3GPP2 C.S0050-B

Version 1.0

Date: 18 May 2007

3GPP2 File Formats for Multimedia Services

No Text

PREFACE

This document describes the file formats to be used in 3GPP2 Multimedia services.

1 Contents

2 List of Figures

3 List of Tables

4 Scope

The objective is to define and standardize a set of common file formats to be used in multimedia services (such as Multimedia Streaming Service (MSS) and Multimedia Messaging Service (MMS)) and to provide interoperability with existing 3G and the Internet multimedia services to the greatest extent possible. The specific media types and descriptions to be covered include: video, audio, images, graphics, high fidelity audio as well as presentation layout and synchronization.

5 References

1. Void.
2. Void.
3. ISO/IEC 14496-12:2005 "Information technology – Coding of audio-visual-objects – Part 12: ISO base media file format".
4. ISO/IEC 14496-14:2003, "Information technology – Coding of audio-visual-objects – Part 14: MP4 file format".
5. 3GPP TS 26.244, "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP) (Release 6)".
6. ISO/IEC 14496-2:2004: "Information Technology — Coding of Audio-Visual Objects – Part 2: Visual".
7. ITU-T Recommendation H.263: "Video Coding for Low Bit rate Communication"
8. 3GPP TS 26.071 "Adaptive Multi-Rate (AMR) Speech Codec; General Description"
9. 3GPP TS 26.171: "AMR Speech Codec, wideband; General Description".
10. 3GPP2 C.S0014-0-1: Enhanced Variable Rate Codec, Speech Service Option 3 for Wideband Spread Spectrum Digital Systems
11. IETF RFC 2658: "RTP Payload Format for PureVoice(tm) Audio".
12. IETF RFC 3558: "RTP Payload Format for Enhanced Variable Rate Codecs (EVRC) and Selectable Mode Vocoders (SMV)", June 2003.
13. IETF RFC 3267: "RTP payload format and file storage format for the Adaptive Multi-Rate (AMR) Adaptive Multi-Rate Wideband (AMR-WB) audio codecs", March 2002.
14. 3GPP2 C.S0020-0: "High Rate Speech Service Option 17 for Wideband Spread Spectrum Communications Systems"
15. 3GPP2 C.S0030-0 Version 2.0 Selectable Mode Vocoder Service Option for Wideband Spread Spectrum Communication Systems.
16. 3GPP TS 26.101: "Adaptive Multi-Rate (AMR) speech codec frame structure"
17. 3GPP TS 26.201: "AMR Wideband Speech Codec; Frame Structure"
18. W3C Recommendation: "Synchronized Multimedia Integration Language (SMIL 2.0)", http://www.w3.org/TR/2005/REC-SMIL2-20050107/.
19. ITU-T Recommendation H.263 (annex X): "Annex X: Profiles and levels definition".
20. IETF RFC 2234: "Augmented BNF for Syntax Specification: ABNF".
21. IETF RFC 3625: "The QCP File Format and Media Types for Speech Data".
22. ""Unicode Standard Annex #13: Unicode Newline guidelines", by Mark Davis. An integral part of "The Unicode Standard, Version 3.1.
23. "Standard MIDI Files 1.0", RP-001, "The Complete MIDI 1.0 Detailed

Specification, Document Version 96.1" The MIDI Manufacturers Association, Los Angeles, CA, USA, February 1996.

24. ITU-T Recommendation T.81: "Information technology; Digital compression and coding of continuous-tone still images: Requirements and guidelines".

25. IETF RFC 2083: "PNG (Portable Networks Graphics) Specification version 1.0".

26. Technical note TN1081: “Understanding the Differences between Apple and Windows IMA-ADPCM Compressed Sound Files”, Developer Connection, http://developer.apple.com/technotes/tn/tn1081.html

27. “Windows BMP” http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/bitmaps_5jhv.asp

28. “Windows Multimedia, Resource Interchange File Format/WAVE” http://msdn.microsoft.com/library/default.asp?url=/library/en-us/multimed/htm/_win32_resource_interchange_file_format_services.asp , http://msdn.microsoft.com/library/default.asp?url=/library/en-us/multimed/htm/_win32_about_waveform_audio.asp

29. “Complete MIDI 1.0 Detailed Specification”, v96.1, (second edition), MIDI Manufacturers Association (MMA), 2001.

30. “General MIDI 2 Specification”, v 1.1, MIDI Manufacturer Association (MMA), September 2003.

31. 3GPP2 C.S0050-0 – 3GPP2 File Formats for Multimedia Services.

32. IETF RFC4348 “Real-Time Transport Protocol (RTP) Payload Format for the Variable-Rate Multimode Wideband (VMR-WB) Audio Codec”.

33. 3GPP2 C.S0052-A v1.0 “Source-Controlled Variable-Rate Multimode Wideband Speech Codec (VMR-WB), Service Options 62 and 63 for Spread Spectrum Systems”, April 2005.

34. ISO/IEC 14496-15:2004: “Information technology – Coding of audio - visual objects – Part 15: Advanced Video Coding (AVC) file format”.

35. 3GPP TS 26.244 “The 3GPP File Format”.

36. 3GPP TS 26.245 “3GPP Timed Text”.

37. ISO/IEC 14496-10:2004: “Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding”.

38. 3GPP2 C.S0046-0 “Multimedia Streaming Service for Spread Spectrum Systems”.

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

40. ISO/IEC 14496-3:2001/Amd 1:2003, "Bandwidth Extension".

41. ISO/IEC 14496-3:2001/Amd 1:2003/Cor 1:2004.

42. IETF RFC 3711: "The Secure Real-time Transport Protocol".

43. 3GPP TS 23.032 “Universal Geographical Area Description”.

44. “Scalable polyphony MIDI specification”, Version 1.0, RP-034, The MIDI

Manufacturers Association, Los Angeles, CA, USA, 2002.

45. W3C Recommendation: "Scalable Vector Graphics (SVG) Tiny 1.2 Specification", http://www.w3.org/TR/SVGMobile12/.
46. “DLS/XMF Format for Mobile Devices”, MIDI Manufacturers Association, Los Angeles, CA, Sept. 2004. http://www.midi.org/about-midi/specshome.shtml
47. IETF RFC 4393: "MIME Type Registrations for 3GPP2 Multimedia files".
48. W3C Recommendation: REC-CSS2-19980512: "Cascading Style Sheets", level 2 CSS2 Specification 12-May-1998. http://www.w3.org/TR/REC-CSS2/
49. 3GPP TS 26.246 "3GPP SMIL Language Profile".
50. W3C Recommendation: "Cascading Style Sheets, Level 2, CSS2 Specification". http://www.w3.org/TR/REC-CSS2/
51. 3GPP2 C.S0014-B: Enhanced Variable rate Codec Speech Service option 3 and 68 for wideband spread spectrum Digital Systems.
52. 3GPP2 C.S0014-C: Enhanced Variable rate Codec Speech Service option 3, 68 and 70 for wideband spread spectrum Digital Systems.
53. IETF RFC 4788: Enhancements to RTP Payload Formats for EVRC Family Codecs, January 2007

6 Abbreviations

For the purpose of this document, the following abbreviations apply:

3G	Third Generation system
3GP	File Format for 3GPP Multimedia Services
3GPP	Third Generation Partnership Project
3GPP2	Third Generation Partnership Project 2
AAC	Advanced Audio Coding
ABNF	Augmented BNF
ADPCM	Adaptive Differential Pulse Code Modulation
AMR	Adaptive Multi-Rate
AMR-WB	Adaptive Multi-Rate Wideband
AVC	Advanced Video Coding
BNF	Backus-Naur Form
BMP	Bit Map Picture
CMF	Compact Multimedia Format
CSS2	Cascading Style Sheets, level 2
DLS	Downloadable Sound(s)
EVRC	Enhanced Variable Rate Codec
FFMS	File Formats for Multimedia Services
GAD	Geographical Area Description
HE AAC	High Efficiency AAC
HRD	Hypothetical Reference Decoder
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IMA	International Multimedia Association
IP	Internet Protocol
ISO	International Standards Organization
ITU-T	International Telecommunication Union - Telecommunication Sector
JPEG	Joint Photographic Experts Group
LED	Light Emitting Diode
MIDI	Musical Instrument Digital Interface
MIME	Multipurpose Internet Mail Extensions
MIP	Maximum Instantaneous Polyphony
MMA	MIDI Manufacturers Association

MMS	Multimedia Messaging Service
MP4	MPEG-4 File Format
MPEG	Motion Picture Experts Group
MSS	Multimedia Streaming Service
PDA	Personal Digital Assistant
PNG	Portable Network Graphics
QCELP	Qualcomm Code Excited Linear Prediction
RFC	Request for Comments
RIFF	Resource Interchange File Format
RTCP	Real-Time Control Protocol
RTP	Real-time Transport Protocol
SBR	Spectral Band Replication
SDP	Session Description Protocol
SMIL	Synchronized Multimedia Integration Language
SRTP	Secure Realtime Transport Protocol
SMV	Selectable Mode Vocoder
TCP	Transport Control Protocol
TOC	Table of Contents
URI	Uniform Resource Identifier
VMR	Variable-Rate Multimode [Wideband Vocoder]

7 Introduction

The purpose of this standard is to define a set of file formats to be used with 3GPP2 multimedia services. Among these file formats is a new format designated as the 3GPP2 file format or ".3g2" file format. It is the recommended format to use and can contain multiple media types (such as, video, audio, and timed text). Also included in this release are a presentation and layout description language and file format, ".smi", and a compact multimedia file format, ".cmf".

These file formats can be used for, but not limited to:

- multimedia content downloading to a terminal,
- multimedia file generation and uploading from the originating terminal,
- multimedia content exchange between MMS and/or MSS servers,
- multimedia content storing to a server, and
- multimedia message exchange with other industry system.

This document does not specify how this file format is used in specific services. In other words, it should be expressly mandated, if necessary, in other service specifications whether to use this specification.

8 3GPP2 File Format “.3g2”

The purpose of this section is to define the 3GPP2 file format for multimedia services. This file format is based on the ISO base media file format [3]. Also, it adopts the methodology defined in [5] to integrate necessary structures for inclusion of non-ISO codecs such as H.263 [7], AMR [8], AMR-WB [9], and extends this approach to include 3GPP2 specific codecs: EVRC [10], SMV [15], VMR-WB [33], and 13K Speech (QCELP) [14].

Currently the 3GPP2 file format also defines extensions for:

- AVC file format [8.3.3],
- Asset information [8.6],
- Encryption [8.7],
- Video buffer information [8.8].

8.1 Conformance

The 3GPP2 file format, used in the specification for timed media (such as video, audio, and timed-text), is structurally based on the ISO base media file format defined in [3]. However, the conformance statement for 3GPP2 files is defined in the present document by addressing file identification (file extension, brand identifier and MIME type definition) and registration of codecs.

NOTE: Future releases may expand the conformance statement for 3GPP2 files to include more codecs or functionalities by defining new boxes. Boxes of unknown type in the 3GPP2 file shall be ignored.

8.1.1 File identification

3GPP2 multimedia files can be identified using several mechanisms. When stored in traditional computer file systems, these files should be given the file extension “.3g2” (readers should allow mixed case for the alphabetic characters). The following MIME types should be used: “video/3gpp2” (for visual or audio/visual content, where visual includes both video and timed text) and “audio/3gpp2” (for purely audio content). [47]

A file-type box in Table 8-1, as defined in the ISO Base Media File Format [3], 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 '3g2c'. This brand identifier shall occur in the compatible brands list, and may also be the primary brand. 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.

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.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 Box	

Table 8-1: The File-Type Box

Brand: Identifies the 'best use' of this file. The brand should match the file extension. For files with extension '.3g2' and conforming to release 0 of this specification, the brand shall be '3g2a'. For files with extension ".3g2" and conforming to release A of this specification, the "brand shall be "3g2b". For files with extension ".3g2" and conforming to release B of this specification, the brand shall be "3g2c".

MinorVersion: This identifies the minor version of the brand. Files with brand '3g2*x*', where *x* is an alphabetic character, shall have a corresponding release X.Y.Z such that X = 1 when *x* = 'a'; X = 2 when *x* = 'b'; and so on. A conforming minor version value for releaseX.y.*z* uses the byte aligned and right adjusted value of release $X*256^2 + y*256+z$.

CompatibleBrands: A list of brand identifiers (to the end of the Box). '3g2c' shall be a member of this list. '3g2a' and '3g2b' shall also be members of this list if the file is in conformance with release 0 and/or release A of this specification. The brand compatibility list shall include major brands '3gp4', '3gp5' and/or '3gp6' as described in [5] when the file content meets the conditions described therein. Brands shall not be placed in the compatibility list if playback is not possible given the applicable methods. See Annex A.3 for additional information.

8.1.2 Registration of codecs

In 3GPP2 files, AVC video, MPEG-4 video, MPEG-4 AAC audio streams, and other ISO codec streams, as well as non-ISO media streams such as AMR narrow-band speech, AMR WB speech, EVRC speech, H.263 video, 13K speech, SMV speech, VMR-WB speech, and timed text, can be included as described in this specification.

8.1.3 Interpretation of 3GPP2 file format

All index numbers used in the 3GPP2 file format start with the value one rather than zero, in particular "first-chunk" in Sample to chunk box, "sample-number" in Sync sample box and "shadowed-sample-number", "sync-sample-number" in Shadow sync sample box.

8.1.4 Limitation of the ISO base media file format

The following limitation to the ISO base media file format [3] shall apply to a 3GPP2 file:

A 3GPP2 file shall be self-contained, i.e., there shall not be references to external media data from inside the 3GPP2 file.

8.2 Codec Registration

8.2.1 Overview

The purpose of this section is to give some background information about the Sample Description Box in the ISO base media file format [3]. The following sections define the necessary structures for integration of video, audio, speech, and timed text in a 3GPP2 file. This specification provides details for support of codecs defined within 3GPP2. Support for codecs not defined by 3GPP2 is provided using external references.

8.2.2 Sample Description Box

In an ISO file, Sample Description Box gives detailed information about the coding type used, and any initialization information needed for that coding. The Sample Description Box can be found in the ISO file format Box Structure Hierarchy shown in Figure 8-1.

