. d(54) for AMR 5.9 according to Table E.1 in Section E.3.6) have been used in this CRC calculation, the contents in CRC register should simply be copied to the corresponding field in the list of CRC's.

Fast calculation of the CRC on a general-purpose CPU is possible using a table-driven algorithm.

E.4.4.3. Speech Data

In octet-aligned mode, speech data is carried in a similar way to that in the bandwidth-efficient mode as discussed in Section E.4.3.3, with the following exceptions:

- The last octet of each speech frame MUST be padded with zeroes at the end if not all bits in the octet are used. In other words, each speech frame MUST be octet-aligned.

- When multiple speech frames are present in the speech data (i.e., compound payload), the speech frames can be arranged either one whole frame after another as usual, or with the octets of all frames interleaved together at the octet level.

Since the bits within each frame are ordered with the most error-sensitive bits first, interleaving the octets collects those sensitive bits from all frames to be nearer the beginning of the packet. This is called "robust sorting order" which allows the application of UED (such as UDP-Lite [15]) or UEP (such as the ULP [18]) mechanisms to the payload data. The details of assembling the payload are given in the next section.

The use of robust sorting order for a session MUST be agreed via

out-of-band means. Section E.8 specifies a MIME parameter for this purpose.

Note, robust sorting order MUST only be performed on the frame level and thus is independent of interleaving which is at the frame-block level, as described in Section E.4.4.1. In other words, robust sorting can be applied to either non-interleaved or interleaved sessions.

E.4.4.4. Methods for Forming the Payload

Two different packetization methods, namely normal order and robust sorting order, exist for forming a payload in octet-aligned mode. In both cases, the payload header and table of contents are packed into the payload the same way; the difference is in the packing of the speech frames.

The payload begins with the payload header of one octet or two if frame interleaving is selected. The payload header is followed by the table of contents consisting of a list of one-octet ToC entries. If frame CRCs are to be included, they follow the table of contents with one 8-bit CRC filling each octet. Note that if a given frame has a ToC entry with FT=14 or 15, there will be no CRC present.

The speech data follows the table of contents, or the CRCs if present. For packetization in the normal order, all of the octets comprising a speech frame are appended to the payload as a unit. The speech frames are packed in the same order as their corresponding ToC entries are arranged in the ToC list, with the exception that if a given frame has a ToC entry with FT=14 or 15, there will be no data octets present for that frame.

For packetization in robust sorting order, the octets of all speech frames are interleaved together at the octet level. That is, the data portion of the payload begins with the first octet of the first frame, followed by the first octet of the second frame, then the first octet of the third frame, and so on. After the first octet of the last frame has been appended, the cycle repeats with the second octet of each frame. The process continues for as many octets as are present in the longest frame. If the frames are not all the same octet length, a shorter frame is skipped once all octets in it have been appended. The order of the frames in the cycle will be sequential if frame interleaving is not in use, or according to the interleave pattern specified in the payload header if frame interleaving is in use. Note that if a given frame has a ToC entry with FT=14 or 15, there will be no data octets present for that frame so that frame is skipped in the robust sorting cycle.

The UED and/or UEP is RECOMMENDED to cover at least the RTP header, payload header, table of contents, and class A bits of a sorted payload. Exactly how many octets need to be covered depends on the network and application. If CRCs are used together with robust sorting, only the RTP header, the payload header, and the ToC SHOULD be covered by UED/UEP. The means to communicate to other layers performing UED/UEP the number of octets to be covered is beyond the scope of this specification.

E.4.4.5. Payload Examples

E.4.4.5.1. Basic Single Channel Payload Carrying Multiple Frames

The following diagram shows an octet aligned payload from a single channel session that carries two AMR frames of 7.95 kbps coding mode (FT=5). In the payload, a codec mode request is sent (CMR=6), requesting the encoder at the receiver's side to use AMR 10.2 kbps coding mode. No frame CRC, interleaving, or robust-sorting is in use.

```
    0                   1                   2                   3

    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | CMR=6 |R|R|R|R|1|FT#1=5 |Q|P|P|0|FT#2=5 |Q|P|P|   f1(0..7)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   f1(8..15)   |  f1(16..23)   |  ....                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   : ...                                                           :
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         ...   |f1(152..158) |P|   f2(0..7)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   f2(8..15)   |  f2(16..23)   |  ....                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   : ...                                                           :
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         ...   |f2(152..158) |P|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Note, in above example the last octet in both speech frames is padded with one 0 to make it octet-aligned.

E.4.4.5.2. Two Channel Payload with CRC, Interleaving, and Robust-sorting

This example shows an octet aligned payload from a two channel session. Two frame-blocks, each containing 2 speech frames of 7.95 kbps coding mode (FT=5), are carried in this payload,

The two channels are left (L) and right (R) with L coming before R. In the payload, a codec mode request is also sent (CMR=6), requesting the encoder at the receiver's side to use AMR 10.2 kbps coding mode.

Moreover, frame CRC and frame-block interleaving are both enabled for the session. The interleaving length is 2 (ILL=1) and this payload is the first one in an interleave group (ILP=0).

The first two frames in the payload are the L and R channel speech frames of frame-block #1, consisting of bits f1L(0..158) and f1R(0..158), respectively. The next two frames are the L and R channel frames of frame-block #3, consisting of bits f3L(0..158) and f3R(0..158), respectively, due to interleaving. For each of the four speech frames a CRC is calculated as CRC1L(0..7), CRC1R(0..7), CRC3L(0..7), and CRC3R(0..7), respectively. Finally, the payload is robust sorted.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| CMR=6 |R|R|R|R| ILL=1 | ILP=0 |1|FT#1L=5|Q|P|P|1|FT#1R=5|Q|P|P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|FT#3L=5|Q|P|P|0|FT#3R=5|Q|P|P|     CRC1L     |     CRC1R     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     CRC3L     |     CRC3R     |   f1L(0..7)   |   f1R(0..7)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   f3L(0..7)   |   f3R(0..7)   |  f1L(8..15)   |  f1R(8..15)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  f3L(8..15)   |  f3R(8..15)   |  f1L(16..23)  |  f1R(16..23)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
: ...                                                           :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| f3L(144..151) | f3R(144..151) |f1L(152..158)|P|f1R(152..158)|P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|f3L(152..158)|P|f3R(152..158)|P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Note, in above example the last octet in all the four speech frames is padded with one zero bit to make it octet-aligned.

E.4.5. Implementation Considerations

An application implementing this payload format MUST understand all the payload parameters in the out-of-band signaling used. For example, if an application uses SDP, all the SDP and MIME parameters in this document MUST be understood. This requirement ensures that an implementation always can decide if it is capable or not of communicating.

No operation mode of the payload format is mandatory to implement. The requirements of the application using the payload format should be used to determine what to implement. To achieve basic interoperability an implementation SHOULD at least implement both bandwidth-efficient and octet-aligned mode for single channel. The other operations mode: interleaving, robust sorting, frame-wise CRC in both single and multi-channel is OPTIONAL to implement.

E.5. AMR and AMR-WB Storage Format

The storage format is used for storing AMR or AMR-WB speech frames in a file or as an e-mail attachment. Multiple channel content is supported.

In general, an AMR or AMR-WB file has the following structure:

```
+------------------+
| Header           |
+------------------+
| Speech frame 1   |
+------------------+
: ...              :
+------------------+
| Speech frame n   |
+------------------+
```

Note, to preserve interoperability with already deployed implementations, single channel content uses a file header format different from that of multi-channel content.

E.5.1. Single channel Header

A single channel AMR or AMR-WB file header contains only a magic number and different magic numbers are defined to distinguish AMR from AMR-WB. The magic number for single channel AMR files MUST consist of ASCII character string:

"#!AMR\n" (or 0x2321414d520a in hexadecimal).

The magic number for single channel AMR-WB files MUST consist of ASCII character string:

"#!AMR-WB\n" (or 0x2321414d522d57420a in hexadecimal).

Note, the "\n" is an important part of the magic numbers and MUST be included in the comparison, since, otherwise, the single channel magic numbers above will become indistinguishable from those of the multi-channel files defined in the next section.

E.5.2. Multi-channel Header

The multi-channel header consists of a magic number followed by a 32 bit channel description field, giving the multi-channel header the following structure:

```
+------------------+
| magic number     |
+------------------+
| chan-desc field  |
+------------------+
```

The magic number for multi-channel AMR files MUST consist of the ASCII character string:

"#!AMR_MC1.0\n" (or 0x2321414d525F4D43312E300a in hexadecimal).

The magic number for multi-channel AMR-WB files MUST consist of the ASCII character string:

"#!AMR-WB_MC1.0\n" (or 0x2321414d522d57425F4D43312E300a in hexadecimal).

The version number in the magic numbers refers to the version of the file format.

The 32 bit channel description field is defined as:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Reserved bits                                    | CHAN  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Reserved bits: MUST be set to 0 when written, and a reader MUST ignore them.

CHAN (4 bit unsigned integer): Specifies the number and formation of audio channels contained in this storage file, as defined in the following table:

```
          |  # of   |              |         channel
     CHAN | channels| description  |  1   2   3   4   5   6
    ============================================================
      1   |  2     | stereo       |  l   r
      2   |  3     |              |  l   r   c
      3   |  4     | quadrophonic |  Fl  Fr  Rl  Rr
      4   |  4     |              |  l   c   r   S
      5   |  5     |              |  Fl  Fr  Fc  Sl  Sr
      6   |  6     |              |  l   lc  c   r   rc  S
    ------+-----------------------------------------------------
    0,7-15| Reserved for future use
    ============================================================
      Legends:
        l - left
        r - right
        c - center
        S - surround
        F - front
        R - rear
```

Table E.2: Channel definitions for the storage format

E.5.3. Speech Frames

After the file header, speech frame-blocks consecutive in time are stored in the file. Each frame-block contains a number of octet-aligned speech frames equal to the number of channels, and stored in increasing order, starting with channel 1.

Each stored speech frame starts with a one octet frame header with the following format:

```
    0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+
   |P|  FT   |Q|P|P|
   +-+-+-+-+-+-+-+-+
```

The FT field and the Q bit are defined in the same way as in Section E.4.1.2. The P bits are padding and MUST be set to 0.

Following this one octet header come the speech bits as defined in E.4.3.3. The last octet of each frame is padded with zeroes, if needed, to achieve octet alignment.

The following example shows an AMR frame in 5.9 kbit coding mode (with 118 speech bits) in the storage format.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P| FT=2  |Q|P|P|                                               |
+-+-+-+-+-+-+-+-+                                               +
|                                                               |
+          Speech bits for frame-block n, channel k             +
|                                                               |
+                                                           +-+-+
|                                                           |P|P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Frame-blocks or speech frames lost in transmission and non-received frame-blocks between SID updates during non-speech periods MUST be stored as NO_DATA frames (frame type 15, as defined in [2] and [4]) or SPEECH_LOST (frame type 14, only available for AMR-WB) in complete frame-blocks to keep synchronization with the original media.

E.6. Congestion Control

The general congestion control considerations for transporting RTP data apply to AMR or AMR-WB speech over RTP as well. However, the multi-rate capability of AMR and AMR-WB speech coding may provide an advantage over other payload formats for controlling congestion since the bandwidth demand can be adjusted by selecting a different coding mode.

Another parameter that may impact the bandwidth demand for AMR and AMR-WB is the number of frame-blocks that are encapsulated in each RTP payload. Packing more frame-blocks in each RTP payload can reduce the number of packets sent and hence the overhead from IP/UDP/RTP headers, at the expense of increased delay.

If forward error correction (FEC) is used to combat packet loss, the amount of redundancy added by FEC will need to be regulated so that the use of FEC itself does not cause a congestion problem.

It is RECOMMENDED that AMR or AMR-WB applications using this payload format employ congestion control. The actual mechanism for congestion control is not specified but should be suitable for real-time flows, e.g. "Equation-Based Congestion Control for Unicast Applications" [17].

E.7. Security Considerations

RTP packets using the payload format defined in this specification are subject to the general security considerations discussed in [8]. As this format transports encoded speech, the main security issues include confidentiality and authentication of the speech itself. The payload format itself does not have any built-in security mechanisms. External mechanisms, such as SRTP [22], MAY be used.

This payload format does not exhibit any significant non-uniformity in the receiver side computational complexity for packet processing and thus is unlikely to pose a denial-of-service threat due to the receipt of pathological data.

E.7.1. Confidentiality

To achieve confidentiality of the encoded AMR or AMR-WB speech, all speech data bits will need to be encrypted. There is less a need to encrypt the payload header or the table of contents due to 1) that they only carry information about the requested speech mode, frame type, and frame quality, and 2) that this information could be useful to some third party, e.g., quality monitoring.

As long as the AMR or AMR-WB payload is only packed and unpacked at either end, encryption may be performed after packet encapsulation so that there is no conflict between the two operations. Interleaving may affect encryption. Depending on the encryption scheme used, there may be restrictions on, for example, the time when keys can be changed. Specifically, the key change may need to occur at the boundary between interleave groups.

The type of encryption method used may impact the error robustness of the payload data. The error robustness may be severely reduced when the data is encrypted unless an encryption method without error-propagation is used, e.g. a stream cipher. Therefore, UED/UEP based on robust sorting may be difficult to apply when the payload data is encrypted.

E.7.2. Authentication

To authenticate the sender of the speech, an external mechanism has to be used. It is RECOMMENDED that such a mechanism protect all the speech data bits. Note that the use of UED/UEP may be difficult to combine with authentication because any bit errors will cause authentication to fail.

Data tampering by a man-in-the-middle attacker could result in erroneous depacketization/decoding that could lower the speech quality. Tampering with the CMR field may result in speech in a different quality than desired.

To prevent a man-in-the-middle attacker from tampering with the payload packets, some additional information besides the speech bits SHOULD be protected. This may include the payload header, ToC, frame CRCs, RTP timestamp, RTP sequence number, and the RTP marker bit.

E.7.3. Decoding Validation

When processing a received payload packet, if the receiver finds that the calculated payload length, based on the information of the session and the values found in the payload header fields, does not match the size of the received packet, the receiver SHOULD discard the packet. This is because decoding a packet that has errors in its length field could severely degrade the speech quality.

E.8. Payload Format Parameters

This section defines the parameters that may be used to select optional features of the AMR and AMR-WB payload formats. The parameters are defined here as part of the MIME subtype registrations for the AMR and AMR-WB speech codecs. A mapping of the parameters into the Session Description Protocol (SDP) [11] is

also provided for those applications that use SDP. Equivalent parameters could be defined elsewhere for use with control protocols that do not use MIME or SDP.

Two separate MIME registrations are made, one for AMR and one for AMR-WB, because they are distinct encodings that must be distinguished by the MIME subtype.

The data format and parameters are specified for both real-time transport in RTP and for storage type applications such as e-mail attachments.

E.8.1. AMR MIME Registration

The MIME subtype for the Adaptive Multi-Rate (AMR) codec is allocated from the IETF tree since AMR is expected to be a widely used speech codec in general VoIP applications. This MIME registration covers both real-time transfer via RTP and non-real-time transfers via stored files. Note, any unspecified parameter MUST be ignored by the receiver.

Media Type name: audio

Media subtype name: AMR

Required parameters: none

Optional parameters:

These parameters apply to RTP transfer only.

octet-align: Permissible values are 0 and 1. If 1, octet-aligned operation SHALL be used. If 0 or if not present, bandwidth efficient operation is employed.

mode-set: Requested AMR mode set. Reestricts the active codec mode set to a subset of all modes. Possible values are a comma separated list of modes from the set: 0,...,7 (see Table 1a [2]). If such mode set is specified by the decoder, the encoder MUST abide by the request and MUST NOT use modes outside of the subset. If not present, all codec modes are allowed for the session.

mode-change-period: Specifies a number of frame-blocks, N, that is the interval at which codec mode changes are allowed. The initial phase of the interval is arbitrary, but changes must be separated by multiples of N frame-blocks. If this parameter is not present, mode changes are allowed at any time during the session.

mode-change-neighbor: Permissible values are 0 and 1. If 1, mode changes SHALL only be made to the neighboring modes in the active codec mode set. Neighboring modes are the ones closest in bit rate to the current mode, either the next higher or next lower rate. If 0 or if not present, change between any two modes in the active codec mode set is allowed.

