Workgroup: cellar
Internet-Draft: draft-ietf-cellar-codec-06
Published: 12 April 2021
Intended Status: Informational
Expires: 14 October 2021
Authors: S. Lhomme M. Bunkus D. Rice

Matroska Media Container Codec Specifications

Abstract

This document defines the Matroska codec mappings, including the codec ID, layout of data in a Block Element and in an optional CodecPrivate Element.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 October 2021.

Table of Contents

1. Introduction

Matroska aims to become THE standard of multimedia container formats. It stores interleaved and timestamped audio/video/subtitle data using various codecs. To interpret the codec data, a mapping between the way the data is stored in Matroska and how it is understood by such a codec is necessary.

This document intends to define this mapping for many commonly used codecs in Matroska.

2. Status of this document

This document is a work-in-progress specification defining the Matroska file format as part of the IETF Cellar working group. It uses basic elements and concept already defined in the Matroska specifications defined by this workgroup.

3. Security Considerations

This document inherits security considerations from the EBML and Matroska documents.

4. IANA Considerations

To be determined.

5. Notation and Conventions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

6. Codec Mappings

A Codec Mapping is a set of attributes to identify, name, and contextualize the format and characteristics of encoded data that can be contained within Matroska Clusters.

Each TrackEntry used within Matroska **MUST** reference a defined Codec Mapping using the Codec ID to identify and describe the format of the encoded data in its associated Clusters. This Codec ID is a unique registered identifier that represents the encoding stored within the Track. Certain encodings **MAY** also require some form of codec initialization in order to provide its decoder with context and technical metadata.

The intention behind this list is not to list all existing audio and video codecs, but rather to list those codecs that are currently supported in Matroska and therefore need a well defined Codec ID so that all developers supporting Matroska will use the same Codec ID. If you feel we missed support for a very important codec, please tell us on our development mailing list (cellar at ietf.org).

6.1. Defining Matroska Codec Support

Support for a codec is defined in Matroska with the following values.

6.1.1. Codec ID

Each codec supported for storage in Matroska **MUST** have a unique Codec ID. Each Codec ID **MUST** be prefixed with the string from the following table according to the associated type of the codec. All characters of a Codec ID Prefix **MUST** be capital letters (A-Z) except

for the last character of a Codec ID Prefix which **MUST** be an underscore ("_").

Codec Type	Codec ID Prefix
Video	"V_"
Audio	"A_"
Subtitle	"S_"
Button	"B_"

Table 1

Each Codec ID **MUST** include a Major Codec ID immediately following the Codec ID Prefix. A Major Codec ID **MAY** be followed by an **OPTIONAL** Codec ID Suffix to communicate a refinement of the Major Codec ID. If a Codec ID Suffix is used, then the Codec ID **MUST** include a forward slash ("/") as a separator between the Major Codec ID and the Codec ID Suffix. The Major Codec ID **MUST** be composed of only capital letters (A-Z) and numbers (0-9). The Codec ID Suffix **MUST** be composed of only capital letters (A-Z), numbers (0-9), underscore ("_"), and forward slash ("/").

The following table provides examples of valid Codec IDs and their components:

Codec ID Prefix	Major Codec ID	Separator	Codec ID Suffix	Codec ID
A_	AAC	/	MPEG2/LC/SBR	A_AAC/MPEG2/LC/SBR
V_	MPEG4	/	ISO/ASP	V_MPEG4/ISO/ASP
V_	MPEG1			V_MPEG1

Table 2

6.1.2. Codec Name

Each encoding supported for storage in Matroska **MUST** have a Codec Name. The Codec Name provides a readable label for the encoding.

6.1.3. Description

An optional description for the encoding. This value is only intended for human consumption.

6.1.4. Initialization

Each encoding supported for storage in Matroska **MUST** have a defined Initialization. The Initialization **MUST** describe the storage of data necessary to initialize the decoder, which **MUST** be stored within the CodecPrivate Element. When the Initialization is updated within a track, then that updated Initialization data **MUST** be written into the CodecState Element of the first Cluster to require it. If the

encoding does not require any form of Initialization, then none **MUST** be used to define the Initialization and the CodecPrivate Element **SHOULD NOT** be written and **MUST** be ignored. Data that is defined Initialization to be stored in the CodecPrivate Element is known as Private Data.

6.1.5. Codec BlockAdditions

Additional data that contextualizes or supplements a Block can be stored within the BlockAdditional Element of a BlockMore Element. This BlockAdditional data **MAY** be passed to the associated decoder along with the content of the Block Element. Each BlockAdditional is coupled with a BlockAddID that identifies the kind of data it contains. The following table defines the meanings of BlockAddID values.

BlockAddID Value	Definition
0	Invalid.
1	Indicates that the context of the BlockAdditional data is defined by the corresponding Codec Mapping.
2 or greater	BlockAddID values of 2 and greater are mapped to the BlockAddIDValue of the BlockAdditionMapping of the associated Track.

Table 3

The values of BlockAddID that are 2 of greater have no semantic meaning, but simply associate the BlockMore Element with a BlockAdditionMapping of the associated Track. See Section 8 on Block Additional Mappings for more information.

The following XML depicts the nested Elements of a BlockGroup Element with an example of BlockAdditions:

```
<BlockGroup>
  <Block>{Binary data of a VP9 video frame in YUV}</Block>
  <BlockAdditions>
    <BlockMore>
      <BlockAddID>1</BlockAddID>
      <BlockAdditional>
        {alpha channel encoding to supplement the VP9 frame}
      </BlockAdditional>
    </BlockMore>
  </BlockAdditions>
</BlockGroup>
```

6.1.6. Citation

Documentation of the associated normative and informative references for the codec is **RECOMMENDED**.

6.1.7. Deprecation Date

A timestamp, expressed in [RFC3339] that notes when support for the Codec Mapping within Matroska was deprecated. If a Codec Mapping is defined with a Deprecation Date, then it is **RECOMMENDED** that Matroska writers **SHOULD NOT** use the Codec Mapping after the Deprecation Date.

6.1.8. Superseded By

A Codec Mapping **MAY** only be defined with a Superseded By value, if it has an expressed Deprecation Date. If used, the Superseded By value **MUST** store the Codec ID of another Codec Mapping that has superseded the Codec Mapping.

6.2. Recommendations for the Creation of New Codec Mappings

Creators of new Codec Mappings to be used in the context of Matroska:

* **SHOULD** assume that all Codec Mappings they create might become standardized, public, commonly deployed, or usable across multiple implementations.

* **SHOULD** employ meaningful values for Codec ID and Codec Name that they have reason to believe are currently unused.

* **SHOULD NOT** prefix their Codec ID with "X_" or similar constructs.

These recommendations are based upon Section 3 of [RFC6648].

6.3. Video Codec Mappings

6.3.1. V_MS/VFW/FOURCC

Codec ID: V_MS/VFW/FOURCC

Codec Name: Microsoft (TM) Video Codec Manager (VCM)

Description: The private data contains the VCM structure BITMAPINFOHEADER including the extra private bytes, as defined by Microsoft.aspx). The data are stored in little-endian format (like on IA32 machines). Where is the Huffman table stored in HuffYUV, not AVISTREAMINFO ??? And the FourCC, not in AVISTREAMINFO.fccHandler ???

Initialization: Private Data contains the VCM structure BITMAPINFOHEADER including the extra private bytes, as defined by Microsoft in https://msdn.microsoft.com/en-us/library/windows/desktop/dd183376(v=vs.85).aspx.

Citation: https://msdn.microsoft.com/en-us/library/windows/desktop/dd183376(v=vs.85).aspx

6.3.2. V_UNCOMPRESSED

Codec ID: V_UNCOMPRESSED

Codec Name: Video, raw uncompressed video frames

Description: All details about the used color specs and bit depth are to be put/read from the KaxCodecColourSpace elements.

Initialization: none

6.3.3. V_MPEG4/ISO/SP

Codec ID: V_MPEG4/ISO/SP

Codec Name: MPEG4 ISO simple profile (DivX4)

Description: Stream was created via improved codec API (UCI) or even transmuxed from AVI (no b-frames in Simple Profile), frame order is coding order.