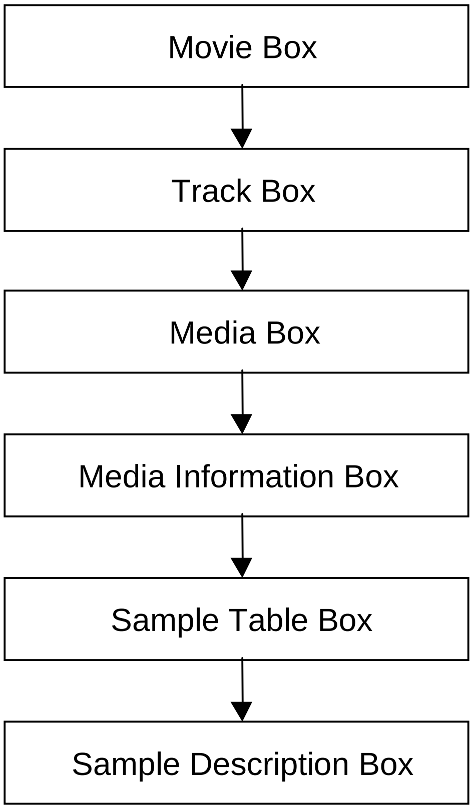


Figure 8-1: ISO File Format Box Structure Hierarchy

The Sample Description Box can have one or more Sample Entry Boxes. Valid Sample Entry Boxes already defined for ISO [10] and MP4 [10] include MP4AudioSampleEntry, MP4VisualSampleEntry, and HintSampleEntry.

In addition, the Sample Entry Box for H.263 video shall be H263SampleEntry.

The Sample Entry Box for AMR and AMR-WB speech shall be AMRSampleEntry.

The Sample Entry Box for EVRC speech shall be EVRCSampleEntry.

The Sample Entry Box for 13K (QCELP) speech shall be QCELPSampleEntry or MP4AudioSampleEntry. (Note: for 13K speech a 3g2 file parser shall be able to read both storage methods.)

The Sample Entry Box for SMV speech shall be SMVSampleEntry.

The Sample Entry Box for VMR-WB speech shall be VMRSampleEntry.

The Sample Entry Box for timed text shall be TextSampleEntry.

The Sample Entry Box for AVC shall be AVCSampleEntry.

The Sample Entry Box for EVRC-B speech shall be EVRCBSampleEntry.

The Sample Entry Box for EVRC-WB speech shall be EVRCWBSampleEntry.

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

```
SampleEntry ::=
        MP4VisualSampleEntry |
        MP4AudioSampleEntry |
        H263SampleEntry |
        AVCSampleEntry |
        AMRSampleEntry |
        EVRCSampleEntry |
        EVRCBSampleEntry |
        EVRCWBSampleEntry |
        QCELPSampleEntry |
        SMVSampleEntry |
        TextSampleEntry |
        VMRSampleEntry |
        HintSampleEntry
```

Field	Type	Details	Value
MP4VisualSampleEntry		Entry type for visual samples defined in section 8.3.1 of the present document.	
MP4AudioSampleEntry		Entry type for audio samples defined in section 8.4.1 of the present document.	
H263SampleEntry		Entry type for H.263 visual samples defined in section 8.3.2 of the present document.	
AVCSampleEntry		Entry type for AVC samples defined in section 8.3.3 of the present document.	
AMRSampleEntry		Entry type for AMR and AMR-WB speech samples defined in section 8.4.2 of the present document.	
EVRCSampleEntry		Entry type for EVRC speech samples defined in section 8.4.3 of the present document.	
EVRCBSampleEntry		Entry type for EVRC-B speech samples defined in section 8.4.4 of the present document.	
EVRCWBSampleEntry		Entry type for EVRC-WB speech samples defined in section 8.4.5 of the present document.	
QCELPSampleEntry		Entry type for 13k (QCELP) speech samples defined in section 0 of the present document.	
SMVSampleEntry		Entry type for SMV speech samples defined in section 8.4.7 of the present document.	
TextSampleEntry		Entry type for timed text samples defined in section 8.5 of the present document.	
VMRSampleEntry		Entry type for VMR-WB speech samples defined in section 8.4.8 of the present document.	
HintSampleEntry		Entry type for hint track samples defined in the ISO specification [10].	

Table 8-2: SampleEntry fields

8.3 Video

This section describes Sample Entries for video.

8.3.1 MPEG-4 Video

If MPEG-4 Video [6] is supported then it shall be supported using the MP4VisualSampleEntry Box as described in [4].

NOTE: Throughout this document MPEG-4 Visual is referred to as MPEG-4 video, which should be taken to mean encoding of natural (pixel based) video using MPEG-4 Visual methods.

8.3.2 H.263

If H.263 Video [7] is supported then it shall be supported using the H263SampleEntry Box as described in [35].

8.3.3 H.264/AVC

If MPEG-4 AVC Video [37] is supported then it shall be supported using the AVCSampleEntry Box as described in [34].

8.4 Audio and Speech

This section describes Sample Entries for audio and speech.

8.4.1 MPEG-4 AAC and HE AAC

If MPEG-4 AAC Profile or MPEG-4 HE AAC Profile [39][40][41] is supported then it shall be supported using the MP4AudioSampleEntry Box as described in [4]. When HE AAC is stored in the 3GPP2 file format, implicit signaling of SBR [39][40][41] shall not be used.

8.4.2 AMR

If AMR [16] or AMR-WB [17] speech is supported then they shall be supported using the AMRSampleEntry Box as described in [35].

8.4.3 EVRC

EVRC speech data shall be stored inside of a media track in such a way that is described in Section 11 of [12]. The magic number shall not be included. The codec data frames are stored in a consecutive order with a single TOC entry field as a prefix per each of data frame, where the TOC field is extended to one octet by setting the four most significant bits of the octet to zero, as illustrated in the following figure.

```
|<-- Octet 1 -->|<-- Octet 2 -->|<-- …… -->|<-- Octet N -->|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
|0|0|0|0|FR Type|     One EVRC speech data frame           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
```

Figure 8-2: EVRC Frame byte alignment

8.4.3.1 EVRCSampleEntry Box

For EVRC, the Box type of the EVRCSampleEntry Box shall be 'sevc'.

The EVRCSampleEntry Box is defined as follows:

```
EVRCSampleEntry ::= BoxHeader
        Reserved_6
        Data-reference-index
        Reserved_8
        Reserved_2
        Reserved_2
        Reserved_4
        TimeScale
        Reserved_2
EVRCSpecificBox
```

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'sevc'
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 Boxs.	
Reserved_8	unsigned int(32) [2]		0
Reserved_2	unsigned int(16)		2
Reserved_2	unsigned int(16)		16
Reserved_4	unsigned int(32)		0
TimeScale	Unsigned int(16)	Copied from media header Box of this media	
Reserved_2	unsigned int(16)		0
EVRCSpecificBox		Information specific to the decoder.	

Table 8-3: EVRCSampleEntry fields

If one compares the MP4AudioSampleEntry Box to the EVRCSampleEntry Box the main difference is in the replacement of the ESDBox, which is specific to MPEG-4 systems, with a box suitable for EVRC speech. The **EVRCSpecificBox** field structure is described in section 8.4.3.2.

8.4.3.2 EVRCSpecificBox field for EVRCSampleEntry Box

The EVRCSpecificBox fields for EVRC shall be as defined in Table 8-4. The EVRCSpecificBox for the EVRCSampleEntry Box shall always be included if the 3GPP2 file contains EVRC media.

Field	Type	Details	Value
BoxHeader. Size	Unsigned int(32)		
BoxHeader. Type	Unsigned int(32)		‘devc’
DecSpecificInfo	EVRCDecSpecStruc	Structure which holds the EVRC Specific information	

Table 8-4: The EVRCSpecificBox fields for EVRCSampleEntry

BoxHeader Size and Type: indicates the size and type of the EVRC decoder-specific Box. The type shall be ‘devc’.

DecSpecificInfo: the structure where the EVRC stream specific information resides. The EVRCDecSpecStruc is defined as follows:

Field	Type	Details	Value
vendor	Unsigned int(32)		
decoder_version	Unsigned int(8)		
frames_per_sample	Unsigned int(8)		

Table 8-5: EVRCDecSpecStruc

The definitions of EVRCDecSpecStruc 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 should be used in this field. Otherwise, a vendor may create a four character code which best expresses the vendor’s name. This field may be 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 associated with the vendor field. It may be used advantageously by vendors, which have optimal encoder-decoder version pairs. The value shall be set to 0 if the decoder version has no importance for the vendor. This field may be ignored.**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 should be carefully chosen since the ‘access unit’ is decided depending on the value defined by this field. For example, 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 aggregated 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.

8.4.4 EVRC-B

EVRC-B speech data shall be stored inside of a media track in such a way that is described in Section 5 of [12]. The magic number shall not be included. The codec data frames are stored in a consecutive order with a single TOC entry field as a prefix per each of data frame, where the TOC field is extended to one octet by setting the four most significant bits of the octet to zero, as illustrated in the following figure.

```
 |<-- Octet 1 -->|<-- Octet 2 -->|<-- …… -->|<-- Octet N -->|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
 |0|0|0|0|FR Type|    One EVRC-B speech data frame          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+......+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8-3: EVRC-B Frame byte alignment

8.4.4.1 EVRCBSampleEntry Box

For EVRC-B, the Box type of the EVRCBSampleEntry Box shall be 'secb'.

The EVRCBSampleEntry Box is defined as follows:

EVRCBSampleEntry ::= BoxHeader
Reserved_6
Data-reference-index
Reserved_8
Reserved_2
Reserved_2
Reserved_4
TimeScale
Reserved_2

EVRCBSpecificBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'secb'
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 Boxs.	
Reserved_8	unsigned int(32) [2]		0
Reserved_2	unsigned int(16)		2
Reserved_2	unsigned int(16)		16
Reserved_4	unsigned int(32)		0
TimeScale	Unsigned int(16)	Copied from media header Box of this media	
Reserved_2	unsigned int(16)		0
EVRCBSpecificBox		Information specific to the decoder.	

Table 8-6 : EVRCBSampleEntry fields

If one compares the MP4AudioSampleEntry Box to the EVRCBSampleEntry Box the main difference is in the replacement of the ESDBox, which is specific to MPEG-4 systems, with a box suitable for EVRC-B speech. The **EVRCBSpecificBox** field structure is described in section 8.4.4.2.

8.4.4.2 EVRCBSpecificBox field for EVRCBSampleEntry Box

The EVRCBSpecificBox fields for EVRC-B shall be as defined in Table 8-7. The EVRCBSpecificBox for the EVRCBSampleEntry Box shall always be included if the 3GPP2 file contains EVRC-B media.

Field	Type	Details	Value
BoxHeader. Size	Unsigned int(32)		
BoxHeader. Type	Unsigned int(32)		'decb'
DecSpecificInfo	EVRCBDecSpecStruc	Structure which holds the EVRC-B Specific information	

Table 8-7 : he EVRCBSpecificBox fields for EVRCBSampleEntry

BoxHeader Size and Type: indicates the size and type of the EVRC-B decoder-specific Box. The type shall be 'decb'.

DecSpecificInfo: the structure where the EVRC-B stream specific information resides. The EVRCBDecSpecStruc is defined as follows:

Field	Type	Details	Value
Vendor	Unsigned int(32)		
decoder_version	Unsigned int(8)		
frames_per_sample	Unsigned int(8)		

Table 8-8: EVRCBDecSpecStruc

The definitions of EVRCBDecSpecStruc 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 should be used in this field. Otherwise, a vendor may create a four character code which best expresses the vendor's name. This field may be 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 associated with the vendor field. It may be used advantageously by vendors, which have optimal encoder-decoder version pairs. The value shall be set to 0 if the decoder version has no importance for the vendor. This field may be ignored.

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 should be carefully chosen since the 'access unit' is decided depending on the value defined by this field. For example, 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 aggregated 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.

8.4.5 EVRC-WB

EVRC-WB speech data shall be stored inside of a media track as described in this section. The EVRC-WB encoded speech data frames are stored in a consecutive order with a single TOC entry field as a prefix per each of data frame, where the TOC field is extended to one octet by setting the four most significant bits of the octet to zero, as illustrated in the following figure.

```
|<-- Octet 1 -->|<-- Octet 2 -->|<-- ……… -->|<-- Octet N-->|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
|0|0|0|0|FR Type|     One EVRC-WB speech data frame        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
```

Figure 8-4: EVRC-WB Frame byte alignment

8.4.5.1 EVRCWBSampleEntry Box

For EVRC-WB, the Box type of the EVRCWBSampleEntry Box shall be 'secw'.

The EVRCWBSampleEntry Box is defined as follows:

EVRCWBSampleEntry ::= BoxHeader
Reserved_6
Data-reference-index
Reserved_8
Reserved_2
Reserved_2
Reserved_4
TimeScale
Reserved_2

EVRCWBSpecificBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'secw'
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 Boxs.	
Reserved_8	unsigned int(32) [2]		0
Reserved_2	unsigned int(16)		2
Reserved_2	unsigned int(16)		16
Reserved_4	unsigned int(32)		0
TimeScale	Unsigned int(16)	Copied from media header Box of this media	
Reserved_2	unsigned int(16)		0
EVRCWBSpecificBox		Information specific to the decoder.	

Table 8-9: EVRCWBSampleEntry fields

If one compares the MP4AudioSampleEntry Box to the EVRCWBSampleEntry Box the main difference is in the replacement of the ESDBox, which is specific to MPEG-4 systems, with a box suitable for EVRC-WB speech. The **EVRCWBSpecificBox** field structure is described in section 8.4.5.2.

8.4.5.2 EVRCWBSpecificBox field for EVRCWBSampleEntry Box

The EVRCWBSpecificBox fields for EVRC-WB shall be as defined in Table 8-10. The EVRCWBSpecificBox for the EVRCWBSampleEntry Box shall always be included if the 3GPP2 file contains EVRC-WB media.

Field	Type	Details	Value
BoxHeader. Size	Unsigned int(32)		
BoxHeader. Type	Unsigned int(32)		'decw'
DecSpecificInfo	EVRCWBDecSpecStruc	Structure which holds the EVRC-WB Specific information	

Table 8-10: The EVRCWBSpecificBox fields for EVRCWBSampleEntry

BoxHeader Size and Type: indicates the size and type of the EVRC-WB decoder-specific Box. The type shall be 'decw'.

DecSpecificInfo: the structure where the EVRC-WB stream specific information resides. The EVRCWBDecSpecStruc is defined as follows:

Field	Type	Details	Value
vendor	Unsigned int(32)		
decoder_version	Unsigned int(8)		
frames_per_sample	Unsigned int(8)		

Table 8-11: EVRCWBDecSpecStruc

The definitions of EVRCWBDecSpecStruc 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 should be used in this field. Otherwise, a vendor may create a four character code which best expresses the vendor's name. This field may be 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 associated with the vendor field. It may be used advantageously by vendors, which have optimal encoder-decoder version pairs. The value shall be set to 0 if the decoder version has no importance for the vendor. This field may be ignored.

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 should be carefully chosen since the 'access unit' is decided depending on the value defined by this field. For example, 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 aggregated 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.

8.4.6 13K (QCELP)

(Note: for 13K speech a 3g2 file parser shall be able to read both the QCELPSampleEntry and the MP4AudioSampleEntry storage methods.)

13K speech data shall be stored inside of a media track in the same way for codec data frame format as described in Section 3.2 of [11]. Each codec data frame is zero-padded to become of multiple of octets and the frames are stored in a consecutive order, as illustrated in the following figure. Here ‘z’ is the stuffing bit used to keep byte alignment; its value is 0.