maxptime: The maximum amount of media which can be encapsulated in a payload packet, expressed as time in milliseconds. The time is calculated as the sum of the time the media present in the packet represents. The time SHOULD be a multiple of the frame size. If this parameter is not present, the sender MAY encapsulate any number of speech frames into one RTP packet.

crc: Permissible values are 0 and 1. If 1, frame CRCs SHALL be included in the payload, otherwise not. If crc=1, this also implies automatically that octet-aligned operation SHALL be used for the session.

robust-sorting: Permissible values are 0 and 1. If 1, the payload SHALL employ robust payload sorting. If 0 or if not present, simple payload sorting SHALL be used. If robust-sorting=1, this also implies automatically that octet-aligned operation SHALL be used for the session.

interleaving: Indicates that frame-block level interleaving SHALL be used for the session and its value defines the maximum number of frame-blocks allowed in an interleaving group (see Section E.4.4.1). If this parameter is not present, interleaving SHALL not be used. The presence of this parameter also implies automatically that octet-aligned operation SHALL be used.

ptime: see RFC2327 [11].

channels: The number of audio channels. The possible values and their respective channel order is specified in section 4.1 in [24]. If omitted it has the default value of 1.

Encoding considerations:

This type is defined for transfer via both RTP (RFC1889) and stored-file methods as described in Sections 4 and 5, respectively, of RFC XXXX. Audio data is binary data, and must be encoded for non-binary transport; the Base64 encoding is suitable for Email.

Security considerations: See Section 7 of RFC XXXX.

Public specification: Please refer to Section 11 of RFC XXXX.

Additional information:

The following applies to stored-file transfer methods:

Magic numbers:

single channel: ASCII character string "#!AMR\n" (or 0x2321414d520a in hexadecimal)

multi-channel: ASCII character string "#!AMR_MC1.0\n" (or 0x2321414d525F4D43312E300a in hexadecimal)

File extensions: amr, AMR

Macintosh file type code: none

Object identifier or OID: none

Person & email address to contact for further information:

johan.sjoberg@ericsson.com

ari.lakaniemi@nokia.com

Intended usage: COMMON.

It is expected that many VoIP applications (as well as mobile applications) will use this type.

Author/Change controller:

johan.sjoberg@ericsson.com

ari.lakaniemi@nokia.com

IETF Audio/Video transport working group

E.8.2. AMR-WB MIME Registration

The MIME subtype for the Adaptive Multi-Rate Wideband (AMR-WB) codec is allocated from the IETF tree since AMR-WB is expected to be a widely used speech codec in general VoIP applications. This MIME registration covers both real-time transfer via RTP and non-real-time transfers via stored files.

Note, any unspecified parameter MUST be ignored by the receiver.

Media Type name: audio

Media subtype name: AMR-WB

Required parameters: none

Optional parameters:

These parameters apply to RTP transfer only.

octet-align: Permissible values are 0 and 1. If 1, octet-aligned operation SHALL be used. If 0 or if not present, bandwidth efficient operation is employed.

mode-set: Requested AMR-WB mode set. Restricts the active codec mode set to a subset of all modes. Possible values are a comma separated list of modes from the set: 0,...,8 (see Table 1a [4]). If such mode set is specified by the decoder, the encoder MUST abide by the request and MUST NOT use modes outside of the subset. If not present, all codec modes are allowed for the session.

mode-change-period: Specifies a number of frame-blocks, N, that is the interval at which codec mode changes are allowed. The initial phase of the interval is arbitrary, but changes must be separated by multiples of N frame-blocks. If this parameter is not present, mode changes are allowed at any time during the session.

mode-change-neighbor: Permissible values are 0 and 1. If 1, mode changes SHALL only be made to the neighboring modes in the active codec mode set. Neighboring modes are the ones closest in bit rate to the current mode, either the next higher or next lower rate. If 0 or if not present, change between any two modes in the active codec mode set is allowed.

maxptime: The maximum amount of media which can be encapsulated in a payload packet, expressed as time in milliseconds. The time is calculated as the sum of the time the media present in the packet represents. The time SHOULD be a multiple of the frame size. If this parameter is not present, the sender MAY encapsulated any number of speech frames into one RTP packet.

crc: Permissible values are 0 and 1. If 1, frame CRCs SHALL be included in the payload, otherwise not. If crc=1, this also implies automatically that octet-aligned operation SHALL be used for the session.

robust-sorting: Permissible values are 0 and 1. If 1, the payload SHALL employ robust payload sorting. If 0 or if not present, simple payload sorting SHALL be used. If robust-sorting=1, this also implies automatically that octet-aligned operation SHALL be used for the session.

interleaving: Indicates that frame-block level interleaving SHALL be used for the session and its value defines the maximum number of frame-blocks allowed in an interleaving group (see Section E.4.4.1). If this parameter is not present, interleaving SHALL not be used. The presence of this parameter also implies automatically that octet-aligned operation SHALL be used.

ptime: see RFC2327 [11].

channels: The number of audio channels. The possible values and their respective channel order is specified in section 4.1 in [24]. If omitted it has the default value of 1.

Encoding considerations:

This type is defined for transfer via both RTP (RFC1889) and stored-file methods as described in Sections 4 and 5, respectively, of RFC XXXX. Audio data is binary data, and must be encoded for non-binary transport; the Base64 encoding is suitable for Email.

Security considerations: See Section 7 of RFC XXXX.

Public specification: Please refer to Section 11 of RFC XXXX.

Additional information:

The following applies to stored-file transfer methods:

Magic numbers:

single channel:

ASCII character string "#!AMR-WB\n" (or 0x2321414d522d57420a in hexadecimal)

multi-channel:

ASCII character string "#!AMR-WB_MC1.0\n" (or 0x2321414d522d57425F4D43312E300a in hexadecimal)

File extensions: awb, AWB

Macintosh file type code: none

Object identifier or OID: none

Person & email address to contact for further information:

johan.sjoberg@ericsson.com

ari.lakaniemi@nokia.com

Intended usage: COMMON.

It is expected that many VoIP applications (as well as mobile applications) will use this type.

Author/Change controller:

johan.sjoberg@ericsson.com

ari.lakaniemi@nokia.com

IETF Audio/Video transport working group

E.8.3. Mapping MIME Parameters into SDP

The information carried in the MIME media type specification has a specific mapping to fields in the Session Description Protocol (SDP) [11], which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing the AMR or AMR-WB codec, the mapping is as follows:

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

- The MIME subtype (payload format name) goes in SDP "a=rtpmap" as the encoding name. The RTP clock rate in "a=rtpmap" MUST be 8000 for AMR and 16000 for AMR-WB, and the encoding parameters (number of channels) MUST either be explicitly set to N or omitted, implying a default value of 1. The values of N that are allowed is specified in Section 4.1 in [24].

- The parameters "ptime" and "maxptime" go in the SDP "a=ptime" and "a=maxptime" attributes, respectively.

- Any remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the MIME media type string as a semicolon separated list of parameter=value pairs.

Some example SDP session descriptions utilizing AMR and AMR-WB encodings follow. In these examples, long a=fmtp lines are folded to meet the column width constraints of this document; the backslash ("\") at the end of a line and the carriage return that follows it should be ignored.

Example of usage of AMR in a possible GSM gateway scenario:

```
m=audio 49120 RTP/AVP 97

a=rtpmap:97 AMR/8000/1

a=fmtp:97 mode-set=0,2,5,7; mode-change-period=2; \

mode-change-neighbor=1

a=maxptime:20
```

Example of usage of AMR-WB in a possible VoIP scenario:

```
m=audio 49120 RTP/AVP 98

a=rtpmap:98 AMR-WB/16000

a=fmtp:98 octet-align=1
```

Example of usage of AMR-WB in a possible streaming scenario (two channel stereo):

```
m=audio 49120 RTP/AVP 99

a=rtpmap:99 AMR-WB/16000/2

a=fmtp:99 interleaving=30

a=maxptime:100
```

Note that the payload format (encoding) names are commonly shown in upper case. MIME subtypes are commonly shown in lower case. These names are case-insensitive in both places. Similarly, parameter names are case-insensitive both in MIME types and in the default mapping to the SDP a=fmtp attribute.

E.9. IANA Considerations

Two new MIME subtypes are to be registered, see Section E.8. A new SDP attribute "maxptime", also defined in Section E.8, needs to be registered. The "maxptime" attribute is expected to be defined in the revision of RFC 2327 [11] and is added here with a consistent definition.

E.10. Acknowledgements

The authors would like to thank Petri Koskelainen, Bernhard Wimmer, Tim Fingscheidt, Sanjay Gupta, Stephen Casner, and Colin Perkins for their significant contributions made throughout the writing and reviewing of this document.

E.11. References

[1] 3GPP TS 26.090, "Adaptive Multi-Rate (AMR) speech transcoding", version 4.0.0 (2001-03), 3rd Generation Partnership Project (3GPP).

[2] 3GPP TS 26.101, "AMR Speech Codec Frame Structure", version 4.1.0 (2001-06), 3rd Generation Partnership Project (3GPP).

[3] 3GPP TS 26.190 "AMR Wideband speech codec; Transcoding functions", version 5.0.0 (2001-03), 3rd Generation Partnership Project (3GPP).

[4] 3GPP TS 26.201 "AMR Wideband speech codec; Frame Structure", version 5.0.0 (2001-03), 3rd Generation Partnership Project (3GPP).

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

[6] 3GPP TS 26.093, "AMR Speech Codec; Source Controlled Rate operation", version 4.0.0 (2000-12), 3rd Generation Partnership Project (3GPP).

[7] 3GPP TS 26.193 "AMR Wideband Speech Codec; Source Controlled Rate operation", version 5.0.0 (2001-03), 3rd Generation Partnership Project (3GPP).

[8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", IETF RFC 1889, January 1996.

[9] GSM 06.92, "Comfort noise aspects for Adaptive Multi-Rate (AMR) speech traffic channels", version 7.1.1 (1999-12), European Telecommunications Standards Institute (ETSI).

[10] 3GPP TS 26.192 "AMR Wideband speech codec; Comfort Noise aspects", version 5.0.0 (2001-03), 3rd Generation Partnership Project (3GPP).

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

[24] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control" IETF RFC 1890, January 1996.

E.11.1 Informative References

[12] GSM 06.60, "Enhanced Full Rate (EFR) speech transcoding", version 8.0.1 (2000-11), European Telecommunications Standards Institute (ETSI).

[13] ANSI/TIA/EIA-136-Rev.C, part 410 - "TDMA Cellular/PCS – Radio Interface, Enhanced Full Rate Voice Codec (ACELP)." Formerly IS-641. TIA published standard, June 1 2001.

[14] ARIB, RCR STD-27H, "Personal Digital Cellular Telecommunication System RCR Standard", Association of Radio Industries and Businesses (ARIB).

[15] Lars-Ake Larzon, Mikael Degermark, and Stephen Pink, "The UDP Lite Protocol", IETF Draft (Work in Progress), February 23, 2001.

[16] 3GPP TS 25.415 "UTRAN Iu Interface User Plane Protocols", version 4.2.0 (2001-09), 3rd Generation Partnership Project (3GPP).

[17] S. Floyd, M. Handley, J. Padhye, J. Widmer, "Equation-Based Congestion Control for Unicast Applications", ACM SIGCOMM 2000, Stockholm, Sweden

[18] A. Li, et al., "An RTP Payload Format for Generic FEC with Uneven Level Protection ", IETF Draft (Work in Progress), October 2001.

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

[20] 3GPP TS 26.102, "AMR speech codec interface to Iu and Uu", version 4.0.0 (2001-03), 3rd Generation Partnership Project (3GPP).

[21] 3GPP TS 26.202 "AMR Wideband speech codec; Interface to Iu and Uu", version 5.0.0 (2001-03), 3rd Generation Partnership Project (3GPP).

[22] Baugher, et al., "The Secure Real Time Transport Protocol", IETF Draft (Work in Progress), November 2001.

[23] C. Perkins, et al., "RTP Payload for Redundant Audio Data", IETF RFC 2198, September 1997.

ETSI documents can be downloaded from the ETSI web server, "http://www.etsi.org/". Any 3GPP document can be downloaded from the 3GPP webserver, "http://www.3gpp.org/", see specifications. TIA documents can be obtained from "www.tiaonline.org".

E.12. Authors' Addresses

Johan Sjoberg Tel: +46 8 50878230
Ericsson Research EMail: Johan.Sjoberg@ericsson.com
Ericsson Radio Systems AB
SE-164 80 Stockholm, SWEDEN

Magnus Westerlund Tel: +46 8 4048287
Ericsson Research EMail: Magnus.Westerlund@ericsson.com
Ericsson Radio Systems AB
SE-164 80 Stockholm, SWEDEN

Ari Lakaniemi Tel: +358-71-8008000
Nokia Research Center EMail: ari.lakaniemi@nokia.com
P.O.Box 407
FIN-00045 Nokia Group, FINLAND

Qiaobing Xie Tel: +1-847-632-3028
Motorola, Inc. EMail: qxie1@email.mot.com
1501 W. Shure Drive, 2-B8
Arlington Heights, IL 60004, USA

Annex F (normative): RDF schema for the PSS base vocabulary

```
<?xml version="1.0"?>

<!--
     This document is the RDF Schema for streaming-specific vocabulary
     as defined in 3GPP TS 26.234 Rel.5 (in the following "the
     specification").

     The URI for unique identification of this RDF Schema is
       http://www.3gpp.org/profiles/PSS/ccppschema-PSS5

     This RDF Schema includes the same information as the respective
     chapter of the specification. Greates care has been taken to keep
     the two documents consistence. However, in case of any divergence
     the specification takes presidence.

     All reference in this RDF Schmea are to be interpreted relative to
     the specification. This means all references using the form
     [ref] are defined in chapter 2 "References of the
     specification. All other references refer to parts within that
     document.

     Note: This Schemas has been aligned in structure and base
     vocabulary to the RDF Schema used by UAProf [40].