Initialization: none

6.3.4. V_MPEG4/ISO/ASP

Codec ID: V_MPEG4/ISO/ASP

Codec Name: MPEG4 ISO advanced simple profile (DivX5, XviD, FFMPEG)

Description: Stream was created via improved codec API (UCI) or transmuxed from MP4, not simply transmuxed from AVI. Note there are differences how b-frames are handled in these native streams, when being compared to a VfW created stream, as here there are no dummy frames inserted, the frame order is exactly the same as the coding order, same as in MP4 streams.

Initialization: none

6.3.5. V_MPEG4/ISO/AP

Codec ID: V_MPEG4/ISO/AP

Codec Name: MPEG4 ISO advanced profile

Description: Stream was created via improved codec API (UCI) or transmuxed from MP4, not simply transmuxed from AVI. Note there are differences how b-frames are handled in these native streams, when being compared to a VfW created stream, as here there are no dummy frames inserted, the frame order is exactly the same as the coding order, same as in MP4 streams.

Initialization: none

6.3.6. V_MPEG4/MS/V3

Codec ID: V_MPEG4/MS/V3

Codec Name: Microsoft (TM) MPEG4 V3

Description: Microsoft (TM) MPEG4 V3 and derivates, means DivX3, Angelpotion, SMR, etc.; stream was created using VfW codec or transmuxed from AVI; note that V1/V2 are covered in VfW compatibility mode.

Initialization: none

6.3.7. V_MPEG1

Codec ID: V_MPEG1

Codec Name: MPEG 1

Description: The Matroska video stream will contain a demuxed Elementary Stream (ES), where block boundaries are still to be defined. Its **RECOMMENDED** to use MPEG2MKV.exe for creating those files, and to compare the results with self-made implementations

Initialization: none

6.3.8. V_MPEG2

Codec ID: V_MPEG2

Codec Name: MPEG 2

Description: The Matroska video stream will contain a demuxed Elementary Stream (ES), where block boundaries are still to be defined. Its **RECOMMENDED** to use MPEG2MKV.exe for creating those files, and to compare the results with self-made implementations

Initialization: none

6.3.9. V_MPEG4/ISO/AVC

Codec ID: V_MPEG4/ISO/AVC

Codec Name: AVC/H.264

Description: Individual pictures (which could be a frame, a field, or 2 fields having the same timestamp) of AVC/H.264 stored as described in [ISO.14496-15].

Initialization: The Private Data contains a AVCDecoderConfigurationRecord structure, as defined in [ISO.14496-15]. For legacy reasons, because Block Addition Mappings are preferred, see Section 6.7, the AVCDecoderConfigurationRecord structure **MAY** be followed by an extension block beginning with a 4-byte extension block size field in big-endian byte order which is the size of the extension block minus 4 (excluding the size of the extension block size field) and a 4-byte field corresponding to a BlockAddIDType of "mvcC" followed by a content corresponding to the content of BlockAddIDExtraData for mvcC; see Section 6.7.8.

6.3.10. V_MPEGH/ISO/HEVC

Codec ID: V_MPEGH/ISO/HEVC

Codec Name: HEVC/H.265

Description: Individual pictures (which could be a frame, a field, or 2 fields having the same timestamp) of HEVC/H.265 stored as described in [ISO.14496-15].

Initialization: The Private Data contains a HEVCDecoderConfigurationRecord structure, as defined in [ISO.14496-15].

6.3.11. V_AVS2

Codec ID: V_AVS2

Codec Name: AVS2-P2/IEEE.1857.4

Description: Individual pictures of AVS2-P2 stored as described in the second part of [IEEE.1857-4].

Initialization: none.

6.3.12. V_REAL/RV10

Codec ID: V_REAL/RV10

Codec Name: RealVideo 1.0 aka RealVideo 5

Description: Individual slices from the Real container are combined into a single frame.

Initialization: The Private Data contains a real_video_props_t structure in big-endian byte order as found in librmff.

6.3.13. V_REAL/RV20

Codec ID: V_REAL/RV20

Codec Name: RealVideo G2 and RealVideo G2+SVT

Description: Individual slices from the Real container are combined into a single frame.

Initialization: The Private Data contains a real_video_props_t structure in big-endian byte order as found in librmff.

6.3.14. V_REAL/RV30

Codec ID: V_REAL/RV30

Codec Name: RealVideo 8

Description: Individual slices from the Real container are combined into a single frame.

Initialization: The Private Data contains a real_video_props_t structure in big-endian byte order as found in librmff.

6.3.15. V_REAL/RV40

Codec ID: V_REAL/RV40

Codec Name: rv40 : RealVideo 9

Description: Individual slices from the Real container are combined into a single frame.

Initialization: The Private Data contains a real_video_props_t structure in big-endian byte order as found in librmff.

6.3.16. V_QUICKTIME

Codec ID: V_QUICKTIME

Codec Name: Video taken from QuickTime(TM) files

Description: Several codecs as stored in QuickTime, e.g., Sorenson or Cinepak.

Initialization: The Private Data contains all additional data that is stored in the 'stsd' (sample description) atom in the QuickTime file **after** the mandatory video descriptor structure (starting with the size and FourCC fields). For an explanation of the QuickTime file format read QuickTime File Format Specification.

6.3.17. V_THEORA

Codec ID: V_THEORA

Codec Name: Theora

Initialization: The Private Data contains the first three Theora packets in order. The lengths of the packets precedes them. The actual layout is:

* Byte 1: number of distinct packets #p minus one inside the CodecPrivate block. This **MUST** be "2" for current (as of 2016-07-08) Theora headers.

* Bytes 2..n: lengths of the first #p packets, coded in Xiph-style lacing. The length of the last packet is the length of the CodecPrivate block minus the lengths coded in these bytes minus one.

* Bytes n+1..: The Theora identification header, followed by the commend header followed by the codec setup header. Those are described in the Theora specs.

6.3.18. V_PRORES

Codec ID: V_PRORES

Codec Name: Apple ProRes

Initialization: The Private Data contains the FourCC as found in MP4 movies:

* ap4x: ProRes 4444 XQ

* ap4h: ProRes 4444

* apch: ProRes 422 High Quality

* apcn: ProRes 422 Standard Definition

* apcs: ProRes 422 LT

*apco: ProRes 422 Proxy

*aprh: ProRes RAW High Quality

*aprn: ProRes RAW Standard Definition

this page for more technical details on ProRes

6.3.19. V_VP8

Codec ID: V_VP8

Codec Name: VP8 Codec format

Description: VP8 is an open and royalty free video compression format developed by Google and created by On2 Technologies as a successor to VP7. [RFC6386]

Codec BlockAdditions: A single-channel encoding of an alpha channel **MAY** be stored in BlockAdditions. The BlockAddId of the BlockMore containing these data **MUST** be 1.

Initialization: none

6.3.20. V_VP9

Codec ID: V_VP9

Codec Name: VP9 Codec format

Description: VP9 is an open and royalty free video compression format developed by Google as a successor to VP8. Draft VP9 Bitstream and Decoding Process Specification

Codec BlockAdditions: A single-channel encoding of an alpha channel **MAY** be stored in BlockAdditions. The BlockAddId of the BlockMore containing these data **MUST** be 1.

Initialization: none

6.3.21. V_FFV1

Codec ID: V_FFV1

Codec Name: FF Video Codec 1

Description: FFV1 is a lossless intra-frame video encoding format designed to efficiently compress video data in a variety of pixel formats. Compared to uncompressed video, FFV1 offers storage compression, frame fixity, and self-description, which makes FFV1

useful as a preservation or intermediate video format. Draft FFV1 Specification

Initialization: For FFV1 versions 0 or 1, Private Data **SHOULD NOT** be written. For FFV1 version 3 or greater, the Private Data **MUST** contain the FFV1 Configuration Record structure, as defined in https://tools.ietf.org/html/draft-ietf-cellar-ffv1-04#section-4.2, and no other data.