```
|<-- Octet 1 -->|<-- Octet 2 -->|<-- …… -->|<-- Octet N -->|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
|     Rate      |     One 13K speech data frame       …|z|z|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
```

Figure 8-5: 13K (QCELP) Frame byte alignment

8.4.6.1 QCELPSampleEntry Box

For 13K, the box type of the QCELPSampleEntry Box shall be 'sqcp'.

The QCELPSampleEntry Box is defined as follows:

QCELPSampleEntry::= BoxHeader
Reserved_6
Data-reference-index
Reserved_8
Reserved_2
Reserved_2
Reserved_4
TimeScale
Reserved_2
QCELPSpecificBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'sqcp’
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 Boxs.	
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 Box of this media	
Reserved_2	Const unsigned int(16)		0
QCELPSpecificBox		Information specific to the decoder.	

Table 8-12: QCELPSampleEntry fields

If one compares the MP4AudioSampleEntry Box to the QCELPSampleEntry Box the main difference is in the replacement of the ESDBox, which is specific to MPEG-4 systems, with a box suitable for 13k. The **QCELPSpecificBox** field structure is described in Section 8.4.6.2.

8.4.6.2 QCELPSpecificBox field for QCELPSampleEntry Box

The QCELPSpecificBox fields for 13K speech shall be as defined in Table 8-13. The QCELPSpecificBox for the QCELPSampleEntry Box shall always be included if the 3GPP2 file contains 13K speech media.

Field	Type	Details	Value
BoxHeader. Size	Unsigned int(32)		
BoxHeader. Type	Unsigned int(32)		‘dqcp’
DecSpecificInfo	QCELPDecSpecStruc	Structure which holds the 13K (QCELP) speech specific information	

Table 8-13: The QCELPSpecificBox fields for QCELPSampleEntry

BoxHeader Size and Type: indicate the size and type of the 13k decoder-specific Box. The type shall be ‘dqcp’.

DecSpecificInfo: the structure where the 13K speech stream specific information resides. The QCELPDecSpecStruc is defined as follows:

Field	Type	Details	Value
vendor	Unsigned int(32)		
decoder_version	Unsigned int(8)		
frames_per_sample	Unsigned int(8)		

Table 8-14: QCELPDecSpecStruc

The definitions of QCELPDecSpecStruc 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 should be used in this field. Otherwise, a vendor may create a four character code which best expresses the vendor’s name. Else, it is recommended that the manufacturer creates a four character code which best addresses the manufacturer’s name. This field may 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 associated with the vendor field. It may be used advantageously by the vendors, which have optimal encoder-decoder version pairs. The value shall be set to 0 if the decoder version has no importance for the vendor. This field may be safely ignored.

frames_per_sample: defines the number of frames to be considered as 'one sample' inside the file. This number shall be greater than 0 and should be carefully chosen since the ‘access unit’ is decided depending on the value defined by this field. 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 aggregated 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.

8.4.6.3 13K (QCELP) Support in MP4AudioSampleEntry Box

(Note: for 13K speech a 3g2 file parser shall be able to read both the QCELPSampleEntry and the MP4AudioSampleEntry storage methods.)

For storage of 13K speech media, MP4AudioSampleEntry also can be used. 13K speech data shall be stored inside of a media track in the same way as described in [21]

When storing a 13K speech bitstream in a 3GPP2 file, the handler-type field within the HandlerAtom shall be set to 'soun' to indicate media of type AudioStream, and the SampleEntry Box type shall be 'mp4a' and the same Box described in Section 8.4.1 is used.

For inclusion of 13K speech media in MP4AudioSampleEntry, the stream type specific information is in the ESDBox structure. The 13K speech codec is to be signaled by new value from the 'User Public' area of 'objectTypeIndication' within the DecoderConfigDescriptor structure.

```
        objectTypeIndication = 0xE1  ;
```

The QCELPDecoderSpecificInfo in ABNF [20] format is specified as

QCELPDecoderSpecificInfo = QLCM fmt

The above ABNF rule indicates that QCELPDecoderSpecificInfo is the same as the header for 13K vocoder in ".qcp" file format as described in [21], but without RIFF, riff-size, or anything after fmt. In addition, if the size of packets is completely constant, i.e. fixed rate encoding, the following rules apply to the definition of fmt (see variable-rate referenced by codec-info).

num-rates	= 0
rate-map	= 0
major	= 1

8.4.6.4 Mapping of QCELPSampleEntry Box and 13K Support in MP4AudioSampleEntry Box

Variables in QCELPSampleEntry Box and DecoderSpecificInfo in MP4AudioSampleEntry Box for 13K is translated as described in the following table.

QCELPSampleEntry	MP4AudioSampleEntry
Vendor	The first 4 bytes of Name field
decoder_version	The fifth byte of Name field
framesPerSample (fPS) fPS = sPB / (sPS * 0.02)	N.A.
N.A.	**AvgBitsPerSec (aBPS)** Calculated based on duration field in Track Header Box and data size in Sample Size Box.
N.A.	**bytesPerBlock (bPB)** Calculated according to the equation: aBPS = bPB * 8(bits/Byte) / (0.02 * fPS)
N.A.	**samplePerBlock (sPB)** sPB = sPS * 0.02 * fPS
N.A.	**numOfRates and bytesPerPacket** When fixed rate encoding is used, all fields should be set 0x00. When variable rate encoding is used, example 1 in packet definition in [21].

Table 8-15: Mapping table

8.4.7 SMV

SMV speech data shall be stored inside of a media track in such a way that is described in Section 11 of [11]. The magic number is not included. The codec data frames are stored in a consecutive order with a single TOC entry field as a prefix per each of data frame, where the TOC field is extended to one octet by setting the four most significant bits of the octet to zero, as illustrated in the following figure.

```
|<-- Octet 1 -->|<-- Octet 2 -->|<-- …… -->|<-- Octet N -->|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
|0|0|0|0|FR Type|     One SMV speech data frame            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
```

Figure 8-6: SMV frame byte alignment

8.4.7.1 SMVSampleEntry Box

For SMV speech, the box type of the SMVSampleEntry Box shall be 'ssmv'.

The SMVSampleEntry Box is defined as follows:

```
SMVSampleEntry ::= BoxHeader
        Reserved_6
        Data-reference-index
        Reserved_8
```

Reserved_2
Reserved_2
Reserved_4
TimeScale
Reserved_2
SMVSpecificBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'ssmv'
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 Boxs.	
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 Box of this media	
Reserved_2	Const unsigned int(16)		0
SMVSpecificBox		Information specific to the decoder.	

Table 8-16: SMVSampleEntry fields

If one compares the AudioSampleEntry Box for the SMVSampleEntry Box the main difference is in the replacement of the ESDBox, which is specific to MPEG-4 systems, with a box suitable for SMV. The **SMVSpecificBox** field structure is described in section8.4.7.2.

8.4.7.2 SMVSpecificBox field for SMVSampleEntry Box

The SMVSpecificBox fields for SMV shall be as defined in **Table 8-17**. The SMVSpecificBox for the SMVSampleEntry Box shall always be included if the MP4 file contains SMV media.

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'dsmv'
DecSpecificInfo	SMVDecSpecStruc	Structure which holds SMV Specific information	

Table 8-17: The SMVSpecificBox fields for SMVSampleEntry

BoxHeader Size and Type: indicate the size and type of the SMV decoder-specific Box. The type shall be 'dsmv'.

DecSpecificInfo: the structure where the SMV stream specific information resides. The SMVDecSpecStruc is defined as follows:

Field	Type	Details	Value
Vendor	Unsigned int(32)		
decoder_version	Unsigned int(8)		
Frames_per_sample	Unsigned int(8)		

Table 8-18: SMV DECSpecStruc

The definitions of SMVDecSpecStruc 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 should be used in this field. Otherwise, a vendor may create a four character code which best expresses the vendor's name. Else, it is recommended that the manufacturer creates a four character code which best addresses the manufacturer's name. This field may 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 associated with the vendor field. It may be used advantageously by the vendors, which have optimal encoder-decoder version pairs. The value shall be set to 0 if the decoder version has no importance for the vendor. This field may be safely ignored.

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 should be carefully chosen since the 'access unit' is decided depending on the value defined by this field. For example, 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 aggregated 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.

8.4.8 VMR-WB

VMR-WB speech data are stored in the stream according to the VMR-WB storage file format (see Section 8.6 in [33]). The codec data frames are stored in a consecutive order with a single TOC entry field as a prefix per each of data frame, as illustrated in the following figure.

```
|<-- Octet 1 -->|<-- Octet 2 -->|<-- …… -->|<-- Octet N -->|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
|P|  FT   |Q|P|P|  One VMR-WB speech data frame           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+……+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.4-7: VMR-WB Frame byte alignment

The FT field (Frame Types) and the Q bit (Frame Quality Indicator) are defined in section 8.5.3 of [33]. The P bits are padding bits and shall be set to 0.

8.4.8.1 VMRSampleEntry Box

For VMR-WB speech the box type of the VMRSampleEntry Box shall be 'svmr'.

The VMRSampleEntry Box is defined as follows:

```
VMRSampleEntry ::= BoxHeader
        Reserved_6
```

Data-reference-index
Reserved_8
Reserved_2
Reserved_2
Reserved_4
TimeScale
Reserved_2
VMRSpecificBox:

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		"svmr’
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 Boxs.	
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 Box of this media	
Reserved_2	Const unsigned int(16)		0
VMRSpecificBox		Information specific to the decoder.	

Table 8-19: VMRSampleEntry fields

If one compares the MP4AudioSampleEntry Box to the VMRSampleEntry Box the main difference is in the replacement of the ESDBox, which is specific to MPEG-4 systems, with a box suitable for VMR-WB. The **VMRSpecificBox** field structure is described in section 8.4.8.2.

8.4.8.2 VMRSpecificBox field for VMRSampleEntry Box

The VMRSpecificBox fields for VMR-WB speech shall be as defined in Table 8-20. The VMRSpecificBox for the VMRSampleEntry Box shall always be included if the 3GPP2 file contains VMR-WB media.

Field	Type	Details	Value
BoxHeader. Size	Unsigned int(32)		
BoxHeader. Type	Unsigned int(32)		‘dvmr’
DecSpecificInfo	VMRDecSpecStruc	Structure which holds the VMR-WB Specific information	

Table 8-20: The VMRSpecificBox fields for VMRSampleEntry

BoxHeader Size and Type: indicate the size and type of the VMR decoder-specific Box. The type shall be ‘dvmr’.

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

The VMRDecSpecStruc is defined as follows:

Field	Type	Details	Value
Vendor	Unsigned int(32)		
decoder_version	Unsigned int(8)		
mode_set	Unsigned int(16)		
media_sampling_frequency	Unsigned int(8)		
Frames_per_sample	unsigned int(8)		

Table 8-21: VMRDecSpecStruc

The definitions of VMRDecSpecStruc 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 should be used in this field. Otherwise, a vendor may create a four character code which best expresses the vendor's name. This field may be 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 associated with the vendor field. It may be used advantageously by vendors, which have optimal encoder-decoder version pairs. The value shall be set to 0 if the decoder version has no importance for the vendor. This field may be ignored.

mode_set: the active codec operating modes. Each bit of the mode_set parameter corresponds to one operating mode. The mode_set bit structure is as follows: (B15xxxxxxB8B7xxxxxxB0) where B0 (Least Significant Bit) corresponds to Mode 0, and B6 corresponds to Mode 4 with maximum half-rate. The mapping of B bits to the VMR-WB operating modes is as follows:

B0	VMR-WB Mode 0
B1	VMR-WB Mode 1
B2	VMR-WB Mode 2
B3	VMR-WB Mode 3 (AMR-WB interoperable mode)
B4	VMR-WB Mode 4
B5	VMR-WB Mode 2 with maximum half-rate
B6	VMR-WB Mode 4 with maximum half-rate
B7-B15	Reserved (shall be set to zero)

Table 8-22: VMR mode_set bit field assignments

If mode_set = 0x0007, VMR-WB modes 0, 1, and 2 are present in the stream. These modes correspond to CDMA Rate-Set II. If mode_set=0x0010, VMR-WB mode 4 is present in the stream. This mode corresponds to CDMA Rate-Set I [34].

If mode_set = 0x0008, only the content generated by the AMR-WB interoperable mode is present in the stream. By default, VMR-WB is interoperable with 3GPP/AMR-WB (ITU-T/G.722.2) only at 12.65 kbps in mode 3.

Note that there is only one AMR-WB interoperable mode in VMR-WB. While in the AMR-WB interoperable mode, mode switching is not allowed. For the duration of an interoperable session/content generation, VMR-WB and AMR-WB shall operate in mode

3 and codec mode 2, respectively.

media_sampling_frequency: the sampling frequency of the input media. The media sampling frequency in VMR-WB by default is 16 kHz (wideband speech). However, VMR-WB can also operate with media (speech/audio) sampled at 8 kHz. If media_sampling_frequency = 0x00 then the media in the bit stream was originally sampled at 16 kHz (default). If the media_sampling_frequency = 0xFF then the media in the bit stream was originally sampled at 8 kHz (narrowband). Note that switching the media sampling frequency within a file is not allowed.

frames_per_sample: defines the number of frames to be considered as 'one sample' inside the 3GPP2 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.

8.5 Timed Text Format

If timed text is supported then 3GPP Timed Text as described in [35] and [36] shall be supported.

8.6 Asset Information

Asset information may be supported using the user-data-box as defined in [35] and [43].

In addition to the sub-boxes defined in [35] the “gadi” or Geographical Area Description [43] information box is defined as shown in Table 8-23. This provides a method for storing a GPS format geographical coordinate with uncertainty and a timestamp associated with a media element.

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'gadi'
BoxHeader.Version	Unsigned int(8)		0
BoxHeader.Flags	Bit(24)		0
week_number	Unsigned int(16)		0
seconds	Unsigned int(24)		
GADSpecInfo	GADstruct		

Table 8-23: The GAD Information box

week_number: (GPS timestamp) represents the current week number from midnight January 5, 1980 (morning of January 6, 1980). This field is encoded as a 16-bit unsigned integer in the range of 0-to-65535.

seconds: (GPS timestamp) represents the seconds in the week. This field is encoded as a 24-bit unsigned integer in the range of 0-to-604799 seconds.

GADSpecInfo: the structure where the GAD location and uncertainty resides. GADstruct is defined in section 7 of [43]. Recommended default is ‘Ellipsoidal Point with Altitude’ (section 7.3.5).