-->

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema" >


 <!-- ****************************************************************** -->
<!-- ***** Properties shared among the components***** -->

  <rdf:Description ID="defaults">
    <rdfs:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
    <rdfs:domain rdf:resource="Streaming"/>
    <rdfs:comment>
      An attribute used to identify the default capabilities.
    </rdfs:comment>
  </rdf:Description>


<!-- ****************************************************************** -->
<!-- ***** Component Definitions ***** -->

  <rdf:Description ID="Streaming">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="http://www.wapforum.org/UAPROF/ccppschema-20010330#Component"/>
    <rdfs:label>Component: Streaming</rdfs:label>
    <rdfs:comment>
      The Streaming component specifies the base vocabulary for
      PSS. PSS servers supporting capability exchange should
      understand the attributes in this component as explained in
      detail in 3GPP TS 26.234 rel. 5.
    </rdfs:comment>
  </rdf:Description>

<!-- **
     ** In the following property definitions, the defined types
     ** are as follows:
     **
     ** Number: A positive integer
     ** [0-9]+
     ** Boolean: A yes or no value
     ** Yes|No
     ** Literal: An alphanumeric string
     ** [A-Za-z0-9/.\-_]+
     ** Dimension: A pair of numbers
     ** [0-9]+x[0-9]+
     **
-->


<!-- ****************************************************************** -->
<!-- ***** Component: Streaming ***** -->

<rdf:Description ID="AudioChannels">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: This attribute describes the stereophonic capability of the natural audio device.
The only legal values are "Mono" and "Stereo".
```

```
    Type: Literal
    Resolution: Locked
    Examples: "Mono", "Stereo"
  </rdfs:comment>
</rdf:Description>


<rdf:Description ID="VideoPreDecoderBufferSize">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: This attribute signals if the optional video
    buffering requirements defined in Annex G are supported. It also
    defines the size of the hypothetical pre-decoder buffer defined in
    Annex G. A value equal to zero means that Annex G is not
    supported. A value equal to one means that Annex G is
    supported. In this case the size of the buffer is the default size
    defined in Annex G.  A value equal to or greater than the default
    buffer size defined in Annex G means that Annex G is supported and
    sets the buffer size to the given number of octets. Legal values are all
    integer values equal to or greater than zero. Values greater than
    one but less than the default buffer size defined in Annex G are
    not allowed.

    Type: Number
    Resolution: Locked
    Examples: "0", "4096"
  </rdfs:comment>
</rdf:Description>


<rdf:Description ID="VideoInitialPostDecoderBufferingPeriod">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: If Annex G is not supported, the attribute has no
    meaning. If Annex G is supported, this attribute defines the
    maximum initial post-decoder buffering period of video. Values are
    interpreted as clock ticks of a 90-kHz clock. In other words, the
    value is incremented by one for each 1/90 000 seconds. For
    example, the value 9000 corresponds to 1/10 of a second initial
    post-decodder buffering. Legal valaues are all integer value equal
    to or greater than zero.

    Type: Number
    Resolution: Locked
    Examples: <VideoInitialPostDecoderBufferingPeriod>
                9000
          </VideoInitialPostDecoderBufferingPeriod>
  </rdfs:comment>
</rdf:Description>

<rdf:Description ID=" VideoDecodingByteRate ">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
   Description: If Annex G is not supported, the attribute has no meaning. If Annex G is supported,
   this attribute defines the peak decoding byte rate the PSS client is able to support. In other
   words, the PSS client fulfils the requirements given in Annex G with the signalled peak decoding
   byte rate. The values are given in bytes per second and shall be greater than or equal to 8000.
   According to Annex G, 8000 is the default peak decoding byte rate for the mandatory video codec
   profile and level (H.263 Profile 0 Level 10).Legal values are integer value greater than or equal
   to 8000.

    Type: Number
    Resolution: Locked
    Examples: <VideoDecodingByteRate>16000</VideoDecodingByteRate>
  </rdfs:comment>
</rdf:Description>

<rdf:Description ID=" MaxPolyphony">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: Attribute definition:  The MaxPolyphony attribute refers to the maximal polyphony
    that the synthetic audio device supports as defined in [44]. Legal values are integer between 5
    to 24.
    NOTE:       MaxPolyphony attribute can be used to signal the maximum polyphony capabilities
                supported by the PSS client. This is a complementary mechanism for the delivery of
                compatible SP-MIDI content and thus the PSS client is required to support Scalable
                Polyphony MIDI i.e. Channel Masking defined in [44].

    Type: Number
    Resolution: Locked
    Examples: <MaxPolyphony>8</MaxPolyphony>
  </rdfs:comment>
</rdf:Description>

<rdf:Description ID="PssAccept">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Bag"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: List of content types (MIME types) the PSS
    application supports. Both CcppAccept (SoftwarePlatform, UAProf)
    and PssAccept can be used but if PssAccept is defined it has
    precedence over CcppAccept and a PSS application shall then use
    PssAccept.

    Type: Literal (bag)
    Resolution: Append
    Examples: "audio/AMR-WB;octet-alignment,application/smil"
  </rdfs:comment>
</rdf:Description>


<rdf:Description ID="PssAccept-Subset">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Bag"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: List of content types for which the PSS application
    supports a subset. MIME-types can in most cases effectively be
    used to express variations in support for different media
    types. Many MIME-types, e.g. AMR-NB has several parameters that
    can be used for this purpose. There may exist content types for
    which the PSS application only supports a subset and this subset
    can not be expressed with MIME-type parameters. In these cases the
    attribute PssAccept-Subset is used to describe support for a
    subset of a specific content type. If a subset of a specific
    content type is declared in PssAccept-Subset, this means that
    PssAccept-Subset has precedence over both PssAccept and CcppAccept.
    PssAccept and/or CcppAccept shall always include the corresponding
    content types for which PSSAccept-Subset specifies subsets of.
    This is to ensure compatibility with those content servers that
    do not understand the PssAccept-Subset attribute but do understand e.g. CcppAccept.

    This is illustrated with an example. If PssAccept="audio/AMR",
    "image/jpeg" and PssAccept-Subset="JPEG-PSS" then "audio/AMR"
    and JPEG Base line is supported. "image/jpeg" in PssAccept is of no
    importance since it is related to "JPEG-PSS" in PssAccept-Subset.
    Subset identifiers and corresponding semantics shall only be defined by
    the TSG responsible for the present document. The following values are defined:
    -   "JPEG-PSS": Only the two JPEG modes described in clause 7.5 of the present
        document are supported.
    -   "SVG-Tiny"
    -   "SVG-Basic"
    Legal values are subset identifiers defined by the specification.

    Type: Literal (bag)
    Resolution: Locked
    Examples: "JPEG-PSS","SVG-Tiny","SVG-Basic"
  </rdfs:comment>
</rdf:Description>


<rdf:Description ID="PssVersion">
e  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: Latest PSS version supported by the client. Legal
    values are "3GPP-R4", "3GPP-R5" and so forth.