6.4. Audio Codec Mappings

6.4.1. A_MPEG/L3

Codec ID: A_MPEG/L3

Codec Name: MPEG Audio 1, 2, 2.5 Layer III

Description: The data contain everything needed for playback in the MPEG Audio header of each frame. Corresponding ACM wFormatTag : 0x0055

Initialization: none

6.4.2. A_MPEG/L2

Codec ID: A_MPEG/L2

Codec Name: MPEG Audio 1, 2 Layer II

Description: The data contain everything needed for playback in the MPEG Audio header of each frame. Corresponding ACM wFormatTag : 0x0050

Initialization: none

6.4.3. A_MPEG/L1

Codec ID: A_MPEG/L1

Codec Name: MPEG Audio 1, 2 Layer I

Description: The data contain everything needed for playback in the MPEG Audio header of each frame. Corresponding ACM wFormatTag : 0x0050

Initialization: none

6.4.4. A_PCM/INT/BIG

Codec ID: A_PCM/INT/BIG

Codec Name: PCM Integer Big Endian

Description: The audio bit depth **MUST** be read and set from the BitDepth Element. Audio samples **MUST** be considered as signed values, except if the audio bit depth is 8 which **MUST** be interpreted as unsigned values. Corresponding ACM wFormatTag : ???

Initialization: none

6.4.5. A_PCM/INT/LIT

Codec ID: A_PCM/INT/LIT

Codec Name: PCM Integer Little Endian

Description: The audio bit depth **MUST** be read and set from the BitDepth Element. Audio samples **MUST** be considered as signed values, except if the audio bit depth is 8 which **MUST** be interpreted as unsigned values. Corresponding ACM wFormatTag : 0x0001

Initialization: none

6.4.6. A_PCM/FLOAT/IEEE

Codec ID: A_PCM/FLOAT/IEEE

Codec Name: Floating Point, IEEE compatible

Description: The audio bit depth **MUST** be read and set from the BitDepth Element (32 bit in most cases). The floats are stored as defined in [IEEE.754] and in little-endian order. Corresponding ACM wFormatTag : 0x0003

Initialization: none

6.4.7. A_MPC

Codec ID: A_MPC

Codec Name: MPC (musepack) SV8

Description: The main developer for musepack has requested that we wait until the SV8 framing has been fully defined for musepack before defining how to store it in Matroska.

6.4.8. A_AC3

Codec ID: A_AC3

Codec Name: (Dolby™) AC3

Description: BSID <= 8 !! The private data is void ??? Corresponding ACM wFormatTag : 0x2000 ; channel number have to be read from the corresponding audio element

6.4.9. A_AC3/BSID9

Codec ID: A_AC3/BSID9

Codec Name: (Dolby™) AC3

Description: The ac3 frame header has, similar to the mpeg-audio header a version field. Normal ac3 is defined as bitstream id 8 (5 Bits, numbers are 0-15). Everything below 8 is still compatible with all decoders that handle 8 correctly. Everything higher are additions that break decoder compatibility. For the samplerates 24kHz (00); 22,05kHz (01) and 16kHz (10) the BSID is 9 For the samplerates 12kHz (00); 11,025kHz (01) and 8kHz (10) the BSID is 10

Initialization: none

6.4.10. A_AC3/BSID10

Codec ID: A_AC3/BSID10

Codec Name: (Dolby™) AC3

Description: The ac3 frame header has, similar to the mpeg-audio header a version field. Normal ac3 is defined as bitstream id 8 (5 Bits, numbers are 0-15). Everything below 8 is still compatible with all decoders that handle 8 correctly. Everything higher are additions that break decoder compatibility. For the samplerates 24kHz (00); 22,05kHz (01) and 16kHz (10) the BSID is 9 For the samplerates 12kHz (00); 11,025kHz (01) and 8kHz (10) the BSID is 10

Initialization: none

6.4.11. A_ALAC

Codec ID: A_ALAC

Codec Name: ALAC (Apple Lossless Audio Codec)

Initialization: The Private Data contains ALAC's magic cookie (both the codec specific configuration as well as the optional channel layout information). Its format is described in ALAC's official source code.

6.4.12. A_DTS

Codec ID: A_DTS

Codec Name: Digital Theatre System

Description: Supports DTS, DTS-ES, DTS-96/26, DTS-HD High Resolution Audio and DTS-HD Master Audio. The private data is void. Corresponding ACM wFormatTag : 0x2001

Initialization: none

6.4.13. A_DTS/EXPRESS

Codec ID: A_DTS/EXPRESS

Codec Name: Digital Theatre System Express

Description: DTS Express (a.k.a. LBR) audio streams. The private data is void. Corresponding ACM wFormatTag : 0x2001

Initialization: none

6.4.14. A_DTS/LOSSLESS

Codec ID: A_DTS/LOSSLESS

Codec Name: Digital Theatre System Lossless

Description: DTS Lossless audio that does not have a core substream. The private data is void. Corresponding ACM wFormatTag : 0x2001

Initialization: none

6.4.15. A_VORBIS

Codec ID: A_VORBIS

Codec Name: Vorbis

Initialization: The Private Data contains the first three Vorbis packet in order. The lengths of the packets precedes them. The actual layout is: - Byte 1: number of distinct packets #p minus one inside the CodecPrivate block. This **MUST** be "2" for current (as of 2016-07-08) Vorbis headers. - Bytes 2..n: lengths of the first #p packets, coded in Xiph-style lacing. The length of the last packet is the length of the CodecPrivate block minus the lengths coded in these bytes minus one. - Bytes n+1..: The Vorbis identification header, followed by the Vorbis comment header followed by the codec setup header.

6.4.16. A_FLAC

Codec ID: A_FLAC

Codec Name: FLAC (Free Lossless Audio Codec)

Initialization: The Private Data contains all the header/metadata packets before the first data packet. These include the first header packet containing only the word fLaC as well as all metadata packets.

6.4.17. *AREAL/144*

Codec ID: A*REAL/144*

Codec Name: Real Audio 1

Initialization: The Private Data contains either the "real*audio*v4*prop*st" or the "real*audio*v5*prop*st" structure (differentiated by their "version" field; big-endian byte order) as found in librmff.

6.4.18. *AREAL/288*

Codec ID: A*REAL/288*

Codec Name: Real Audio 2

Initialization: The Private Data contains either the "real*audio*v4*prop*st" or the "real*audio*v5*prop*st" structure (differentiated by their "version" field; big-endian byte order) as found in librmff.

6.4.19. A_REAL/COOK

Codec ID: A_REAL/COOK

Codec Name: Real Audio Cook Codec (codename: Gecko)

Initialization: The Private Data contains either the "real*audio*v4*prop*st" or the "real*audio*v5*prop*st" structure (differentiated by their "version" field; big-endian byte order) as found in librmff.

6.4.20. A_REAL/SIPR

Codec ID: A_REAL/SIPR

Codec Name: Sipro Voice Codec

Initialization: The Private Data contains either the "real*audio*v4*prop*st" or the "real*audio*v5*prop*st" structure (differentiated by their "version" field; big-endian byte order) as found in librmff.

6.4.21. A_REAL/RALF

Codec ID: A_REAL/RALF

Codec Name: Real Audio Lossless Format

Initialization: The Private Data contains either the "real*audio*v4*prop*st" or the "real*audio*v5*prop*st" structure (differentiated by their "version" field; big-endian byte order) as found in librmff.

6.4.22. A_REAL/ATRC

Codec ID: A_REAL/ATRC

Codec Name: Sony Atrac3 Codec

Initialization: The Private Data contains either the "real*audio*v4*prop*st" or the "real*audio*v5*prop*st" structure (differentiated by their "version" field; big-endian byte order) as found in librmff.

6.4.23. A_MS/ACM

Codec ID: A_MS/ACM

Codec Name: Microsoft(TM) Audio Codec Manager (ACM)

Description: The data are stored in little-endian format (like on IA32 machines).

Initialization: The Private Data contains the ACM structure WAVEFORMATEX including the extra private bytes, as defined by Microsoft.