8.7 Encryption

A .3g2 file may support encrypted media using the method defined in section 7, Streaming-server extensions, and section 10, Encryption, of [35] including, but not limited to, the additional 3GPP2 media types in Table 8-24.

Section 10 describes encrypted SampleEntrys: EncryptedVideoSampleEntry, EncryptedAudioSampleEntry and EncryptedTextSampleEntry, as well as, Boxes for signaling the Key Manager Scheme. Section 7, identifies the support needed for SRTP including the attributes to be included in the SchemeTypeBox and SchemeInformationBox.

NOTE: This specification does not describe which schemes must be supported.

Format	Original format	Media content
‘encv’	‘avc1’	Encrypted video: AVC/H.264
'enca'	'sevc', 'ssmv', 'svmr', 'sqcp', …	encrypted audio: EVRC, EVRC-B, EVRC-WB, SMV, VMR-WB and 13K

Table 8-24 Additional formats for encrypted media tracks

8.8 Video-Buffer

Video-buffer parameters may be supported using the PSS Annex G and AVC HRD Sample groupings as defined in section 9, Video buffer information, of [35] with the following modifications:

- The data structures (e.g., AnnexGstruc and AVCHRDstruc) as defined in PSS Annex G [38] shall be used for the related functions described in MSS Annex C.

Note: PSS annex G is Equivalent to MSS Annex C as defined in [38].

Note: The AVC HRD video buffer only applies to the buffering requirements related to video encoding and decoding. Since interleaving requirements are determined by the server at the time of packetization additional buffer requirements are not communicated as part of the data structures (e.g., AnnexGstruc and AVCHRDstruc) as defined in PSS Annex G [38].

9 Presentation and Layout Support (SMIL)

This section describes the 3GPP2 SMIL profile.

9.1 Media Synchronization and Presentation Format

3GPP2 SMIL is a markup language based on SMIL Basic [18] and SMIL Scalability Framework.

3GPP2 SMIL consists of the modules required by SMIL Basic Profile (and SMIL 2.0 Host Language Conformance) and additional BasicAnimation, AudioLayout, MediaAccessibility, MediaDescription, MediaClipping, MediaParam, MetaInformation, PrefetchControl, MultiArcTiming, EventTiming, AccessKeyTiming and BasicTransitions modules. All of the following modules are included:

- SMIL 2.0 Animation Module – BasicAnimation
- SMIL 2.0 Content Control Modules – BasicContentControl, SkipContentControl and PrefetchControl
- SMIL 2.0 Layout Modules – BasicLayout, AudioLayout
- SMIL 2.0 Linking Modules – BasicLinking, LinkingAttributes
- SMIL 2.0 Media Object Modules – BasicMedia, MediaClipping, MediaParam, 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, EventTiming, AccessKeyTiming and MultiArcTiming
- SMIL 2.0 Transition Effects Module – BasicTransitions

9.1.1 Document Conformance

A conforming 3GPP2 SMIL document shall be a conforming SMIL 2.0 document.

All 3GPP2 SMIL documents use SMIL 2.0 namespace as the default namespace.

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

3GPP2 SMIL documents may declare requirements using '**systemRequired**' attribute:

EXAMPLE1:

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

Namespace URI http://www.3gpp2.org/SMIL20/FFMS10/ identifies the version of the 3GPP2 SMIL profile described in release 0 of this document [31]. Namespace URI http://www.3gpp2.org/SMIL20/FFMSA/ identifies the version of the 3GPP2 SMIL profile described in the present document. Authors may use this URI to indicate requirement for exact 3GPP2 SMIL semantics for a document or a subpart of a document:

EXAMPLE2:

```
        <smil xmlns="http://www.w3.org/2001/SMIL20/Language"
        xmlns:ffms10="http://www.3gpp2.org/SMIL20/FFMSA/"
            systemRequired="ffmsA">
```

The content authors should generally not include the FFMS requirement in the document unless the SMIL document relies on FFMS specific semantics that are not part of the W3C SMIL. The reason for this is that SMIL players that are not conforming 3GPP2 FFMS10 user agents may not recognize the FFMS10 URI and thus refuse to play the document.

9.1.2 User Agent Conformance

A conforming 3GPP2 SMIL user agent shall be a conforming SMIL Basic User Agent.

A conforming user agent shall implement the semantics of 3GPP2 SMIL as described in Sections 9.1.3 and 9.1.4.

A conforming user agent shall recognize:

- the URIs of all included SMIL 2.0 modules,
- the URI http://www.3gpp2.org/SMIL20/FFMS10/ as referring to all modules and semantics of the release 0 version of the 3GPP2 SMIL profile as described in [31].
- the URI http://www.3gpp2.org/SMIL20/FFMSA/ as referring to all modules and semantics of the release A version of the 3GPP2 SMIL profile described in the present document.

9.1.3 3GPP2 SMIL Language Profile definition

3GPP2 SMIL is based on SMIL 2.0 Basic language profile [18]. This section defines the content model and integration semantics of the included modules where they differ from those defined by SMIL Basic.

9.1.3.1 Animation Module

3GPP2 SMIL includes the BasicAnimation module of SMIL 2.0. BasicAnimation is not part of SMIL Basic and is an additional module in this profile. The SMIL 2.0 BasicAnimation module can incorporate animation onto a timeline, and can provide a mechanism for composing the effects of multiple animations. This module is optional.

User agents that implement the semantics of this module shall at least support **animate** element specified in SMIL 2.0. In this specification, animating a video object and animating over a video object is not supported.

9.1.3.2 Content Control Modules

3GPP2 SMIL 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.
Annex E extends the SMIL 2.0 BasicContentControl specification [18] by additional

definitions on the '**systemComponent**' test attribute.

NOTE: The SMIL specification [18] defines that all functionality of PrefetchControl module is optional. This means that although PrefetchControl is mandatory, user agents may implement some of none of the semantics of PrefetchControl module.

The PrefetchControl module adds the **prefetch**[1] 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.

9.1.3.3 Layout Module

3GPP2 SMIL shall use the BasicLayout module of SMIL 2.0 for spatial layout. The module is part of SMIL Basic. In addition, 3GPP2 SMIL should use the AudioLayout module for controlling aural media volumes via 'soundLevel' attribute on a region element. AudioLayout is not part of SMIL Basic and is an additional module in this profile. Default values of the width and height attributes for root-layout shall be the dimensions of the device display area.

9.1.3.4 Linking Module

3GPP2 SMIL shall use the SMIL 2.0 BasicLinking module for providing hyperlinks between documents and document fragments. The BasicLinking 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. For the same reason, a value "pause" of the '**destinationPlayState**' may be ignored. When the values of 'sourcePlaystate' and 'show' are ignored the player may also ignore the 'sourceLevel' attribute since it is of no use then.

9.1.3.5 Media Object Modules

3GPP2 SMIL includes the media elements from the SMIL 2.0 BasicMedia module and additional element and attributes from the MediaAccessibility, MediaDescription, MediaParam and MediaClipping modules. MediaAccessibility, MediaDescription, MediaParam and MediaClipping modules are additions in this profile to the SMIL Basic.

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.

MediaParam module allows the passing of additional parameters to the rendering of a media object. This specification extends the SMIL 2.0 specification [19] by defining some values for '**name**' and '**value**' attributes of MediaParam module and the expected

[1] Bold indicates elements that are not part of SMIL 2.0 Basic

behavior of 3GPP2 SMIL player when these are used.

A 3GPP2 SMIL player should render the content as specified whenever one of the following name value pairs are encoded as a parameter to a media object of one of the listed MIME types (note, the behavior of the 3GPP2 SMIL player is undefined for all other cases).

MIME type of the media object	value of the '**name**' attribute	value of the '**value**' attribute	Intended rendering of the media content.
application/text, application/xhtml+xml, application/vnd.wap.xhtml+xml, text/plain	color or foreground -color	Any legal value for the CSS2 color attribute [48] (e.g. “#ff0000”, “red”)	The text document is rendered with the given (default) color. Note: Attribute name="foreground-color" is included for compatibility.
application/text, application/xhtml+xml, application/vnd.wap.xhtml+xml, text/plain	font-size or textsize	Any legal value for the CSS2 font-size attribute [48] (e.g. “medium”, “12pt”)	The text document is rendered with the given (default) text size. The size values are interpreted as in CSS2 [48], Note: Attribute name="fontsize" is included for compatibility.
application/text, application/xhtml+xml, application/vnd.wap.xhtml+xml, text/plain	font-family	Allowed values are all generic font family names defined by CSS2 [48].	The text document is rendered with the font-family that is determined by the font matching algorithm of CSS2 [48].
image/jpeg, image/gif, image/png, text/plain	tile	true or false	The media element is tiled (repeated). All tiling covers the region. For the tiled media, no animation and no transition shall be used.
image/jpeg, image/gif, image/png, text/plain	opacity	Alpha value within the range 0.0 (fully transparent) to 1.0 (fully opaque). Default value is 1.0.	The media element is rendered where opaque colors are made transparent.

Table 9-1 3GPP2 SMIL MIME types and attributes

9.1.3.6 MetaInformation Module

The MetaInformation module of SMIL 2.0 is included in the profile. This module is an 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.

9.1.3.7 Structure Module

The Structure module defines the top-level structure of the document. It is included in SMIL Basic.

9.1.3.8 Timing and Synchronization modules

The timing modules included in the 3GPP2 SMIL are BasicInlineTiming, MinMaxTiming, BasicTimeContainers, RepeatTiming EventTiming, AccessKeyTiming and MultiArcTiming. The EventTiming, AccessKeyTiming and MultiArcTiming modules are additions in this profile to the SMIL Basic profile.

For 'begin' and 'end' attributes any number of offset-values, event-values, and accesskey-values should be allowed. If multiple of these values are used, they shall be separated by semicolon. 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 following event types:

- activateEvent;
- beginEvent;
- endEvent.

The following SMIL 2.0 Language event types should be supported:

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

Access key timing attributes that have invalid access keys of user agents shall be treated as being indefinite.

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.

9.1.3.9 Transition Effects Module

3GPP2 SMIL profile includes the SMIL 2.0 BasicTransitions module to provide a framework for describing transitions between media elements.

NOTE: The SMIL specification [18] defines that all functionality of BasicTransitions module is optional: "Transitions are hints to the presentation. Implementations shall be able to ignore transitions if they so desire and still play the media of the presentation".

This means 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 [18]:

- 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.

9.1.4 Content Model

Table 9-2shows the full content model and attributes of the 3GPP2 SMIL profile. The attribute collections used are defined by SMIL Basic [18], SMIL Host Language Conformance requirements, chapter 2.4. Changes to SMIL Basic are shown in **bold**.

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, soundLevel
ref, animation, audio, img, video, text, textstream	Area, param, animate	COMMON-ATTRS, CONTCTRL-ATTRS, TIMING-ATTRS, repeat, region, MEDIA-ATTRS, **clipBegin(clip-begin), clipEnd(clip-end), alt, longDesc, readIndex, abstract**, **author**, **copyright, transIn, transOut**
param	EMPTY	**name, value, skip-content**
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, dur, type, subtype, startProgress, endProgress, direction, fadeColor. skip-content**
animate	**EMPTY**	**COMMON-ATTRS, CONTCTRL-ATTRS, TIMING-ATTRS, attributeName, attributeType, targetElement, from, to, by, values, calcMode, accumulate, additive, skip-content**

Table 9-2: Content model for the 3GPP2 SMIL profile

10 File Format for 13K Speech “.QCP”

This section describes the qcp file format for reading and writing 13K vocoder packets. RFC2658 [11] specifies RTP streaming for 13K vocoder but does not include a file format. The qcp file format is described in [21]. The MIME type for “.qcp” files with 13K vocoder is “audio/qcelp”.

11 Compact Multimedia Format “.cmf”

This section specifies a binary file format container for multimedia elements with embedded time synchronization information. This syntax is called Compact Media Format (CMF) and can be employed to create a multimedia with 13K vocoder speech, WAVE/RIFF sound [28], IMA ADPCM sound [26], MIDI [23], text, JPEG [24] pictures, PNG pictures [25], BMP pictures [27] and animation data in messaging and other applications.

CMF media may be received, generated, or stored by Internet-connected devices such as cell phones, laptops, PDAs, desktops, servers, etc. for various applications.

Typical applications of CMF include:

- Multimedia ringers with graphics, text, MIDI and speech
- Audio postcard messages with speech and JPEG
- Advertisements with graphics, text and audio
- Karaoke with graphics
- Animated cartoons with MIDI, text and speech

CMF files consist of header information and track chunks. The header contains metadata such as title, author, and copyright as well as global parameters used to interpret the track chunks. Each track chunk describes a particular multimedia element and its timing information.

CMF uses the application/cmf media type and the .cmf file extension.

11.1 Description of CMF Content

A CMF file is composed of file identifier, file length, header information, and one or more content tracks.

The file identifier and length identify the CMF file and its length.

The header sets up necessary global parameters for interpreting the CMF tracks. It contains the number of tracks, content type, and detailed information about the tracks. Content type shows whether the media is text, melody, picture, animations, vibration, or LED. Content type also specifies the format of the media such as character set in case of text and so on.

In addition, the CMF header contains the metadata such as title, copyright, date, source, etc.

The CMF tracks contain events which specify the multimedia contents and how they should be temporally synchronized in relation to each other. The events also contain information on how the media should be played back. For instance, how a picture, text, or animation should be positioned on the display, and how a melody should be played. Non-MIDI media is limited to the first track.