    Type: Literal
    Resolution: Locked
    Examples: "3GPP-R4", "3GPP-R5"
  </rdfs:comment>
</rdf:Description>


<rdf:Description ID="RenderingScreenSize">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: The rendering size of the device's screen in unit of
    pixels. The horizontal size is given followed by the vertical
    size. Legal values are pairs of integer values equal or greater
    than zero. A value equal "0x0"means that there exist no display or
    just textual output is supported.

    Type: Dimension
    Resolution: Locked
    Examples: "160x120"
  </rdfs:comment>
</rdf:Description>


<rdf:Description ID="SmilBaseSet">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: Indicates a base set of SMIL 2.0 modules that the
    client supports. Leagal values are the following pre-defined
    identifiers: "SMIL-3GPP-R4" indicates all SMIL 2.0
    modules required for scene description support according to clause
    8 of Release 4 of TS 26.234. "SMIL-3GPP-R5" indicates all SMIL 2.0
    modules required for scene description support according to clause
    8 of the specification.

    Type: Literal
    Resolution: Locked
    Examples: "SMIL-3GPP-R4", "SMIL-3GPP-R5"
  </rdfs:comment>
</rdf:Description>


<rdf:Description ID="SmilModules">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdfschema#Property"/>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Bag"/>
  <rdfs:domain rdf:resource="#Streaming"/>
  <rdfs:comment>
    Description: This attribute defines a list of SMIL 2.0 modules
    supported by the client. If the SmilBaseSet is used those modules
    do not need to be explicitly listed here. In that case only
    additional module support needs to be listed. Legal values are all
    SMIL 2.0 module names defined in the SMIL 2.0 recommendation [31],
    section 2.3.3, table 2.

    Type: Literal (bag)
    Resolution: Locked
    Examples: "BasicTransitions,MulitArcTiming"
  </rdfs:comment>
</rdf:Description>

</rdf:RDF>
```

Annex G (normative): Buffering of video

G.1 Introduction

This annex describes video buffering requirements in the PSS. As defined in clause 7.4 of the present document, support for the annex is optional and may be signalled in the PSS capability exchange and in the SDP. This is described in clause 5.2 and clause 5.3.3 of the present document. When the annex is in use, the content of the annex is normative.

In other words, PSS clients shall be capable of receiving an RTP packet stream that complies with the specified buffering model and PSS servers shall verify that the transmitted RTP packet stream complies with the specified buffering model.

G.2 PSS Buffering Parameters

The behaviour of the PSS buffering model is controlled with the following parameters: the initial pre-decoder buffering period, the initial post-decoder buffering period, the size of the hypothetical pre-decoder buffer, the peak decoding byte rate, and the decoding macroblock rate. The default values of the parameters are defined below.

- The default initial pre-decoder buffering period is 1 second.
- The default initial post-decoder buffering period is zero.
- The default size of the hypothetical pre-decoder buffer is defined according to the maximum video bit-rate according to the table below:

Table G.1: Default size of the hypothetical pre-decoder buffer

Maximum video bit-rate	**Default size of the hypothetical pre-decoder buffer**
65536 bits per second	20480 bytes
131072 bits per second	40960 bytes
Undefined	51200 bytes

- The maximum video bit-rate can be signalled in the media-level bandwidth attribute of SDP as defined in clause 5.3.3 of this document. If the video-level bandwidth attribute was not present in the presentation description, the maximum video bit-rate is defined according to the video coding profile and level in use.
- The size of the hypothetical post-decoder buffer is an implementation-specific issue. The buffer size can be estimated from the maximum output data rate of the decoders in use and from the initial post-decoder buffering period.
- By default, the peak decoding byte rate is defined according to the video coding profile and level in use. For example, H.263 Level 10 requires support for bit-rates up to 64000 bits per second. Thus, the peak decoding byte rate equals to 8000 bytes per second.
- The default decoding macroblock rate is defined according to the video coding profile and level in use. If MPEG-4 Visual is in use, the default macroblock rate equals to VCV decoder rate. If H.263 is in use, the default macroblock rate equals to (1 / minimum picture interval) multiplied by number of macroblocks in maximum picture format. For example, H.263 Level 10 requires support for picture formats up to QCIF and minimum picture interval down to 2002 / 30000 sec. Thus, the default macroblock rate would be 30000 x 99 / 2002 ≈ 1484 macroblocks per second.

PSS clients may signal their capability of providing larger buffers and faster peak decoding byte rates in the capability exchange process described in clause 5.2 of the present document. The average coded video bit-rate should be smaller than or equal to the bit-rate indicated by the video coding profile and level in use, even if a faster peak decoding byte rate were signalled.

Initial parameter values for each stream can be signalled within the SDP description of the stream. Signalled parameter values override the corresponding default parameter values. The values signalled within the SDP description guarantee pauseless playback from the beginning of the stream until the end of the stream (assuming a constant-delay reliable transmission channel).

PSS servers may update parameter values in the response for an RTSP PLAY request. If an updated parameter value is present, it shall replace the value signalled in the SDP description or the default parameter value in the operation of the PSS buffering model. An updated parameter value is valid only in the indicated playback range, and it has no effect after that. Assuming a constant-delay reliable transmission channel, the updated parameter values guarantee pauseless playback of the actual range indicated in the response for the PLAY request. The indicated pre-decoder buffer size and initial post-decoder buffering period shall be smaller than or equal to the corresponding values in the SDP description or the corresponding default values, whichever ones are valid. The following header fields are defined for RTSP:

- x-predecbufsize:<size of the hypothetical pre-decoder buffer>
  This gives the suggested size of the Annex G hypothetical pre-decoder buffer in bytes.

- x-initpredecbufperiod:<initial pre-decoder buffering period>
  This gives the required initial pre-decoder buffering period specified according to Annex G. Values are interpreted as clock ticks of a 90-kHz clock. That is, the value is incremented by one for each 1/90 000 seconds. For example, value 180 000 corresponds to a two second initial pre-decoder buffering.

- x-initpostdecbufperiod:<initial post-decoder buffering period>
  This gives the required initial post-decoder buffering period specified according to Annex G. Values are interpreted as clock ticks of a 90-kHz clock.

These header fields are defined for the response of an RTSP PLAY request only. Their use is optional.

The following example plays the whole presentation starting at SMPTE time code 0:10:20 until the end of the clip. The playback is to start at 15:36 on 23 Jan 1997. The suggested initial post-decoder buffering period is half a second.