6.4.24. A_AAC/MPEG2/MAIN

Codec ID: A_AAC/MPEG2/MAIN

Codec Name: MPEG2 Main Profile

Description: Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal Matroska frame based muxing scheme is applied. AAC audio always uses wFormatTag 0xFF.

Initialization: none

6.4.25. A_AAC/MPEG2/LC

Codec ID: A_AAC/MPEG2/LC

Codec Name: Low Complexity

Description: Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal Matroska frame based muxing scheme is applied. AAC audio always uses wFormatTag 0xFF.

Initialization: none

6.4.26. A_AAC/MPEG2/LC/SBR

Codec ID: A_AAC/MPEG2/LC/SBR

Codec Name: Low Complexity with Spectral Band Replication

Description: Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal Matroska frame based muxing scheme is applied. AAC audio always uses wFormatTag 0xFF.

Initialization: none

6.4.27. A_AAC/MPEG2/SSR

Codec ID: A_AAC/MPEG2/SSR

Codec Name: Scalable Sampling Rate

Description: Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal Matroska frame based muxing scheme is applied. AAC audio always uses wFormatTag 0xFF.

Initialization: none

6.4.28. A_AAC/MPEG4/MAIN

Codec ID: A_AAC/MPEG4/MAIN

Codec Name: MPEG4 Main Profile

Description: Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal Matroska frame based muxing scheme is applied. AAC audio always uses wFormatTag 0xFF.

Initialization: none

6.4.29. A_AAC/MPEG4/LC

Codec ID: A_AAC/MPEG4/LC

Codec Name: Low Complexity

Description: Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal Matroska frame based muxing scheme is applied. AAC audio always uses wFormatTag 0xFF.

Initialization: none

6.4.30. A_AAC/MPEG4/LC/SBR

Codec ID: A_AAC/MPEG4/LC/SBR

Codec Name: Low Complexity with Spectral Band Replication

Description: Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal Matroska frame based muxing scheme is applied. AAC audio always uses wFormatTag 0xFF.

Initialization: none

6.4.31. A_AAC/MPEG4/SSR

Codec ID: A_AAC/MPEG4/SSR

Codec Name: Scalable Sampling Rate

Description: Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal Matroska frame based muxing scheme is applied. AAC audio always uses wFormatTag 0xFF.

Initialization: none

6.4.32. A_AAC/MPEG4/LTP

Codec ID: A_AAC/MPEG4/LTP

Codec Name: Long Term Prediction

Description: Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS

headers and normal Matroska frame based muxing scheme is applied. AAC audio always uses wFormatTag 0xFF.

Initialization: none

6.4.33. A_QUICKTIME

Codec ID: A_QUICKTIME

Codec Name: Audio taken from QuickTime(TM) files

Description: Several codecs as stored in QuickTime, e.g., QDesign Music v1 or v2.

Initialization: The Private Data contains all additional data that is stored in the 'stsd' (sample description) atom in the QuickTime file **after** the mandatory sound descriptor structure (starting with the size and FourCC fields). For an explanation of the QuickTime file format read QuickTime File Format Specification.

6.4.34. A_QUICKTIME/QDMC

Codec ID: A_QUICKTIME/QDMC

Codec Name: QDesign Music

Description:

Initialization: The Private Data contains all additional data that is stored in the 'stsd' (sample description) atom in the QuickTime file **after** the mandatory sound descriptor structure (starting with the size and FourCC fields). For an explanation of the QuickTime file format read QuickTime File Format Specification.

Superseded By: A_QUICKTIME

6.4.35. A_QUICKTIME/QDM2

Codec ID: A_QUICKTIME/QDM2

Codec Name: QDesign Music v2

Description:

Initialization: The Private Data contains all additional data that is stored in the 'stsd' (sample description) atom in the QuickTime file **after** the mandatory sound descriptor structure (starting with the size and FourCC fields). For an explanation of the QuickTime file format read QuickTime File Format Specification.

Superseded By: A_QUICKTIME

6.4.36. A_TTA1

Codec ID: A_TTA1

Codec Name: The True Audio lossless audio compressor

Description: TTA format description Each frame is kept intact, including the CRC32. The header and seektable are dropped. SamplingFrequency, Channels and BitDepth are used in the TrackEntry. wFormatTag = 0x77A1

Initialization: none

6.4.37. A_WAVPACK4

Codec ID: A_WAVPACK4

Codec Name: WavPack lossless audio compressor

Description: The Wavpack packets consist of a stripped header followed by the frame data. For multi-track (> 2 tracks) a frame consists of many packets. For more details, check the WavPack muxing description.

Codec BlockAdditions: For hybrid A_WAVPACK4 encodings (that include a lossy encoding with a supplemental correction to produce a lossless encoding), the correction part is stored in BlockAdditional. The BlockAddId of the BlockMore containing these data **MUST** be 1.

Initialization: none

6.5. Subtitle Codec Mappings

6.5.1. S_TEXT/UTF8

Codec ID: S_TEXT/UTF8

Codec Name: UTF-8 Plain Text

Description: Basic text subtitles. For more information, see Section 7 on Subtitles.

6.5.2. S_TEXT/SSA

Codec ID: S_TEXT/SSA

Codec Name: Subtitles Format

Description: The [Script Info] and [V4 Styles] sections are stored in the codecprivate. Each event is stored in its own Block. For more information, see Section 7.3 on SSA/ASS.

6.5.3. S_TEXT/ASS

Codec ID: S_TEXT/ASS

Codec Name: Advanced Subtitles Format

Description: The [Script Info] and [V4 Styles] sections are stored in the codecprivate. Each event is stored in its own Block. For more information, see Section 7.3 on SSA/ASS.

6.5.4. S_TEXT/WEBVTT

Codec ID: S_TEXT/WEBVTT

Codec Name: Web Video Text Tracks Format (WebVTT)

Description: Advanced text subtitles. For more information, see Section 7.4 on WebVTT.

6.5.5. S_IMAGE/BMP

Codec ID: S_IMAGE/BMP

Codec Name: Bitmap

Description: Basic image based subtitle format; The subtitles are stored as images, like in the DVD. The timestamp in the block header of Matroska indicates the start display time, the duration is set with the Duration element. The full data for the subtitle bitmap is stored in the Block's data section.

6.5.6. S_DVBSUB

Codec ID: S_DVBSUB

Codec Name: Digital Video Broadcasting (DVB) subtitles

Description: This is the graphical subtitle format used in the Digital Video Broadcasting standard. For more information, see Section 7.7 on Digital Video Broadcasting (DVB).

6.5.7. S_VOBSUB

Codec ID: S_VOBSUB

Codec Name: VobSub subtitles

Description: The same subtitle format used on DVDs. Supported is only format version 7 and newer. VobSubs consist of two files, the .idx containing information, and the .sub, containing the actual data. The .idx file is stripped of all empty lines, of all comments and of lines beginning with alt: or langidx:. The line beginning with id: **SHOULD** be transformed into the appropriate Matroska track language element and is discarded. All remaining lines but the ones containing timestamps and file positions are put into the CodecPrivate element.

For each line containing the timestamp and file position data is read from the appropriate position in the .sub file. This data consists of a MPEG program stream which in turn contains SPU packets. The MPEG program stream data is discarded, and each SPU packet is put into one Matroska frame.

6.5.8. S_HDMV/PGS

Codec ID: S_HDMV/PGS

Codec Name: HDMV presentation graphics subtitles (PGS)

Description: This is the graphical subtitle format used on Blu-rays. For more information, see Section 7.6 on HDMV text presentation.

6.5.9. S_HDMV/TEXTST

Codec ID: S_HDMV/TEXTST

Codec Name: HDMV text subtitles

Description: This is the textual subtitle format used on Blu-rays. For more information, see Section 7.5 on HDMV graphics presentation.

6.5.10. S_KATE

Codec ID: S_KATE

Codec Name: Karaoke And Text Encapsulation

Description: A subtitle format developed for ogg. The mapping for Matroska is described on the Xiph wiki. As for Theora and Vorbis, Kate headers are stored in the private data as xiph-laced packets.