11.2 Formal Syntax of CMF Content

This section describes CMF using ABNF format [20].

```
CMF-file                 = cmid length4 CMF-header *media-chunk 1*4CMF-
                         track
length4                  = 4OCTET
OCTET                    = %x00-FF
cmid                     = %x63 %x6d %x69 %x64
```

```
CMF-header              = length2 content-type nTracks *sub-chunk
length2                 = 2OCTET
content-type            = (melody (complete / part)) / (song
                        instruments)
melody                  = %x01
                        ; used for ringers
complete                = %x01
                        ; all of the melody
part                    = %x02
                        ; part of the melody
song                    = %x02
                        ; used for pictures plus audio
instruments             = OCTET
                        ; bit field
                        ; The octet contains bits set with meanings as
                        follows
                        ; %x01: contains musical event
                        ; %x02: contains wave data
                        ; %x04: contains text data
                        ; %x08: contains picture data
                        ; %x10: contains female vocal parts
                        ; %x20: contains male vocal parts
                        ; %x40: contains other vocal parts
                        ; %x80: Always zero

nTracks                 = %x01-04
                        ; Number of track chunks is limited to 4.
                        ; This provides up to 16 active instruments.

sub-chunk               = 1*required-chunk *optional-chunk
                        ; one or more required sub-chunks
                        ; Only one of each type is allowed.
                        ; If identical sub-chunks are present,
                        ; only the last of the sub-chunks shall be used.
                        ; The player shall not fail when receiving
                        ; unsupported sub-chunks. The unsupported
                        ; sub-chunks shall be ignored.

required-chunk          = vers-chunk
                        / note-chunk
                        / cnts-chunk

optional-chunk          = code-chunk
                        / titl-chunk
                        / date-chunk
                        / sorc-chunk
                        / copy-chunk
                        / exsn-chunk
                        / exsa-chunk
                        / exsb-chunk
                        / exsc-chunk
                        / cuep-chunk
                        / pcpi-chunk
                        / cnts-chunk
                        / prot-chunk
                        / poly-chunk
                        / wave-chunk

vers-chunk              = "vers" %x0004 "0500"
```

```
                        ; A version number of "0500" refers to C.S0050-
                        0.
                        ; A version number of "0530" refers to C.S0050-
                        A.
code-chunk              = "code" %x0001 code-value

titl-chunk              = "titl" length2 title

title                   = *OCTET
                        ; number of octets specified in length2
                        ; field of titl-chunk

date-chunk              = "date" length2 date
date                    = *OCTET
                        ; number of octets specified in length2
                        ; field of date-chunk

copy-chunk              = "copy" length2 copyright-notice
                        ; Content provider’s copyright notice.

copyright-notice        = *OCTET
                        ; number of octets specified in length2
                        ; field of copyright-notice

sorc-chunk              = "sorc" %x0001 source-info

note-chunk              = "note" %x0002 note-msg-config

note-msg-config         = [%x0000 / %x0001]
                        ; %x0000 : Note message is of length 3 octet
                        ; %x0001 : Note message is of length 4 octet
                        ; In the second case, the extra (fourth) octet
                        ; is used to include velocity and octave shift
                        ; information.

exsn-chunk              = "exsn" %x0002 2data
                        ; exsn-chunk specifies the length of normal
                        extension
                        ; status-A message

exsa-chunk              = "exsa" %x0002 2data
                        ; exsa-chunk specifies the length of extension
                        ; status-A, class A message

exsb-chunk              = "exsb" %x0002 2data
                        ; exsb-chunk specifies the length of extension
                        ; status-A, class B message

exsc-chunk              = "exsc" %x0002 2data
                        ; exsc-chunk specifies the length of extension
                        ; status-A, class C message

cuep-chunk              = "cuep" 4nTracks *OCTET
                        ; cuep-chunk specifies the location of the cue
                        ; point start point, which is the starting
                        ; position of the main theme music in the track.
                        ; The length of cuep-chunk shall be equal to
                        number
                        ; of tracks multiplied by 4 bytes. Every 4 bytes
                        ; consists of the location of the cue point
                        start
                        ; point in the corresponding track chunk.
                        ; Each cue point start point is defined to be a
                        ; byte offset to the beginning of the theme
                        music
                        ; event in that track chunk. If the value of
                        cuep-
                        ; chunk is FFFFFFFF, the cuep-chunk is
                        considered
                        ; to be invalid and shall not be used.

pcpi-chunk              = "pcpi" %x0001 axis-offset
axis-offset             = %x00-01
                        ; pcpi-chunk describes the picture packet
                        ; information.
                        ; %0x00 : XY offsets in pcpi are in percent
                        ; %0x01 : XY offsets in pcpi are in pixels

cnts-chunk              = "cnts" length2 multi-media-type
multi-media-type        = media-type *(";" media-type)
                        ; cnts-chunk describes the various media
                        contents
                        ; that are present in the file.
                        ; Multiple media are separated by ";" in cnts-
                        chunk.
                        ; Examples: SONG; WAVE; TEXT; PICT
                        ; length2 specifies the length of the data in
                        ; cnts-chunk

prot-chunk              = "prot" length2 *OCTET
                        ; length2 specifies the length of the data in
                        ; prot-chunk

poly-chunk              = "poly" %x0001 data

wave-chunk              = "wave" %x0001 data

code-value              = %b00000000-10000110
                        ; %b00000000 :          ANSI CHARSET
                        ; %b00000001 :          ISO8859-1
                        ; %b00000010 :          ISO8859-2
                        ; %b00000011 :          ISO8859-3
                        ; %b00000100 :          ISO8859-4
                        ; %b00000101 :          ISO8859-5
                        ; %b00000110 :          ISO8859-6
                        ; %b00000111 :          ISO8859-7
                        ; %b00001000 :          ISO8859-8
                        ; %b00001001 :          ISO8859-9
                        ; %b00001010 :          ISO8859-10
                        ; %b10000000 :          Shift-JIS
                        ; %b10000001 :          HANGUL CHARSET
                        ; %b10000010 :          Chinese Simplified
                        ; %b10000011 :          Chinese Traditional
                        ; %b10000100 :          Hindi
                        ; %b10000101 :          Thai
                        ; %b10000110 :          UTF-16

source-info             = no-copyright/copyright-DL/copyright-
                        MO/copyright-DT
```

```
no-copyright            = %b00
                        ; No copyright, downloaded (from the net)
copyright-DL            = %b01
                        ; Copyrighted,  downloaded (from the net)
copyright-MO            = %b11
                        ; Copyrighted,  mobile originated
copyright-DT            = %b101
                        ; Copyrighted, from desktop

data                    = OCTET
media-type              = "SONG"          ; Contains MIDI
                        /"WAV"            ; Contains Wave sounds
                        /"TEXT"           ; Contains text data
                        /"PICT"           ; Contains still image data
                        /"ANIM"           ; Contains animation data
                        /"LED"            ; Contains LED data
                        /"VIB"            ; Contains VIB data
media-chunk             = dls-chunk / anim-chunk / image-chunk
dls-chunk               = "DLS" length4 *OCTET
                        ; A single DLS file is placed as the chunk data.
                        ; The DLS file shall conform to the Mobile DLS
                        ; specification [46].
anim-chunk              = "ANIM" length4 anim-attrib0 *OCTET
                        ; A single animation file is placed as the chunk
                        data.
anim-attrib0            = anim-chunk-id anim-p-format
anim-chunk-id           = %b00000-11111
image-chunk             = "IMAG" length4 imag-attrib0 *OCTET
                        ; A single image file is placed as the chunk
                        data.
imag-attrib0            = imag-chunk-id imag-format
imag-chunk-id           = reserved id
imag-format             = reserved pic-format

CMF-track               = "trac" length4 *event

event                   = delta-time event-message
delta-time              = OCTET
                        ; Delta time is described the elapsed time from
                        ; a previous event. The unit of time is
                        determined
                        ; from timebase-tempo, defined later in this
                        syntax.
                        ; Default tempo Value is 125.
                        ; Default timeBase Value is 48. See section
                        11.3.1.

event-message           =  note-message
                        / ext-A-message
                        / ext-B-message
                        / ext-info-message

note-message            = note-status gate-time
                        / note-status gate-time vel-oct-shift
                        ; If note-msg-config (defined in this syntax) is
                        1,
                        ; we have velocity-octaveShift info.

note-status             = channel-index key-number
```

```
                        ; One octet containing channel index and key
                        number

channel-index           = %b00-11
                        ; Assigned channel index, defined
                        ; with respect to the channel reference index.

key-number              =  %b000000-111110
                        ; key-number is 0 to 62.
                        ; key-number 63 is prohibited.
                        ; key-number 15 is middle C of keyboard

gate-time               = OCTET
                        ; Continuation time from note-on to note-off.
                        ; If a gate-time value of more than 255 is
                        required
                        ; multiple note-messages are used.

vel-oct-shift           = velocity octave-shift
                        ; octet containing velocity (6 bits)
                        ; and octave-shift (2 bits)

velocity                = %b000000-111111
                        ; velocity is 0 to 63.

octave-shift            = %b00-11
                        ; %b00 : No change
                        ; %b01 : Increase one octave
                        ; %b10 : decrease two octaves
                        ; %b11 : decrease one octave

ext-A-message           = %xFF A-command-data

ext-B-message           = %xFF B-command-data

ext-info-message        = %xFF ((%b11110001 wav-data-length wav-data)
                        /        (%b11110010 text-data-length text-data)
                        /        (%b11110011 pict-data-length picture-
                        data)
                        /        (%b11110100 anim-data-length animation-
                        data)
                        /        (%x11110101 mip-data-length MIP-
                        Message)
                        /        (%b11110110 dls-bank-change-length dls-
                        bank-change))

wav-data-length         = length2;
text-data-length        = length2;
pict-data-length        = length2;
anim-data-length        = length2;
mip-data-length         = length2;
dls-bank-change-length  = length2;

A-command-data          = assigned-channel fine-pitch-bend
                        ; two octets containing assigned-channel
                        ; and fine-pitch-bend.

assigned-channel        = %b000-011
                        ; Assigned channel index (0..3)
```

```
fine-pitch-bend         = %b0000000000000-1111111111111
                        ; range: %x0000 to %x1fff (see table in sec.
                        11.3.3)
                        ; Fine pitchbend message sets the change value
                        of
                        ; the pitch specified in the note message.

B-command-data          = master-volume
                        / master-balance
                        / master-tune
                        / part-configuration
                        / pause
                        / stop
                        / reset
                        / timebase-tempo
                        / cuepoint
                        / jump
                        / NOP
                        / end-of-track
                        / program-change
                        / bank-change
                        / volume
                        / panpot
                        / pitchbend
                        / channel-assign
                        / pitchbend-range
                        / wave-channel-volume
                        / wave-channel-panpot
                        / text-control
                        / picture-control
                        / vib-control
                        / LED-control

master-volume           = %b10110000 %x00-7F
                        ; specifies the volume adjustment
                        ; for all audio events.  The
                        ; default value is 100 (0 dB).
                        ; Range is from 0 to 127.

master-balance          =     %b10110001 %x00-7F
                        ; specifies the range of master
                        ; balance adjustment where
                        ; %x00 defines Pan Left, %x40
                        ; defines Center, and
                        ; %x7F defines Pan Right

master-tune             = %b10110011 %x34-4C
                        ; Master Tune for music synthesizer
                        ; %x34 : -(12 x 100 ) [cents]
                        ; ...
                        ; %x3E : -( 2 x 100 ) [cents]
                        ; %x3F : -( 1 x 100 ) [cents]
                        ; %x40 : 0 [cents]
                        ; %x41 : ( 1 x 100 ) [cents]
                        ; %x42 : ( 2 x 100 ) [cents]
                        ; ...
                        ; %x4C : (12 x 100 ) [cents]
                        ; A cent is a change in frequency by 2^(1/1200).
                        ; So frequencies f1 and f2 are one cent apart if
                        ; f2 = f1 x 2^(1/1200), and three cents apart if
```

```
                         ; f2 = f1 x 2^(3/1200)

part-configuration       = %b10111001 %x00
                         ; reserved

pause                    = %b10111101 %x00
                         ; Pause player

stop                     = %b10111110 %x00
                         ; Stop player

reset                    = %b10111111 %x00
                         ; Reset controllers

timebase-tempo           = timebase tempo

timebase                 = %b1100000-11001111
                         ; timebase - %b11000000 is index into the table
                         in
                         ; section 11.3.1.

tempo                    = %x14-FF
                         ; number of quarter notes in one minute

cuepoint                 = %b11010000 cuep-start-end
cuep-start-end           = cuep-startpoint / cuep-endpoint
cuep-startpoint          = %x00
                         ; cuepoint start point
cuep-endpoint            = %x01
                         ; cuepoint end point

jump                     = %b11010001 jump-data

jump-data                = destination jump-id no-of-jumps
                         ; one octet with following three fields

destination              = dest / jump
dest                     = %b00
                         ; destination point
jump                     = %b01
                         ; jump point

jump-id                  = %b00-11
                         ; jump ID (0 to 3)

no-of-jumps              = %b0000-1111
                         ; (15 is infinity)

NOP                      = %b11011110 NOP-data

NOP-data                 = OCTET
                         ; NOP-data contains value N in
                         ; equation 256 * N + (delta time).

end-of-track             = %b11011111 %x00
                         ; end of track

program-change           = %b11100000 prog-data

prog-data                = channel-index prog-change
```

```
                        ; one octet containing 2 fields

channel-index           = %b00-11

prog-change             = %b000000-111111
                        ; program change value

bank-change             = %b11100001 bank-change-attr

bank-change-attr        = channel-index bank-change
                        ; one octet containing 2 fields

channel-index           = %b00-11

bank-change             = %b000000-111111
                        ; bank change value

volume                  = %b11100010 volume-attr

volume-attr             = channel-index volume-change
                        ;one octet containing 2 fields

channel-index           = %b00-11

volume-change           = %b000000-111111
                        ; volume change value

panpot                  = %b11100011 panpot-attr

panpot-attr             = channel-index panpot-change
                        ; one octet containing two fields

channel-index           = %b00-11

panpot-change           = %b000000-111111
                        ; panpot change value
                        ; %b000000 : Far Left
                        ; %b100000 : Center
                        ; %b111111 : Far Right

pitchbend               = %b11100100 pitchbend-attr
pitchbend-attr          = channel-index pitchbend-change
                        ; one octet containing two fields
channel-index           = %b00-11
pitchbend-change        = %b000000-111111
                        ; pitchbend change value (see table in
                        ; section 11.3.2)

channel-assign          = %b11100101 channel-data

channel-data            = channel-index channel-value
                        ; one octet containing two fields

channel-index           = %b00-11

channel-value           = %b000000-001111

pitchbend-range         = %b11100111 pitchrange-data

pitchrange-data         = channel-index pitch-range
```

```
                        ; one octet containing two fields

channel-index           = %b00-11

pitch-range             = %b000000-001100
                        ; pitch bend range

wave-channel-volume     = %b11101000 wave-vol

wave-vol                = channel-index volume-change
                        ; one octet containing two fields

channel-index           = %b00-11

volume-change           = %b000000-111111
                        ; volume change value

wave-channel-panpot     = %b11101001 wave-panpot

wave-panpot             = channel-index panpot-change
                        ; one octet containing two fields

channel-index           = %b00-11

panpot-change           = %b000000-111111
                        ; wave panpot change value

text-control            = %b11101011 tex-cont

tex-cont                = %x00-05
                        ; %x00 : Text Enable
                        ; %x01 : Text Disable
                        ; %x02 : Clear text
                        ; %x03 : reserved
                        ; %x04 : Increase cursor position by 1 byte
                        ; %x05 : Increase cursor position by 2 bytes

picture-control         = %b11101100 pict-cont
pict-cont               = pict-enable / pict-disable / clear-pict
pict-enable             = %x00
                        ; Picture Enable
pict-disable            = %x01
                        ; Picture Disable
clear-pict              = %x02
                        ; Clear picture

vib-control             = %b11101110 vib-data

vib-data                = %b0 off-on vib-pattern
                        ; one octet containing one zero bit and two
                        fields
off-on                  = %b0-1
                        ; enable is %b1 and disable is %b0
vib-pattern             = %b000000-111111
                        ; vibrator pattern

LED-control             = %b11101101 led-data

led-data                = %b0 off-on color-pattern
```

```
                        ; one octet containing one zero bit and two
                        fields

off-on                  = %b0-1
                        ; enable is %b1 and disable is %b0

color-pattern           = %b000000-111111
                        ; color pattern

wav-data                = wav-data-normal / wav-data-ADPCM
wav-data-normal         = wav-atrb1 wav-atrb2 packet-offset prev-flag *
                        OCTET
wav-data-ADPCM          = wav-atrb1 wav-p-mode wav-ima packet-offset
                        prev-flag wav-data-adpcm-info *OCTET
wav-atrb1               = channel-index channel-id
                        ; one octet containing two fields

wav-data-adpcm-info     = adpcm-samplingrate %b00 adpcm-blocksize
adpcm-samplingrate      = %b00-%b11
                        ; %b00 -> 8 KHz
                        ; %b01 -> 16 KHz
                        ; %b10 -> 32 KHz
                        ; %b11 -> Reserved
adpcm-blocksize         = %b000000000000-%b111111111111
                        ; Typically, 256 bytes for 8 and 16 KHz or
                        ; 512 bytes for 32 KHz.
channel-index           = %b00-11
channel-id              = %b000000-111111
wav-atrb2               = wav-p-mode wav-format
                        ; one octet containing two fields

wav-p-mode              = wav-store / wav-set / wav-recycle
wav-store               = %b00
                        ; Store mode, see section 11.5.9.1
wav-set                 = %b01
                        ; Set mode, see section 11.5.9.2
wav-recycle             =  %b10
                        ; Recycle mode, see section 11.5.9.3

wav-format              = wav-riff / wav-mp3 / wav-qcp / wav-aac / wav-
                        vmr / wav-evrc /wav-evrcb /wav-evrcwb
wav-riff                = %b000000
                        ; WAV/RIFF
wav-mp3                 = %b000011
                        ; MP3
wav-qcp                 = %b000100
                        ; QCP 13k
wav-ima                 = %b000101
                        ;IMA ADPCM
wav-aac                 = %b000110
                        ; AAC ATDS or HE AAC ADTS

wav-vmr                 = %b000111
                        ; VMR-WB
wav-evrc                = %b001000
                        ; EVRC
wav-evrcb               = %b001001
                        ; EVRC-B
wav-evrcwb              = %b001010
                        ; EVRC-WB
```

```
packet-offset           = length4
                        ; specifies offset in bytes to next Wave packet

prev-flag               = prev-flag-en / prev-flag-dis
prev-flag-en            = %x01
prev-flag-dis           = %x00
                        ; specifies if current wave packet is continued
                        ; from previous (for those formats with frame
                        ; history).
                        ; Implementations should ignore the seven most
                        ; significant bits

MIP-Message             = mip-entry-count mip-entry

mip-entry-count         = %x01-%x10
                        ; this field describes the number of MIP entries
                        ; contained in the MIP-Message
                        ; between 1 and 16 channels may have MIP entries

mip-entry               = mip-channel mip-value

mip-channel             = OCTET
                        ; mip-channel = 0000cccc
                        ; cccc = MIDI channel number

mip-value               = OCTET
                        ; MIP value for the corresponding
                        ; channel index (range 1-127)

dls-bank-change         = dls-midi-channel dls-msb-bank dls-lsb-bank
dls-midi-channel        = %x00-%x0F
                        ; The MIDI channel that the dls-bank-change is
                        ; providing additional information to uniquely
                        ; associate a DLS instrument with.

dls-msb-bank            = %b00000000-%b01111111
                        ; The MIDI MSB bank to be associated with the
                        ; dls-midi-channel
dls-lsb-bank            = %b00000000-%b01111111
                        ; The MIDI LSB bank to be associated with the
                        ; dls-midi-channel

text-data               =  text-atrb * OCTET

text-atrb               =  %b0 set-append x-align y-align
                        ; Set/Append and XY Alignment
                        ; one octet containing a zero bit followed
                        ; by three fields

set-append              = set-string / append-string
set-string              = %b0
                        ; Set a string
append-string           = %b1
                        ; Append a string

x-align                 = txt-x-left / txt-x-center / txt-x-right
```

```
txt-x-left              = %b000
                        ; Left x-alignment
txt-x-center            = %b001
                        ; Center x-alignment
txt-x-right             = %b010
                        ; Right x- alignment

y-align                 = txt-y-bottom / txt-y-center / txt-y-top
txt-y-bottom            = %b000
                        ; Bottom  y-alignment
txt-y-center            = %b001
                        ; Center y-alignment
txt-y-top               = %b010
                        ; Top y-alignment

picture-data            =  pict-atrb1 pict-atrb2 pict-atrb3 pict-x-off
                        pict-y-off * OCTET

pict-atrb1              =  reserved id

reserved                =  %b00-11
                        ; should set to %b00 on creation
                        ; should ignore value when reading

id                      = %b000000-111111
                        ; Picture packet ID (0-63)

pict-atrb2              = pic-p-mode pic-format

pic-p-mode              = pict-store / pict-set / pict-recycle
pict-store              = %b00
                        ; Store mode, see section 11.5.9.1
pict-set                = %b01
                        ; Set mode, see section 11.5.9.2
pict-recycle            = %b10
                        ; Recycle mode, see section 11.5.9.3

pic-format              = BMP-format / JPEG-format / PNG-format
BMP-format              = %b000001
JPEG-format             = %b000010
PNG-format              = %b000011

pict-atrb3              = %x00
                        ; Draw Mode : Normal
pict-x-off              = OCTET
                        ; If subchunk for Picture packet = 0
                        ; %b00000000 : X-offset 0%
                        ; %b00000001 : X-offset 1%
                        ; …
                        ; %b01100100 : X-offset 100%
                        ; %b01100101 : Left
                        ; %b01100110 : Center
                        ; %b01100111 : Right
                        ; If subchunk for Picture packet = 1
                        ; pict-x-off = pixel offset from left (0..255)

pict-y-off              = OCTET
                        ; If subchunk for Picture packet = 0
                        ; %b00000000 : Y-offset 0%
                        ; %b00000001 : Y-offset 1%
                        ; ...
                        ; %b01100100 : Y-offset 100%
                        ; %b01100101 : Top
                        ; %b01100110 : Center
                        ; %b01100111 : Bottom
                        ; If subchunk for Picture packet = 1
                        ; pict-y-off = pixel offset from top (0..255)


animation-data          = anim-atrb0 anim-atrb1 anim-cmd-spcfc
anim-atrb0              =  length4
                        ; four bytes to indicate the length of the
                        ; animation if length2 in ext-info-message
                        ; is set to zero. Otherwise they are specified
                        ; as a continuation flag indicating that
                        ; the next animation packet is continued from
                        the
                        ; current one.

anim-atrb1              = anim-p-mode anim-id
                        ; one octet containing two fields
                        ; note both fields contain fixed (reserved)
                        ; values

anim-p-mode             = %b01
                        ; Animation packet mode
                        ; %b01 : reserved

anim-id                 = %b000000
                        ; Animation packet ID
                        ; %b000000 : reserved

anim-cmd-spcfc          = anim-imag-obj-data / anim-frame-id /
                        anim-frame-cmd / anim-chunk-frame-cmd

anim-imag-obj-data      = anim-p-format imag-obj-data
                        anim-x-off anim-y-off *OCTET

anim-frame-id           = anim-p-format frame-id
                        anim-x-off anim-y-off request-frame-id

anim-frame-cmd          = anim-p-format frame-cmd
                        anim-x-off anim-y-off *OCTET

anim-chunk-frame-cmd    = anim-p-format chunk-frame-cmd
                        anim-x-off anim-y-off request-chunk-frame-cmd

request-chunk-frame-cmd = %b000 anim-chunk-id request-frame-id2

request-frame-id        = %x0000-%xFFFF
request-frame-id2       = %x00000000-%xFFFFFFFF
                        ; ID of the frame to be requested from the
                        ; animation decoder

anim-p-format           = anim-fmt-SVG

anim-fmt-SVG            = %b011
                        ;  SVG Tiny version 1.2 [45].

imag-obj-data           = %b00000
```

```
                         ; Image object data

frame-id                 = %b00001
                         ; Frame ID

frame-cmd                = %b00010
                         ; Frame command

chunk-frame-cmd          = %b10000
                         ; Frame command processed according to the
                         ; animation referenced by anim-chunk-id.

anim-x-off               = OCTET
                         ; If subchunk for Animation packet = 0
                         ; %b00000000 : X-offset 0%
                         ; %b00000001 : X-offset 1%
                         ; ...
                         ; %b01100100 : X-offset 100%
                         ; %b01100101 : Left
                         ; %b01100110 : Center
                         ; %b01100111 : Right
                         ; If subchunk for Animation packet = 1
                         ; anim-x-off = pixel offset from left (0..255)

anim-y-off               = OCTET
                         ; If subchunk for Animation packet = 0
                         ; %b00000000 : Y-offset 0%
                         ; %b00000001 : Y-offset 1%
                         ; ...
                         ; %b01100100 : Y-offset 100%
                         ; %b01100101 : Top
                         ; %b01100110 : Center
                         ; %b01100111 : Bottom
                         ; If subchunk for Animation packet = 1
                         ; anim-y-off = pixel offset from top (0..255)
```