```
     C->S: PLAY rtsp://audio.example.com/twister.en RTSP/1.0
           CSeq: 833
           Session: 12345678
           Range: smpte=0:10:20-;time=19970123T153600Z

     S->C: RTSP/1.0 200 OK
           CSeq: 833
           Date: 23 Jan 1997 15:35:06 GMT
           Range: smpte=0:10:22-;time=19970123T153600Z
           x-initpredecbufperiod: 45000
```

G.3 PSS server buffering verifier

The PSS server buffering verifier is specified according to the PSS buffering model. The model is based on two buffers and two timers. The buffers are called the hypothetical pre-decoder buffer and the hypothetical post-decoder buffer. The timers are named the decoding timer and the playback timer.

The PSS buffering model is presented below.

1. The buffers are initially empty.

2. A PSS Server adds each transmitted RTP packet having video payload to the pre-decoder buffer immediately when it is transmitted. All protocol headers at RTP or any lower layer are removed.

3. Data is not removed from the pre-decoder buffer during a period called the initial pre-decoder buffering period. The period starts when the first RTP packet is added to the buffer.

4. When the initial pre-decoder buffering period has expired, the decoding timer is started from a position indicated in the previous RTSP PLAY request.

5. Removal of a video frame is started when both of the following two conditions are met: First, the decoding timer has reached the scheduled playback time of the frame. Second, the previous video frame has been totally removed from the pre-decoder buffer.

6. The duration of frame removal is the larger one of the two candidates: The first candidate is equal to the number of macroblocks in the frame divided by the decoding macroblock rate. The second candidate is equal to the number of bytes in the frame divided by the peak decoding byte rate. When the coded video frame has been removed from the pre-decoder buffer entirely, the corresponding uncompressed video frame is located into the post-decoder buffer.

7. Data is not removed from the post-decoder buffer during a period called the initial post-decoder buffering period. The period starts when the first frame has been placed into the post-decoder buffer.

8. When the initial post-decoder buffering period has expired, the playback timer is started from the position indicated in the previous RTSP PLAY request.

9. A frame is removed from the post-decoder buffer immediately when the playback timer reaches the scheduled playback time of the frame.

10. Each RTSP PLAY request resets the PSS buffering model to its initial state.

A PSS server shall verify that a transmitted RTP packet stream complies with the following requirements:

- The PSS buffering model shall be used with the default or signalled buffering parameter values. Signalled parameter values override the corresponding default parameter values.
- The occupancy of the hypothetical pre-decoder buffer shall not exceed the default or signalled buffer size.
- Each frame shall be inserted into the hypothetical post-decoder buffer before or on its scheduled playback time.

G.4 PSS client buffering requirements

When the annex is in use, the PSS client shall be capable of receiving an RTP packet stream that complies with the PSS server buffering verifier, when the RTP packet stream is carried over a constant-delay reliable transmission channel. Furthermore, the video decoder of the PSS client, which may include handling of post-decoder buffering, shall output frames at the correct rate defined by the RTP time-stamps of the received packet stream.

Annex H (informative): Content creator guidelines for the synthetic audio medium type

It is recommended that the first element of the MIP (Maximum Instantaneous Polyphony) message of the SP-MIDI content intended for synthetic audio PSS/MMS should be no more than 5. For instance the following MIP figures {4, 9, 10, 12, 12, 16, 17, 20, 26, 26, 26} complies with the recommendation whereas {6, 9, 10, 12, 12, 16, 17, 20, 26, 26, 26} does not.

Annex I (informative): SP MIDI Device 5–24 Note Profile for 3GPP, SP-MIDI implementation guideline using a non-compliant hardware

I.1 Introduction

This informative annex describes some implementation guidelines intended for SP-MIDI device 5-24 Note Profile for 3GPP [45]. These guidelines are here to give the possibility for manufacturers to develop early SP-MIDI implementations using MIDI hardware available at the time of the approval of release 5. These guidelines are valid only for release 5 implementations of SP-MIDI and are expected to be removed . It should be noted that these guidelines may reduce the musical performance of the synthesiser depending on the content and should be used with extreme caution.

I.2 Guidelines

I.2.1 Support of multiple rhythm channels

Scalable Polyphony synthesisers conformant to this Profile shall support at least two MIDI Channels that can function as Rhythm Channels, to enable a fluent scalable polyphony implementation.

If the two rhythm Channels are not natively supported by the MIDI hardware, the SP-MIDI player could redirect the events intended to the additional rhythm channels toward the default rhythm channel (MIDI channel 10). The rendering of the SP-MIDI content should not be affected until different Channel settings (e.g. Channel Volume, Bank Setting, Panning etc.) are applied to the different rhythm Channels. It is recommended that only Channel settings intended for the default rhythm channel be applied.

I.2.2 Support of individual stereophonic panning

When the support of individual stereophonic panning is not possible by the stereophonic MIDI synthesiser, central panning should be used as default instead.

Annex J (informative): Mapping of SDP parameters to UMTS QoS parameters

This Annex gives recommendation for the mapping rules needed by the PSS applications to request the appropriate QoS from the UMTS network (see Table J.1).

Table J.1: Mapping of SDP parameters to UMTS QoS parameters for PSS

QoS parameter	**Parameter value**	**comment**
Delivery of erroneous SDUs	"no"[TBC]	
Delivery order	Yes	
Traffic class	"Streaming class"	
Maximum SDU size	1520 bytes	
Guaranteed bit rate for downlink	1.025 * SDP session bandwidth [TBC]	
Maximum bit rate for downlink	Equal or higher to guaranteed bit rate in downlink	Specifying a minimum overhead bit rate per media might be useful and is FFS
Guaranteed bit rate for uplink	0.025 * SDP session bandwidth [TBC]	
Maximum bit rate for uplink	Equal or higher to guaranteed bit rate in uplink	
Residual BER	1*10-5 [TBC]	16 bit CRC should be enough
SDU error ratio	1*10-4 or better	1*10-3 could be acceptable. RLC AM mode should easily enable 10-4.
Traffic handling priority	Subscribed traffic handling priority	Ignored
Transfer delay	[1s to 1.5s]	