6.6. Button Codec Mappings

6.6.1. B_VOBBTN

Codec ID: B_VOBBTN

Codec Name: VobBtn Buttons

Description: Based on MPEG/VOB PCI packets. The file contains a header consisting of the string "butonDVD" followed by the width and height in pixels (16 bits integer each) and 4 reserved bytes. The rest is full PCI packets.

6.7. Block Addition Mappings

Registered BlockAddIDType are:

6.7.1. Use BlockAddIDValue

Block type identifier: 0

Block type name: Use BlockAddIDValue

Description: This value indicates that the actual type is stored in BlockAddIDValue instead. This value is expected to be used when it is important to have a strong compatibility with players or derived formats not supporting BlockAdditionMapping but using BlockAdditions with an unknown BlockAddIDValue, and **SHOULD NOT** be used if it is possible to use another value.

6.7.2. Opaque data

Block type identifier: 1

Block type name: Opaque data

Description: the BlockAdditional data is interpreted as opaque additional data passed to the codec with the Block data. BlockAddIDValue **MUST** be 1.

6.7.3. ITU T.35 metadata

Block type identifier: 4

Block type name: ITU T.35 metadata

Description: the BlockAdditional data is interpreted as ITU T.35 metadata, as defined by ITU-T T.35 terminal codes. BlockAddIDValue **MUST** be 4.

6.7.4. avcE

Block type identifier: 0x61766345

Block type name: Dolby Vision enhancement-layer AVC configuration

Description: the BlockAddIDExtraData data is interpreted as the Dolby Vision enhancement-layer AVC configuration box as described in [DolbyVisionWithinIso]. This extension **MUST NOT** be used if Codec ID is not V_MPEG4/ISO/AVC.

6.7.5. dvcC

Block type identifier: 0x64766343

Block type name: Dolby Vision configuration

Description: the BlockAddIDExtraData data is interpreted as DOVIDecoderConfigurationRecord structure, as defined in [DolbyVisionWithinIso], for Dolby Vision profiles less than and equal to 7.

6.7.6. dvvC

Block type identifier: 0x664767643

Block type name: Dolby Vision configuration

Description: the BlockAddIDExtraData data is interpreted as DOVIDecoderConfigurationRecord structure, as defined in [DolbyVisionWithinIso], for Dolby Vision profiles greater than 7.

6.7.7. hvcE

Block type identifier: 0x68766345

Block type name: Dolby Vision enhancement-layer HEVC configuration

Description: the BlockAddIDExtraData data is interpreted as the Dolby Vision enhancement-layer HEVC configuration as described in [DolbyVisionWithinIso]. This extension **MUST NOT** be used if Codec ID is not V_MPEGH/ISO/HEVC.

6.7.8. mvcC

Block type identifier: 0x6D766343

Block type name: MVC configuration

Description: the BlockAddIDExtraData data is interpreted as MVCDecoderConfigurationRecord structure, as defined in [ISO.14496-15].

6.8. This extension MUST NOT be used if Codec ID is not V_MPEG4/ISO/AVC.

6.9. title: Subtitles

7. Subtitles

Because Matroska is a general container format, we try to avoid specifying the formats to store in it. This type of work is really outside of the scope of a container-only format. However, because the use of subtitles in A/V containers has been so limited (with the exception of DVD) we are taking the time to specify how to store some of the more common subtitle formats in Matroska. This is being done to help facilitate their growth. Otherwise, incompatibilities could prevent the standardization and use of subtitle storage.

This page is not meant to be a complete listing of all subtitle formats that will be used in Matroska, it is only meant to be a guide for the more common, current formats. It is possible that we will add future formats to this page as they are created, but it is not likely as any other new subtitle format designer would likely have their own specifications. Any specification listed here **SHOULD** be strictly adhered to or it **SHOULD NOT** use the corresponding Codec ID.

Here is a list of pointers for storing subtitles in Matroska:

- Any Matroska file containing only subtitles **SHOULD** use the extension ".mks".
- As a general rule of thumb for all codecs, information that is global to an entire stream **SHOULD** be stored in the CodecPrivate element.
- Start and stop timestamps that are used in a timestamps native storage format SHOULD be removed when being placed in Matroska as they could interfere if the file is edited afterwards. Instead, the Blocks timestamp and Duration **SHOULD** be used to say when the timestamp is displayed.
- Because a "subtitle" stream is actually just an overlay stream, anything with a transparency layer could be use, including video.

7.1. Images Subtitles

The first image format that is a goal to import into Matroska is the VobSub subtitle format. This subtitle type is generated by exporting the subtitles from a DVD.

The requirement for muxing VobSub into Matroska is v7 subtitles (see first line of the .IDX file). If the version is smaller, you must remux them using the SubResync utility from VobSub 2.23 (or MPC) into v7 format. Generally any newly created subs will be in v7 format.

The .IFO file will not be used at all.

If there is more than one subtitle stream in the VobSub set, each stream will need to be separated into separate tracks for storage in Matroska. E.g. the VobSub file contains streams for both English and German subtitles. Then the resulting Matroska file SHOULD contain two tracks. That way the language information can be dropped and mapped to Matroska's language tags.

The .IDX file is reformatted (see below) and placed in the CodecPrivate.

Each .BMP will be stored in its own Block. The Timestamp with be stored in the Blocks Timestamp and the duration will be stored in the Default Duration.

Here is an example .IDX file:

```
# VobSub index file, v7 (do not modify this line!)
#
# To repair desynchronization, you can insert gaps this way:
# (it usually happens after vob id changes)
#
# delay: [sign]hh:mm:ss:ms
#
# Where:
# [sign]: +, - (optional)
# hh: hours (0 <= hh)
# mm/ss: minutes/seconds (0 <= mm/ss <= 59)
# ms: milliseconds (0 <= ms <= 999)
#
# Note: You can't position a sub before the previous with a negative
# value.
#
# You can also modify timestamps or delete a few subs you don't
# like. Just make sure they stay in increasing order.

# Settings

# Original frame size
size: 720x480

# Origin, relative to the upper-left corner, can be overloaded by
# alignment
org: 0, 0

# Image scaling (hor,ver), origin is at the upper-left corner or at
# the alignment coord (x, y)
scale: 100%, 100%

# Alpha blending
alpha: 100%

# Smoothing for very blocky images (use OLD for no filtering)
smooth: OFF

# In millisecs
fadein/out: 50, 50

# Force subtitle placement relative to (org.x, org.y)
align: OFF at LEFT TOP

# For correcting non-progressive desync. (in millisecs or
# hh:mm:ss:ms)
# Note: Not effective in DirectVobSub, use "delay: ... " instead.
time offset: 0

# ON: displays only forced subtitles, OFF: shows everything
```

```
forced subs: OFF

# The original palette of the DVD
palette: 000000, 7e7e7e, fbff8b, cb86f1, 7f74b8, e23f06, 0a48ea, \
b3d65a, 6b92f1, 87f087, c02081, f8d0f4, e3c411, 382201, e8840b, \
fdfdfd

# Custom colors (transp idxs and the four colors)
custom colors: OFF, tridx: 0000, colors: 000000, 000000, 000000, \
000000

# Language index in use
langidx: 0

# English
id: en, index: 0
# Uncomment next line to activate alternative name in DirectVobSub /
# Windows Media Player 6.x
# alt: English
# Vob/Cell ID: 1, 1 (PTS: 0)
timestamp: 00:00:01:101, filepos: 000000000
timestamp: 00:00:08:708, filepos: 000001000
```

First, lines beginning with "#" are removed. These are comments to make text file editing easier, and as this is not a text file, they aren't needed.

Next remove the "langidx" and "id" lines. These are used to differentiate the subtitle streams and define the language. As the streams will be stored separately anyway, there is no need to differentiate them here. Also, the language setting will be stored in the Matroska tags, so there is no need to store it here.

Finally, the "timestamp" will be used to set the Block's timestamp. Once it is set there, there is no need for it to be stored here. Also, as it may interfere if the file is edited, it **SHOULD NOT** be stored here.