11.3 Tables

11.3.1 TimeBase

TimeBase is expressed by the lower 4-bits of the status byte. The default value is 48.

%b----0000	TimeBase = 6
%b----0001	TimeBase = 12
%b----0010	TimeBase = 24
%b----0011	TimeBase = 48
%b----0100	TimeBase = 96
%b----0101	TimeBase = 192
%b----0110	TimeBase = 384
%b----0111	Reserved
%b----1000	TimeBase = 15

%b----1001	TimeBase = 30
%b----1010	TimeBase = 60
%b----1011	TimeBase = 120
%b----1100	TimeBase = 240
%b----1101	TimeBase = 480
%b----1110	TimeBase = 960
%b----1111	Reserved

Table 11-1: TimeBase Values

11.3.2 Pitch Bend

The following table contains a description of the pitch bend value when the pitch bend range is assigned RangValue. The default value for RangValue is 2.

%b000000	-(32 x RangeValue x 100 / 32) [cents]
…	…
%b011110	-(2 x RangeValue x 100 / 32) [cents]
%b011111	-(1 x RangeValue x 100 / 32) [cents]
%b100000	0 [cent]
%b100001	(1 x RangeValue x 100 / 32) [cents]
%b100010	(2 x RangeValue x 100 / 32) [cents]
…	…
%b111111	(31 x RangeValue x 100 / 32) [cents]

Table 11-2: Pitch Bend Range values

11.3.3 Fine Pitch Bend

The following table contains a description of the fine pitch bend value.

%b0000000000000	-(4096 x RangeValue x 100 / 4096) [cents]
…	…
%b0111111111110	-(2 x RangeValue x 100 / 4096) [cents]
%b0111111111111	-(1 x RangeValue x 100 / 4096) [cents]
%b1000000000000	0 [cent]
%b1000000000001	(1 x RangeValue x 100 / 4096) [cents]
%b1000000000010	(2 x RangeValue x 100 / 4096) [cents]
…	…
%b1111111111111	(4095 x RangeValue x 100 / 4096) [cents]

Table 11-3: Fine PITCH bend range values

11.4 Acceptable Profiles for CMF file format

A CMF profile identifies a set of media combinations.

Compliant players shall check the acceptable profiles in the `cnts-chunks`. Only the following profiles are specified. Other profiles are invalid configurations of the CMF file format. A list of these profiles is documented here.

Table 11-4 maps the media types used by the profiles to the allowed media formats. Only allowed media formats may be used. In all profiles, only one media format is allowed per media type per CMF file.

Media Types	Allowed Formats
WAV	IMA ADPCM, 13K QCELP, VMR-WB, AAC, HE AAC
PICT	JPEG, PNG
ANIM	SVG Tiny
SONG	General MIDI 2, General MIDI 2 and SP-MIDI

Table 11-4: Allowed formats for each media type

11.4.1 Talking Picture Messaging

cnts = WAV;PICT

This profile is primarily used for messaging applications.

11.4.2 Audio-only Profile

cnts = SONG;WAV

This profile is primarily used for ringers and other audio only applications such as the audio portion of a game application.

11.4.3 Picture Ringers

cnts = SONG;WAV;PICT

This profile is an enhancement on 11.4.2 that adds graphics capability for picture or audio postcards.

11.4.4 Animated Ringers

cnts = SONG;WAV;ANIM, PICT

This profile is used for animations with audio, such as an animated cartoon.

11.5 CMF Conformance Guidelines

In order to interoperate with existing deployments, the guidelines in this section shall be followed.

11.5.1 AAC Requirements

The distribution of HE AAC within the wav-aac track uses implicit signaling. This signaling assumes that HE AAC decoders will parse the wav-aac data stream to discover whether or not the data stream contains SBR data. If it contains SBR data, HE AAC decoders will set the output sample rate to double the indicated AAC LC sample rate and the SBR data will be decoded accordingly. The signaling also assumes that if an HE AAC data stream is presented to an AAC LC decoder, the AAC LC decoder shall decode the AAC LC portion of the wav-aac data stream.

Data carried in the wav-aac track shall be AAC Level 2 or HE AAC Level 2 [40], [41], [42] and shall use the ADTS format.

11.5.2 Subchunk Requirements

There are 3 required subchunks: **note**, **vers**, and **cnts**. All encoders are REQUIRED to include these subchunks and all decoders are REQUIRED to verify the existence of these subchunks before playing the content.

11.5.3 MIDI Requirements

All MIDI related parameters should be interpreted according to General MIDI Level 2 requirements; see [29] and [30]. In those instances where parameters have different precision than the equivalent General MIDI Level 2 parameters, those parameters should be mapped to equivalent dynamic range.

11.5.4 MIP Requirements

A MIP message shall occur only in the first track and contain MIP values for all MIDI channels playing notes. Two pieces of information are present in the MIP message. The first is the priority of the MIDI channels. The order of the entries in the MIP message defines the priorities of the channels with the first entry having the highest priority. The second piece of information is the MIP value assignments, one value for each channel. Any channels not included in the message shall be muted by the player as described in section 2.2 of Scalable Polyphony MIDI Specification [44]. Also, MIP value assignments are cumulative as described in section 2.2 of Scalable Polyphony MIDI Specification [44].

11.5.5 Wave Packet Requirements

CMF encoders are recommended to break wave packets into subchunks with reasonable duration. These subchunks represent events and are time stamped by delta-time which is the elapsed time from one event to the previous one. The recommendation for subchunking allows CMF players to implement effective fast-forward and rewind operations without affecting the ability to properly handle wave packets.