Once all of these items are removed, the data to store in the CodecPrivate **SHOULD** look like this:

```
size: 720x480
org: 0, 0
scale: 100%, 100%
alpha: 100%
smooth: OFF
fadein/out: 50, 50
align: OFF at LEFT TOP
time offset: 0
forced subs: OFF
palette: 000000, 7e7e7e, fbff8b, cb86f1, 7f74b8, e23f06, 0a48ea, \
b3d65a, 6b92f1, 87f087, c02081, f8d0f4, e3c411, 382201, e8840b, \
fdfdfd
custom colors: OFF, tridx: 0000, colors: 000000, 000000, 000000, \
000000
```

There **SHOULD** also be two Blocks containing one image each with the timestamps "00:00:01:101" and "00:00:08:708".

7.2. SRT Subtitles

SRT is perhaps the most basic of all subtitle formats.

It consists of four parts, all in text:

1. A number indicating which subtitle it is in the sequence. 2. The time that the subtitle appears on the screen, and then disappears. 3. The subtitle itself. 4. A blank line indicating the start of a new subtitle.

When placing SRT in Matroska, part 3 is converted to UTF-8 (S_TEXT/UTF8) and placed in the data portion of the Block. Part 2 is used to set the timestamp of the Block, and BlockDuration element. Nothing else is used.

Here is an example SRT file:

```
1
00:02:17,440 --> 00:02:20,375
Senator, we're making
our final approach into Coruscant.

2
00:02:20,476 --> 00:02:22,501
Very good, Lieutenant.
```

In this example, the text "Senator, we're making our final approach into Coruscant." would be converted into UTF-8 and placed in the Block. The timestamp of the block would be set to "00:02:17,440". And the BlockDuration element would be set to "00:00:02,935".

The same is repeated for the next subtitle.

Because there are no general settings for SRT, the CodecPrivate is left blank.

7.3. SSA/ASS Subtitles

SSA stands for Sub Station Alpha. It's the file format used by the popular subtitle editor, SubStation Alpha. This format is widely used by fansubbers.

It allows you to do some advanced display features, like positioning, karaoke, style managements...

For detailed information on SSA/ASS, see the SSA specs. It includes an SSA specs description and the advanced features added by ASS format (standing for Advanced SSA). Because SSA and ASS are so similar, they are treated the same here.

Like SRT, this format is text based with a particular syntax.

A file consists of 4 or 5 parts, declared ala INI file (but it's not an INI !)

The first, "[Script Info]" contains some information about the subtitle file, such as it's title, who created it, type of script and a very important one: "PlayResY". Be careful of this value, everything in your script (font size, positioning) is scaled by it. Sub Station Alpha uses your desktops Y resolution to write this value, so if a friend with a large monitor and a high screen resolution gives you an edited script, you can mess everything up by saving the script in SSA with your low-cost monitor.

The second, "[V4 Styles]", is a list of style definitions. A style describe how will look a text on the screen. It defines font, font size, primary/.../outile colour, position, alignment, etc.

For example this:

```
Format: Name, Fontname, Fontsize, PrimaryColour, SecondaryColour, \
TertiaryColour, BackColour, Bold, Italic, BorderStyle, Outline, \
Shadow, Alignment, MarginL, MarginR, MarginV, AlphaLevel, Encoding
Style: Wolf main,Wolf_Rain,56,15724527,15724527,15724527,4144959,0,\
0,1,1,2,2,5,5,30,0,0
```

The third, "[Events]", is the list of text you want to display at the right timing. You can specify some attribute here. Like the style to use for this event (**MUST**be defined in the list), the position of the text (Left, Right, Vertical Margin), an effect. Name is mostly used by translator to know who said this sentence. Timing is in h:mm:ss.cc (centisec).

```
Format: Marked, Start, End, Style, Name, MarginL, MarginR, MarginV, \
Effect, Text
Dialogue: Marked=0,0:02:40.65,0:02:41.79,Wolf main,Cher,0000,0000,\
0000,,Et les enregistrements de ses ondes delta ?
Dialogue: Marked=0,0:02:42.42,0:02:44.15,Wolf main,autre,0000,0000,\
0000,,Toujours rien.
```

"[Pictures]" or "[Fonts]" part can be found in some SSA file, they contains UUE-encoded pictures/font but those features are only used by Sub Station Alpha -- i.e. no filter (Vobsub/Avery Lee Subtiler filter) use them.

Now, how are they stored in Matroska?

*All text is converted to UTF-8

*All the headers are stored in CodecPrivate (Script Info and the Styles list)

*Start & End field are used to set TimeStamp and the BlockDuration element. the data stored is:

*Events are stored in the Block in this order: ReadOrder, Layer, Style, Name, MarginL, MarginR, MarginV, Effect, Text (Layer comes from ASS specs ... it's empty for SSA.) "ReadOrder field is needed for the decoder to be able to reorder the streamed samples as they were placed originally in the file."

Here is an example of an SSA file.

```
[Script Info]
; This is a Sub Station Alpha v4 script.
; For Sub Station Alpha info and downloads,
; go to \
; [http://www.eswat.demon.co.uk/](http://www.eswat.demon.co.uk/)
; or email \
; [kotus@eswat.demon.co.uk](mailto:kotus@eswat.demon.co.uk)
Title: Wolf's rain 2
Original Script: Anime-spirit Ishin-francais
Original Translation: Coolman
Original Editing: Spikewolfwood
Original Timing: Lord_alucard
Original Script Checking: Spikewolfwood
ScriptType: v4.00
Collisions: Normal
PlayResY: 1024
PlayDepth: 0
Wav: 0, 128697,D:\Alex\Anime\- Fansub -\- TAFF -\WR_-_02_Wav.wav
Wav: 0, 120692,H:\team truc\WR_-_02.wav
Wav: 0, 116504,E:\sub\wolf's_rain\WOLF'S RAIN 02.wav
LastWav: 3
Timer: 100,0000

[V4 Styles]
Format: Name, Fontname, Fontsize, PrimaryColour, SecondaryColour, \
TertiaryColour, BackColour, Bold, Italic, BorderStyle, Outline, \
Shadow, Alignment, MarginL, MarginR, MarginV, AlphaLevel, Encoding
Style: Default,Arial,20,65535,65535,65535,-2147483640,-1,0,1,3,0,2,\
30,30,30,0,0
Style: Titre_episode,Akbar,140,15724527,65535,65535,986895,-1,0,1,1,\
0,3,30,30,30,0,0
Style: Wolf main,Wolf_Rain,56,15724527,15724527,15724527,4144959,0,\
0,1,1,2,2,5,5,30,0,0

[Events]
Format: Marked, Start, End, Style, Name, MarginL, MarginR, MarginV, \
Effect, Text
Dialogue: Marked=0,0:02:40.65,0:02:41.79,Wolf main,Cher,0000,0000,\
0000,,Et les enregistrements de ses ondes delta ?
Dialogue: Marked=0,0:02:42.42,0:02:44.15,Wolf main,autre,0000,0000,\
0000,,Toujours rien.
```

Here is what would be placed into the CodecPrivate element.

```
[Script Info]
; This is a Sub Station Alpha v4 script.
; For Sub Station Alpha info and downloads,
; go to \
; [http://www.eswat.demon.co.uk/](http://www.eswat.demon.co.uk/)
; or email \
; [kotus@eswat.demon.co.uk](mailto:kotus@eswat.demon.co.uk)
Title: Wolf's rain 2
Original Script: Anime-spirit Ishin-francais
Original Translation: Coolman
Original Editing: Spikewolfwood
Original Timing: Lord_alucard
Original Script Checking: Spikewolfwood
ScriptType: v4.00
Collisions: Normal
PlayResY: 1024
PlayDepth: 0
Wav: 0, 128697,D:\Alex\Anime\- Fansub -\- TAFF -\WR_-_02_Wav.wav
Wav: 0, 120692,H:\team truc\WR_-_02.wav
Wav: 0, 116504,E:\sub\wolf's_rain\WOLF'S RAIN 02.wav
LastWav: 3
Timer: 100,0000

[V4 Styles]
Format: Name, Fontname, Fontsize, PrimaryColour, SecondaryColour, \
TertiaryColour, BackColour, Bold, Italic, BorderStyle, Outline, \
Shadow, Alignment, MarginL, MarginR, MarginV, AlphaLevel, Encoding
Style: Default,Arial,20,65535,65535,65535,-2147483640,-1,0,1,3,0,2,\
30,30,30,0,0
Style: Titre_episode,Akbar,140,15724527,65535,65535,986895,-1,0,1,1,\
0,3,30,30,30,0,0
Style: Wolf main,Wolf_Rain,56,15724527,15724527,15724527,4144959,0,\
0,1,1,2,2,5,5,30,0,0
```

And here are the two blocks that would be generated.