A typical implementation breaks wave packets into 0.5 second. Using 0.5 second chunks allows a typical CMF player to achieve 0.5 second resolution in forward and rewind increments. The subchunks also contain a prev-flag parameter so that a CMF player is able to correctly implement a continuous bit-stream interface to the wave decoder when wave packets are provided with prev-flag set to %x01. When prev-flag is %x00, the CMF player should reset the wave decoder. The information in prev-flag is to ensure continuous decoding of audio packets.

11.5.6 "dls-bank-change" event

The intent of the dls-bank-change event is to supplement the limited addressing of the existing bank-change event, addressing the need for CMF to uniquely identify DLS instruments in the numerous ways a DLS editor can number the MSB (Most Significant Byte), LSB (Least Significant Byte), and Programs of DLS instruments. The dls-bank-change event is supplemental information to the existing bank-change event and should occur after the bank-change event changes a MIDI channel to a DLS bank. If the specified dls-midi-channel is not currently a DLS bank, this event will have no effect. The dls-bank-change event shall occur only in the first track.

11.5.7 ADPCM Requirements

A WAV file using the RIFF format is typically how 4-bit mono IMA-ADPCM is contained. When embedding IMA-ADPCM into a CMF file, only the data of the WAV file’s data RIFF chunk is used. Since this ADPCM data does not contain sampling rate or block size information, the first two bytes of the wav-data’s data are reserved for this information, where the sampling rate is just an index into the sampling rate table and the block-size is the size of each ADPCM frame (or block of data).

When the prev-flag is enabled, the sampling rate index and the block size shall not change from the previous wav-data.

11.5.8 Cue and Jump Points

11.5.8.1 Cue Points

Cue points are used to provide an alternative play mode for CMF files. When in cue-point play mode, the decoder should jump to the cue start point when starting playback. All rules for setup that are observed for normal playback at the beginning of the file should be observed. For example, an encoder is required to insert all configuration events in between cuepoint boundaries even if those events are redundant with configuration events outside cue-point boundaries.

11.5.8.2 Jump Points

Jump points are used to reuse portions of the playback using loops to reduce file size. The decoder is REQUIRED to parse a jump destination point and save a pointer to the file. Up to 4 JUMP IDs can be saved for later reference. When a jump command is received for a given destination ID, the decoder should continue playback from the destination point. The loop number specifies the number of times the jump should be taken. After the final jump, decoding should continue as normal ignoring the final jump command.

11.5.9 Recycle Requirements

Recycling is supported in Picture, Wave, and Animation packets. The use of recycle is recommended to optimize file sizes for data transmission. Each packet group allows for up to 64 individual IDs to be used for recycle.

11.5.9.1 Store Command

The “store” operation specifies that the decoder should not display the data, but instead cache the data for displaying in the future.

11.5.9.2 Set Command

The “set” operation specifies that the decoder should both cache the data and displaying it.

11.5.9.3 Recycle Command

The “recycle” operation specifies that the decoder should redisplay picture image data previously cached by a “store” or “set” operation that used the same packet ID value specified in the “Attributes 1” field.

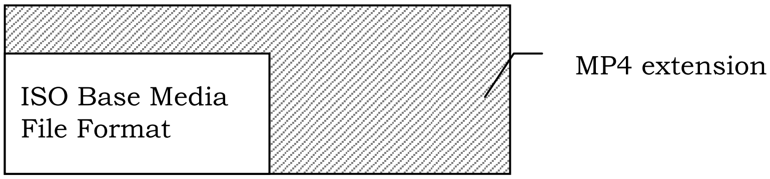
11.6 File Extension and MIME type for Media presentation

The media files created as per the above format specification shall use the extension of “.cmf”, short for Compact Multimedia Format. Note: the MIME type “application/cmf” is expected to be registered and used.

Annex A File formats: difference with 3GPP (Informative)

Annex A.1 Relations between ISO, 3GPP, and 3GPP2 file format

ISO defines the ISO Base Media File Format as a basis of developing a media container for various purposes. It describes a basic architecture of the multimedia file, and mandatory /optional elements in it. There are some extensions over ISO Base Media File Format, one of which is an MP4 file format to support MPEG-4 visual/audio codecs and various MPEG-4 Systems features such as object descriptors and scene



descriptions.

Figure A 1: File formats in ISO

3GPP extended ISO Base Media File Format to incorporate new media codecs and a timed text feature. They also use MPEG-4 visual/audio codecs, a portion of MP4 extension is included. The relation is depicted in Figure A.2. It is noted that 3GPP file format does not use some features in ISO Base Media File Format.

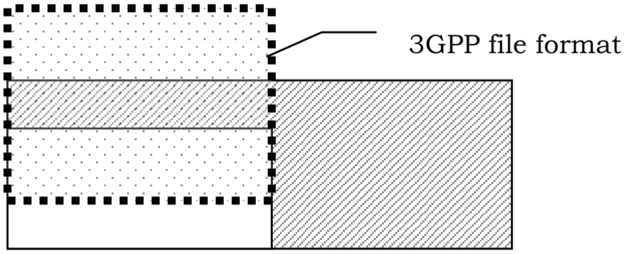


Figure A 2: 3GPP file format

3GPP2 employs full aspects of ISO Base Media File Format. It also adds new codecs and extends a 3GPP timed text. On the other hand, it only uses the same portion of MP4 extension as 3GPP does. Figure A.3 illustrates it.

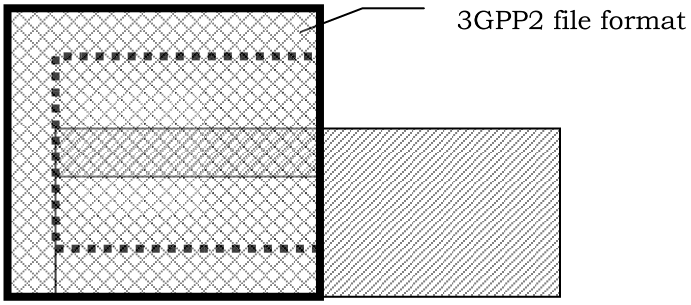


Figure A 3: 3GPP2 file format

Annex A.2 Differences between 3GPP2 and 3GPP

a) Features in ISO Base Media File Format

- Movie fragment

3GPP2 Release 0 and A and 3GPP Release 6 allow movie fragmentation, which is useful for various applications such as pseudo-streaming and live authoring of a movie file; 3GPP Releases 4 and 5 do not support movie fragmentation.

b) New extensions

- QCELPSampleEntry and 13K speech support in MP4AudioSampleEntry
- SMVSampleEntry
- EVRCSampleEntry
- EVRCBSampleEntry
- EVRCWBSampleEntry

These codecs are used in 3GPP2 and there are no definitions in 3GPP file format, so their encapsulations are defined.

c) Enhancements to 3GPP features

- Enhancements to 3GPP Timed Text
  - Link functionality for phone and mail is enhanced compared to 3GPP Timed Text
  - Word wrap is enhanced compared to 3GPP Release 4 and 5 Timed Text. (3GPP Release 6 includes word wrap feature.)

d) File identifications

3GPP2 has its own file extension, MIME types, and file brand identifier.

Annex A.3 Usage of 3GPP branding

Conditions for using 3GPP branding are that the media types contained in the ".3g2" file are restricted to those identified for use in the ".3gp" file format [5]. Specifically, AMR and AMR-WB speech; H.263, MPEG-4, and MPEG-4 AVC/H.264 video, MPEG-4

AAC and HE AAC audio; and timed text.

Note: Since movie fragments and the optional textwrap feature are not allowed in 3GPP Release 4/5, a file with one of these features should not contain ‘3gp4’ or ‘3gp5’ as a compatible brand.

It is left to implementers to understand the implications of the absence of minor versioning support in the compatible branding list.

The table below shows what features and codecs are supported by the different 3GPP file format versions and, for comparison purposes, those supported by release 0, release A and release B of the 3GPP2 file format.

Feature	Method	3GPP2	3GPP2	3GPP2	3GPP	3GPP	3GPP	3GPP
		Rel 0	Rel A	Rel B	Rel 4	Rel 5	Rel 6	Rel 7
		(3g2a)	(3g2b)	(3g2c)	(3gp4)	(3gp5)	(3gp6)	(3gp7)
Speech	AMR	●	●	●	●	●	●	●
	AMR WB	●	●	●	●	●	●	●
	AMR WB+[1]						●	●
	EVRC	●	●	●				
	EVRC-B			●				
	EVRC-WB			●				
	13K[2]	●	●	●				
	SMV	●	●	●				
	VMR-WB[3]		●	●				
Audio	AAC	●	●	●	●	●	●	●
	HE AAC[4]		●	●			●	●
	Enhanced aacPlus						●	●
Video	H.263	●	●	●	●	●	●	●
	MPEG-4 Visual	●	●	●	●	●	●	●
	H.264/AVC		●	●			●	●
Text	Timed Text	●	●	●		●	●	●
Transport	Fragmentation	●	●	●			●	●

Table A-11-5: Brand usage in 3G2 files: ● = defined support

[1] Can also be used for audio.

[2] 13K (QCELP) can be stored using either MP4AudioSampleEntry or QCELPSampleEntry.

[3] VMR-WB mode 3 data can be stored in AMRSampleEntry.

[4] HE AAC is also known as AAC-SBR, AAC+, and aacPlus.

Annex A.4 Relationship of 3GPP2 and 3GPP Profiles

This section describes the relationship between 3GPP2 file format and each profile specified in 3GPP Release 6 file format.

3GPP files --> 3GPP2 clients	3GPP2 files --> 3GPP clients
3GPP general profile	
	3GPP2 files with the features as described in Annex A.2 and Annex A.3 are compatible with 3GPP general profile clients.
3GPP basic profile	
3GPP basic profile files are compatible with this specification. Therefore, the “3gp7” files should include “3g2c” (or “3g2a” or “3g2b”) as the compatible brand.	3GPP2 files with the features as described in Annex A.2 and Annex A.3 in addition to the 3GPP basic profile constraints are compatible with 3GPP basic profile clients.
3GPP progressive-download profile	
3GPP progressive-download profile files satisfies all requirements for 3GPP2 pseudo streaming in Annex B.2, and the 3GPP files are compatible with the 3GPP2 file format specification. Therefore the “3gr7” files should include “3g2c” (or “3g2b” or “3g2a”) as the compatible brand.	If 3GPP2 files fulfill all the following conditions, they are compatible with 3GPP progressive-download profile clients: ● The descriptions in Annex A.2 and Annex A.3. ● All media tracks (if more than one) are interleaved with an interleaving depth of one second or less.

Table A-11-6: Relationship of 3GPP2 and 3GPP profiles.

Annex B Guideline for File Format Usage (Informative)

FFMS is a generic standard and includes all features. Since some features may be useful but others may not in some services, this section shows a usage guideline in various services.

Annex B.1 MSS (Multimedia Streaming Service)

Annex B.2 Server storage for RTP streaming

A streaming server stores multimedia content in 3GPP2 file format, reads out media data from the file, and transmits to a client in an RTP packet. Hint tracks can be useful to the server when creating RTP packets.

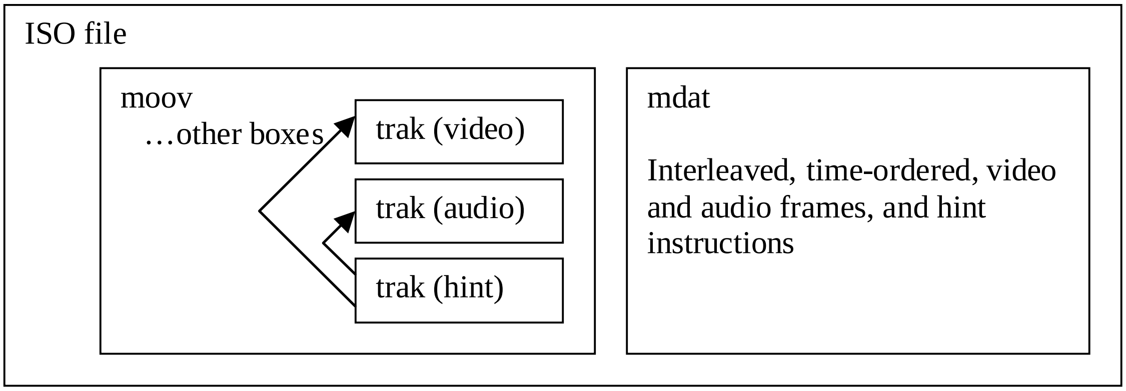

ISO file
moov
…other boxes
trak (video)
trak (audio)
trak (hint)
mdat
Interleaved, time-ordered, video and audio frames, and hint instructions

Figure B 1: Hinted Presentation for Streaming (Reprint from ISO/IEC 14496-12)

Annex B.3 Transmission format for pseudo-streaming

The definition of pseudo-streaming is a stream of content distributed by progressive download via a reliable delivery protocol (e.g. http) meant for real-time consumption. It is assumed that the download is carried out by some non realtime protocol such as HTTP (TCP).

HTTP is used for control and data transmission in the following example.

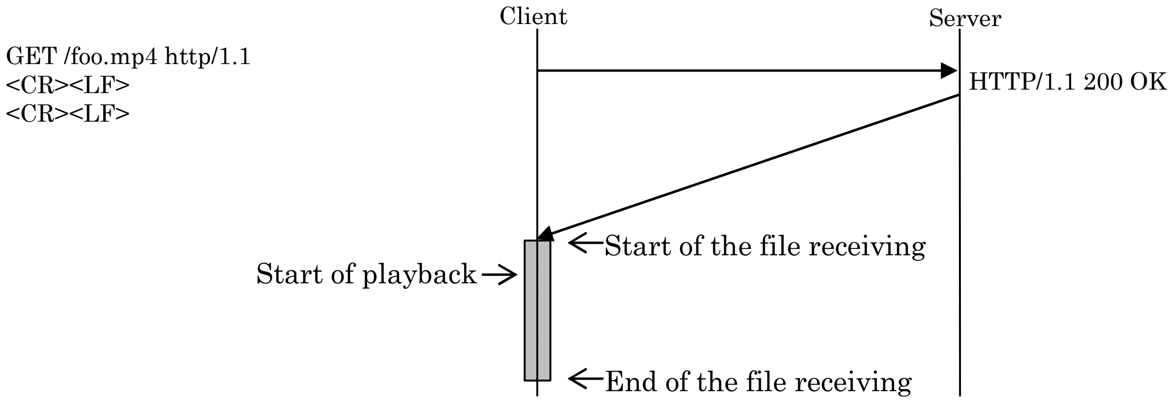


Figure B 2: Basic sequence of pseudo-streaming

HTTP specifies the file as a control protocol, and the file is transmitted in the response to the request. Playback starts during file download. This avoids a lengthy wait for a file to finish downloading (due to the size of the file and/or the transmission channel bitrate). Additionally, some receivers may not have sufficient memory to store an entire file. To ensure smooth playback, the client must receive all necessary header information and the first part of media data of sufficient temporal length to accommodate the maximum jitter in the system. Accordingly, requirements for the file format are as follows.