Block's timestamp: 00:02:40.650 BlockDuration: 00:00:01.140

```
1,,Wolf main,Cher,0000,0000,0000,,Et les enregistrements de ses \
ondes delta ?
```

Block's timestamp: 00:02:42.420 BlockDuration: 00:00:01.730

```
2,,Wolf main,autre,0000,0000,0000,,Toujours rien.
```

7.4. WebVTT

The "Web Video Text Tracks Format" (short: WebVTT) is developed by the World Wide Web Consortium (W3C). Its specifications are freely available.

The guiding principles for the storage of WebVTT in Matroska are:

- Consistency: store data in a similar way to other subtitle codecs
- Simplicity: making decoding and remuxing as easy as possible for existing infrastructures
- Completeness: keeping as much data as possible from the original WebVTT file

7.4.1. Storage of WebVTT in Matroska

7.4.1.1. CodecID: codec identification

The CodecID to use is S_TEXT/WEBVTT.

7.4.1.2. CodecPrivate: storage of global WebVTT blocks

This element contains all global blocks before the first subtitle entry. This starts at the "WEBVTT" file identification marker but excludes the optional byte order mark.

7.4.1.3. Storage of non-global WebVTT blocks

Non-global WebVTT blocks (e.g., "NOTE") before a WebVTT Cue Text are stored in Matroska's BlockAddition element together with the Matroska Block containing the WebVTT Cue Text these blocks precede (see below for the actual format).

7.4.1.4. Storage of Cues in Matroska blocks

Each WebVTT Cue Text is stored directly in the Matroska Block.

A muxer **MUST** change all WebVTT Cue Timestamps present within the Cue Text to be relative to the Matroska Block's timestamp.

The Cue's start timestamp is used as the Matroska Block's timestamp.

The difference between the Cue's end timestamp and its start timestamp is used as the Matroska Block's duration.

7.4.1.5. BlockAdditions: storing non-global WebVTT blocks, Cue Settings Lists and Cue identifiers

Each Matroska Block may be accompanied by one BlockAdditions element. Its format is as follows:

1. The first line contains the WebVTT Cue Text's optional Cue Settings List followed by one line feed character (U+0x000a). The Cue Settings List may be empty, in which case the line consists of the line feed character only.

2. The second line contains the WebVTT Cue Text's optional Cue Identifier followed by one line feed character (U+0x000a). The line may be empty indicating that there was no Cue Identifier in the source file, in which case the line consists of the line feed character only.

3. The third and all following lines contain all WebVTT Comment Blocks that precede the current WebVTT Cue Block. These may be absent.

If there is no Matroska BlockAddition element stored together with the Matroska Block, then all three components (Cue Settings List, Cue Identifier, Cue Comments) **MUST** be assumed to be absent.

7.4.2. Examples of transformation

Here's an example how a WebVTT is transformed.

7.4.2.1. Example WebVTT file

Let's take the following example file:

```
WEBVTT with text after the signature

STYLE
::cue {
  background-image: linear-gradient(to bottom, dimgray, lightgray);
  color: papayawhip;
}
/* Style blocks cannot use blank lines nor "dash dash greater \
than" */

NOTE comment blocks can be used between style blocks.

STYLE
::cue(b) {
  color: peachpuff;
}

REGION
id:bill
width:40%
lines:3
regionanchor:0%,100%
viewportanchor:10%,90%
scroll:up

NOTE
Notes always span a whole block and can cover multiple
lines. Like this one.
An empty line ends the block.

hello
00:00:00.000 --> 00:00:10.000
Example entry 1: Hello <b>world</b>.

NOTE style blocks cannot appear after the first cue.

00:00:25.000 --> 00:00:35.000
Example entry 2: Another entry.
This one has multiple lines.

00:01:03.000 --> 00:01:06.500 position:90% align:right size:35%
Example entry 3: That stuff to the right of the timestamps are cue \
settings.

00:03:10.000 --> 00:03:20.000
Example entry 4: Entries can even include timestamps.
For example:<00:03:15.000>This becomes visible five seconds
after the first part.
```

7.4.2.2. Example of CodecPrivate

The resulting CodecPrivate element will look like this:

```
WEBVTT with text after the signature

STYLE
::cue {
  background-image: linear-gradient(to bottom, dimgray, lightgray);
  color: papayawhip;
}
/* Style blocks cannot use blank lines nor "dash dash greater \
than" */

NOTE comment blocks can be used between style blocks.

STYLE
::cue(b) {
  color: peachpuff;
}

REGION
id:bill
width:40%
lines:3
regionanchor:0%,100%
viewportanchor:10%,90%
scroll:up

NOTE
Notes always span a whole block and can cover multiple
lines. Like this one.
An empty line ends the block.
```

7.4.2.3. Storage of Cue 1

Example Cue 1: timestamp 00:00:00.000, duration 00:00:10.000,
Block's content:

```
Example entry 1: Hello <b>world</b>.
```

BlockAddition's content starts with one empty line as there's no Cue
Settings List:

```

hello
```

7.4.2.4. Storage of Cue 2

Example Cue 2: timestamp 00:00:25.000, duration 00:00:10.000, Block's content:

```
Example entry 2: Another entry.
This one has multiple lines.
```

BlockAddition's content starts with two empty lines as there's neither a Cue Settings List nor a Cue Identifier:

```
NOTE style blocks cannot appear after the first cue.
```

7.4.2.5. Storage of Cue 3

Example Cue 3: timestamp 00:01:03.000, duration 00:00:03.500, Block's content:

```
Example entry 3: That stuff to the right of the timestamps are cue \
settings.
```

BlockAddition's content ends with an empty line as there's no Cue Identifier and there were no WebVTT Comment blocks:

```
position:90% align:right size:35%
```

7.4.2.6. Storage of Cue 4

Example Cue 4: timestamp 00:03:10.000, duration 00:00:10.000, Block's content:

Example entry 4: Entries can even include timestamps. For example: 00:00:05.000This becomes visible five seconds after the first part.

This Block does not need a BlockAddition as the Cue did not contain an Identifier, nor a Settings List, and it wasn't preceded by Comment blocks.

7.4.3. Storage of WebVTT in Matroska vs. WebM

Note: the storage of WebVTT in Matroska is not the same as the design document for storage of WebVTT in WebM. There are several reasons for this including but not limited to: the WebM document is old (from February 2012) and was based on an earlier draft of WebVTT and ignores several parts that were added to WebVTT later; WebM does still not support subtitles at all; the proposal suggests splitting the information across multiple tracks making demuxer's and remuxer's life very difficult.

7.5. HDMV presentation graphics subtitles

The specifications for the HDMV presentation graphics subtitle format (short: HDMV PGS) can be found in the document "Blu-ray Disc Read-Only Format; Part 3 - Audio Visual Basic Specifications" in section 9.14 "HDMV graphics streams".

7.5.1. Storage of HDMV presentation graphics subtitles

The CodecID to use is S_HDMV/PGS. A CodecPrivate element is not used.

7.5.1.1. Storage of HDMV PGS Segments in Matroska Blocks

Each HDMV PGS Segment (short: Segment) will be stored in a Matroska Block. A Segment is the data structure described in section 9.14.2.1 "Segment coding structure and parameters" of the Blu-ray specifications.