1) moov position

   moov has information necessary for decoding the media data in the file and therefore needs to be positioned at the beginning of the file.

2) media interleave

   If multiple media are included in the file, then they should be interleaved as a “chunk”. The size of a chunk relates waiting time to playback. A typical size of a chunk is a few seconds.

3) movie fragmentation

   The size of moov becomes large for lengthy movies. Therefore, long movies should be fragmented with each fragment having a header (moov or moof).

4) I-frame beginning

   The first video frame in mdat should start with an I-frame so that the client can start decoding from the first frame.

5) Timed Text

   Timed Text can be supported for the pseudo-streaming service. Text samples are also interleaved as well as audio and video samples.

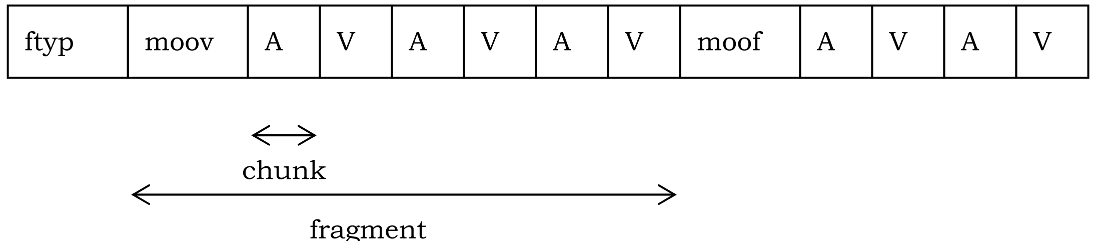


Figure B 3: Fragmented movie file format.

Annex B.4 MMS

3GPP2 files used for the purpose of MMS contain rather short duration clips that are transferred from a client to a server and vice versa. Considering the MMS feature should require less complexity, the following restrictions are useful.

- Movie fragment is not useful for short duration video, therefore it should not be used.
- The maximum number of tracks should be one for video, one for audio and one for text.
- The maximum number of sample entries should be one per track for video and audio (but unrestricted for text).
- Compact sample sizes ('stz2') should not be used.

Annex B.5 File download and play back

A 3GPP2 file is downloaded to a client and played back locally. Random positioning in the downloaded clip is an attractive feature. However, addressing information included in the default boxes such as ‘stts’ and ‘stsc’ only have a relative time difference, thus some processing is required to get the absolute address within the file that corresponds to the indicated relative time difference. mfra Box provides direct address information in the file and is useful for random positioning during play back.

Annex C SMIL Profile Differences Between 3GPP2 and 3GPP (Informative)

Annex C.1 Additional functionality

This informative annex includes the differences between the 3GPP2 SMIL specification and the 3GPP SMIL profile specification. The 3GPP2 SMIL profile described in the present document is a superset of the 3GPP SMIL profile [49] and a subset of the SMIL 2.0 Language Profile. The additional modules to the 3GPP SMIL profile are the following, all of which are optional:

- SMIL 2.0 Animation Module – BasicAnimation
- SMIL 2.0 Layout Module – AudioLayout
- SMIL 2.0 Timing and Synchronization Modules – AccessKeyTiming and MultiArcTiming

BasicAnimation module is added for the purpose of enhancing (motion) presentation capabilities. User agents that implement the semantics of this module should at least support **animate** element specified in SMIL 2.0. In the 3GPP2 SMIL specification, animating a video object and animating over a video object are not supported.

AudioLayout module controls aural media volumes via the '**soundLevel**' attribute. If AudioLayout module is used together with BasicAnimation module, content authors can animate audio volume, such as fade in/out in a SMIL presentation. See Annex D for details.

AccessKeyTiming module enhances interactivity by assigning a use event to a specific access key, such as a dial key. It reduces restrictions on input devices on terminals. MultiArcTiming module allows any number of offset-values, event-values, and accesskey-values for '**begin**' and '**end**' attributes, by separating them by a semicolon.

Also, the following name value pairs for MediaParam module are additional to 3GPP SMIL profile.

MIME type of the media object	value of the '**name**' attribute	value of the '**value**' attribute	Intended rendering of the media content.
application/text, application/xhtml+xml, application/vnd.wap.xhtml+xml, text/plain	font-family	Allowed values are all generic font family names defined by CSS2 [50].	The text document is rendered with the font-family that is determined by the font matching algorithm of CSS2 [50].
image/jpeg, image/gif, image/png, text/plain	tile	true or false	The media element is tiled (repeated). All tiling covers the region. For the tiled media, no animation and no transition shall be used.
image/jpeg, image/gif, image/png, text/plain	opacity	Alpha value within the range 0.0 (fully transparent) to 1.0 (fully opaque). Default value is 1.0.	The media element is rendered where opaque colors are made transparent.

Table C-11-7: Name value pairs for MediaParam module that are additional to 3GPP.

Annex C.2 Interoperability between 3GPP2 and 3GPP SMIL

W3C SMIL 2.0 recommendation allows user agents to securely ignore unknown element or attribute using SkipContentControl mechanism; the default value of '**skip-content**' attribute of SkipContentControl module (specified in both 3GPP2 SMIL and 3GPP SMIL) is "true", which means that the content of the element is ignored. Similarly unimplemented attributes should be treated as if they were not specified. Therefore interoperability between 3GPP2 SMIL profile and 3GPP SMIL profile is guaranteed, as far as a user agent can correctly parse the namespaces for both 3GPP2 SMIL and 3GPP SMIL.

Annex D 3GPP2 SMIL Authoring Guidelines (Informative)

Annex D.1 General

This is an informative annex for SMIL presentation authors. Authors can expect that 3GPP2 clients can handle the SMIL module collection defined in this document, 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 Section 9.1.3 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 a time.

Annex D.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 behavior 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 Web site.

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 the player being able to handle **area** elements.

Annex D.3 BasicLayout

When defining the layout of a SMIL presentation, a content author needs to be aware that the targeted devices might have diverse properties that affect how the content can be rendered. The different sizes of the display area that can be used to render content on the targeted devices should be considered for defining the layout of the SMIL presentation. The root-layout window might represent the entire display or only part of it.

Content authors are encouraged to create SMIL presentations that will work well with different resolutions of the rendering area. As mentioned in the SMIL 2.0 recommendation, content authors should use SMIL ContentControl functionality for defining multiple layouts for their SMIL presentation that are tailored to the specific needs of the whole range of targeted devices. Furthermore, authors should include a default layout (i.e. a layout determined by the SMIL player) that will be used when none of the author-defined layouts can be used.

A 3GPP2 SMIL player should use the layout definition of a SMIL presentation for presenting the content whenever possible. When the SMIL player fails to use the layout information defined by the author it is free to present the content using a layout it determines by itself.

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. Therefore **`fit=scroll`** should not be used except for text content.

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, video content should be encoded in a size suitable for the targeted terminals and it is recommended to use "fit=hidden".

Annex D.4 EventTiming

The two values **`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 of these events rely on the SMIL player receiving information 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., using the **`dur`** attribute in SMIL or by some external source such 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 initiating the event. For these reasons, it is recommended that the **`endEvent`** and **`repeatEvent`** values 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 be, e.g., the **`dur`** attribute in SMIL or the **`a=range`** field in SDP.

The **`inBoundsEvent`** and **`outOfBoundsEvent`** values 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.

Annex D.5 AccessKeyTiming

Access-key values used in a SMIL presentation will be valid dial keys: 0-9, *, and #, e.g. **`accesskey(1)`**. Since 3GPP2 SMIL Profile supports the LinkingAttributes module, content authors can also define access keys to activate hyperlinks using **`accesskey`** attribute. To avoid conflicts, content authors should not use the same key for an event trigger and a hyperlink.

Annex D.6 MultiArcTiming

Any combination of offset-values, event-values, and accesskey-values are possible for **begin** and **end** attributes if using the MultiArcTiming module. Content authors are recommended to describe multiple values in an order; offset-values, event-values, and accesskey-values, in considering a user terminal which does not support MultiArcTiming functionality. Content authors should also take care not to create contradictory combinations of these values. For example, the following sample is illegal since the same value **accesskey(0)** is used in both **begin** and **end** attributes.

```
<video id="video" src="video.3g2" region="vid" type="video/3gpp2"
begin="5s; img1.beginEvent; accesskey(0)" end="30s; accesskey(0)">
```

Annex D.7 BasicAnimation

Animating a video object and animating over a video object should not be used due to hardware restrictions in mobile devices. However, the maximum number of media objects to be animated simultaneously is not restricted in anticipation of advanced capabilities of a future terminal. The usage of **by** and/or **calcMode** attributes in a short period may not take effect due to the processing complexity.

BasicAnimation can be used together with AudioLayout for animating audio volume. In the following example, the audio track of "sample.3g2" file on a region "av" fades in (from silence (=0%) to an original volume (=100%)) for 3 seconds from the beginning. Note that values outside 0-100%, though permitted, should not be used due to potential limitations of the user terminal sound device. This description implies that a value of **targetElement** attribute is a region "av" since **animate** element is a child of **video** element identified by "video1".

```

<head>
  <layout>
    ...
    <region id="av" top="0" width="80" height="60" soundLevel="0%"/>
  </layout>
</head>
<body>
  ...
  <video id="video1" src="sample.3g2" region="av" type="video/3gpp2"
begin="0" end="30s">
    <animate attributeName="soundLevel" begin="0" dur="3s" from="0%"
to="100%"/>
  </video>
  ...
</body>

```

Annex D.8 MediaParam

In the 3GPP2 SMIL profile [12], 5 name and value pairs of **param** element are specified for a SMIL presentation. Other values are ignored and implementation dependent. The following example shows the descriptions for displaying the text file "sample.txt" in red, Times character.

```
<text id="txt1" src="sample.txt" region="txt" type="text/plain"
begin="0" end="30s">
  <param name="color" value="#ff0000">
  <param name="font-family" value="Times">
</text>
```

The following example shows the descriptions for tiling a JPEG image. This functionality is useful since a small sized image file can be used for background tiling. Note that a tiling functionality is valid for an image or a text, and only when the value of **fit** attribute is "hidden". And content authors should apply neither transition nor animation to a tiled media object.

```
<img id="bgimg" src="background.jpg" region="background"
type="image/jpeg">
  <param name="tile" value="true">
</text>
```

As in the above examples, it is strongly encouraged to utilize a **type** attribute in a media element in order to allow a 3GPP2 SMIL player to correctly recognize the MIME type of a media object to be played.

Annex D.9 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.

Annex E Additional Specification for the System Component Test Attribute (Normative)

Annex E.1 General

This annex includes additional normative specification on the encoding of the SMIL 2.0 BasicContentControl module '**systemComponent**' test attribute value. The purpose is to allow a SMIL presentation to test if a 3GPP2 SMIL player supports a media type.

Annex E.2 Definition of Attribute Encoding

To test support for a certain media type, the value of the systemComponent attribute shall be encoded as a URI as follows:

```
systemComponentAttrValue --> "ContentType:" mimeMediaTypeName "/"
mimeSubTypeName options?
```

```
options --> "?" parameters
```

where:

- "ContentType:" is a static pre-fix that shall always be encoded,
- '*mimeMediaTypeName*' and '*mimeSubtypeName*' are a MIME type and subtype. These two shall be encoded and shall be separated by a dash ("/"), and
- encoding *'options'* is optional.
- '*parameters*' stands for any parameter to the MIME type that can optionally be encoded. When encoded, parameters shall be separated from the MIME type and sub-type names by a question mark ("?").

Annex E.3 Behavior of a 3GPP2 SMIL Player

For any '**systemComponent**' test attribute value that is prefixed with the string 'ContentType:' a 3GPP2 SMIL player is required to evaluate the '**systemComponent**' test attribute based on *'mimeMediaTypeName*' and '*mimeSubtypeName*' as follows:

- Evaluation of the test attribute returns true whenever the 3GPP2 SMIL player supports rendering media content of this MIME type,
- In all other cases the evaluation returns false.

A 3GPP2 SMIL player must be able to ignore any encoded parameters for performing this evaluation. A 3GPP2 SMIL player is allowed, but not required, to also include parameters into the evaluation.

NOTE: The specification on parameters makes a 3GPP2 SMIL player forward compatible with any future version of the specification that will possibly define how to encode MIME type parameters and how to evaluate the '**systemComponent**' test attribute when parameters are included into its value.

NOTE: This specification intentionally leaves it open how '**systemComponent**' test attribute values that are not prefixed with the string "ContentType:" are evaluated. Again, this makes a 3GPP2 SMIL player forward compatible with any future version of the specification that will possibly define other URI schemes for the '**systemComponent**' attribute value.

Annex F Description of CMF to SMIL Conversion (Informative)

This informative annex discusses the process of converting CMF files to SMIL presentations. The conversion is useful for compatibility with other multimedia delivery methods that support SMIL.

Annex F.1 Conversion Mechanics

A CMF file contains media objects, timing information (events), and meta-data. The media objects are extracted from the CMF file and stored in individual files along with the timing information that is translated into SMIL syntax and saved as a SMIL file. The start time and duration of playback of each media object are determined when the events are parsed from the CMF file.

These steps summarize the conversion process:

1. Extract all media objects from the CMF file and store them in separate files. This is a straightforward process, since transcoding will not be necessary most of the time.
2. Build a timeline for the presentation by calculating the start and end times of each media object.
3. Extract loop information from the CMF file and section the timeline at loop boundaries.
4. For each media object overlapping a section boundary, logically split the object, or physically split the file, into two objects at the overlap point. Update the timeline to reference the split objects.
5. A SMIL file is constructed according to the modified timeline that describes the timing and repetition of the media objects.
6. Package the SMIL and media files into a format suitable for network transfer.

The use of jumps, or loops, within the CMF file becomes a complicating factor because the loop boundaries may occur within media objects. SMIL does have the capability to repeat playback of media objects, but not change the playback position during playback. Any repeated sections will be referenced separately from the rest of the object. This is achieved by splitting affected objects into multiple objects or by specifying the desired portion of the object in the reference to it.

Smooth playback of split media objects depends on the ability of media decoders to be started or restarted quickly to minimize the delays between the parts of a split media object. Also, some media formats or types are difficult to physically split at arbitrary positions, and the outcome of splitting may introduce clicks or dropped notes.

Text is placed within regions defined in SMIL. The sizes and placement of these regions must be calculated by the conversion software, but the needed information is not always available, for example font sizes.