Each Segment contains a presentation timestamp. This timestamp will be used as the timestamp for the Matroska Block.

A Segment is normally shown until a subsequent Segment is encountered. Therefore the Matroska Block **MAY** have no Duration. In that case, a player **MUST** display a Segment within a Matroska Block until the next Segment is encountered.

A muxer **MAY** use a Duration, e.g., by calculating the distance between two subsequent Segments. If a Matroska Block has a Duration, a player **MUST** display that Segment only for the duration of the Block's Duration.

7.6. HDMV text subtitles

The specifications for the HDMV text subtitle format (short: HDMV TextST) can be found in the document "Blu-ray Disc Read-Only Format; Part 3 - Audio Visual Basic Specifications" in section 9.15 "HDMV text subtitle streams".

7.6.1. Storage of HDMV text subtitles

The CodecID to use is S_HDMV/TEXTST.

A CodecPrivate Element is required. It **MUST** contain the stream's Dialog Style Segment as described in section 9.15.4.2 "Dialog Style Segment" of the Blu-ray specifications.

7.6.1.1. Storage of HDMV TextST Dialog Presentation Segments in Matroska Blocks

Each HDMV Dialog Presentation Segment (short: Segment) will be stored in a Matroska Block. A Segment is the data structure described in section 9.15.4.3 "Dialog presentation segment" of the Blu-ray specifications.

Each Segment contains a start and an end presentation timestamp (short: start PTS & end PTS). The start PTS will be used as the timestamp for the Matroska Block. The Matroska Block **MUST** have a Duration, and that Duration is the difference between the end PTS and the start PTS.

A player **MUST** use the Matroska Block's timestamp and Duration instead of the Segment's start and end PTS for determining when and how long to show the Segment.

7.6.1.2. Character set

When TextST subtitles are stored inside Matroska, the only allowed character set is UTF-8.

Each HDMV text subtitle stream in a Blu-ray can use one of a handful of character sets. This information is not stored in the MPEG2 Transport Stream itself but in the accompanying Clip Information file.

Therefore a muxer **MUST** parse the accompanying Clip Information file. If the information indicates a character set other than UTF-8, it **MUST** re-encode all text Dialog Presentation Segments from the indicated character set to UTF-8 prior to storing them in Matroska.

7.7. Digital Video Broadcasting (DVB) subtitles

The specifications for the Digital Video Broadcasting subtitle bitstream format (short: DVB subtitles) can be found in the document "ETSI EN 300 743 - Digital Video Broadcasting (DVB); Subtitling systems". The storage of DVB subtitles in MPEG transport streams is specified in the document "ETSI EN 300 468 - Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

7.7.1. Storage of DVB subtitles

7.7.1.1. CodecID

The CodecID to use is S_DVBSUB.

7.7.1.2. CodecPrivate

The CodecPrivate element is five bytes long and has the following structure:

*2 bytes: composition page ID (bit string, left bit first)

*2 bytes: ancillary page ID (bit string, left bit first)

*1 byte: subtitling type (bit string, left bit first)

The semantics of these bytes are the same as the ones described in section 6.2.41 "Subtitling descriptor" of ETSI EN 300 468.

7.7.1.3. Storage of DVB subtitles in Matroska Blocks

Each Matroska Block consists of one or more DVB Subtitle Segments as described in segment 7.2 "Syntax and semantics of the subtitling segment" of ETSI EN 300 743.

Each Matroska Block **SHOULD** have a Duration indicating how long the DVB Subtitle Segments

7.8. in that Block SHOULD be displayed.

7.9. title: Block Additional Mapping

8. Block Additional Mapping

Extra data or metadata can be added to each Block using BlockAdditional data. Each BlockAdditional contains a BlockAddID that identifies the kind of data it contains. When the BlockAddID is set to "1" the contents of the BlockAdditional Element are define by the Codec Mappings defines; see Section 6.1.5. When the BlockAddID is set a value greater than "1", then the contents of the BlockAdditional Element are defined by the BlockAdditionalMapping Element, within the associated Track Element, where the BlockAddID Element of BlockAdditional Element equals the BlockAddIDValue of the associated Track's BlockAdditionalMapping Element. That BlockAdditionalMapping Element identifies a particular Block Additional Mapping by the BlockAddIDType.

The following XML depicts a use of a Block Additional Mapping to associate a timecode value with a Block:

```
<Segment>
  <!--Mandatory elements ommitted for readability-->
  <Tracks>
    <TrackEntry>
      <TrackNumber>1</TrackNumber>
      <TrackUID>568001708</TrackUID>
      <TrackType>1</TrackType>
      <BlockAdditionalMapping>
        <BlockAddIDValue>2</BlockAddIDValue><!--arbitrary value
          used in BlockAddID-->
        <BlockAddIDName>timecode</BlockAddIDName>
        <BlockAddIDType>12</BlockAddIDType>
      </BlockAdditionalMapping>
      <CodecID>V_FFV1</CodecID>
      <Video>
        <PixelWidth>1920</PixelWidth>
        <PixelHeight>1080</PixelHeight>
      </Video>
    </TrackEntry>
  </Tracks>
  <Cluster>
    <Timestamp>3000</Timestamp>
    <BlockGroup>
      <Block>{binary video frame}</Block>
      <BlockAdditions>
        <BlockMore>
          <BlockAddID>2</BlockAddID><!--arbitrary value from
            BlockAdditionalMapping-->
          <BlockAdditional>01:00:00:00</BlockAdditional>
        </BlockMore>
      </BlockAdditions>
    </BlockGroup>
  </Cluster>
</Segment>
```

Block Additional Mappings detail how additional data **MAY** be stored in the BlockMore Element with a BlockAdditionMapping Element, within the Track Element, which identifies the BlockAdditional content. Block Additional Mappings define the BlockAddIDType value reserved to identify that type of data as well as providing an optional label stored within the BlockAddIDName Element. When the Block Additional Mapping is dependent on additional contextual information, then the Mapping **SHOULD** describe how such additional contextual information is stored within the BlockAddIDExtraData Element.

The following Block Additional Mappings are defined.

8.1. Summary of Assigned BlockAddIDType Values

For convenience, the following table shows the assigned BlockAddIDType values along with the BlockAddIDName and Citation.

BlockAddIDType	BlockAddIDName	Citation
121	SMPTE ST 12-1 timecode	Section 8.2

Table 4

8.2. SMPTE ST 12-1 Timecode

8.2.1. Timecode Description

SMPTE ST 12-1 timecode values can be stored in the BlockMore Element to associate the content of a Matroska Block with a particular timecode value. If the Block uses Lacing, the timecode value is associated with the first frame of the Lace.

The Block Additional Mapping contains a full binary representation of a 64 bit SMPTE timecode value stored in big-endian format and expressed exactly as defined in Section 8 and 9 of SMPTE 12M [ST12]. For convenience, here are the bit assignments for a SMPTE ST 12-1 binary representation as described in Section 6.2 of [RFC5484]:

Bit Positions	Label
0--3	Units of frames
4--7	First binary group
8--9	Tens of frames
10	Drop frame flag
11	Color frame flag
12--15	Second binary group
16--19	Units of seconds
20--23	Third binary group
24--26	Tens of seconds
27	Polarity correction
28--31	Fourth binary group
32--35	Units of minutes
36--39	Fifth binary group
40--42	Tens of minutes
43	Binary group flag BGF0
44--47	Sixth binary group
48--51	Units of hours
52--55	Seventh binary group
56--57	Tens of hours
58	Binary group flag BGF1
59	Binary group flag BGF2
60--63	Eighth binary group

Table 5

For example, a timecode value of "07:32:54;18" can be expressed as a 64 bit SMPTE 12M value as:

```
10000000 01100000 01100000 01010000
00100000 00110000 01110000 00000000
```

8.2.2. BlockAddIDType

The BlockAddIDType value reserved for timecode is "121".

8.2.3. BlockAddIDName

The BlockAddIDName value reserved for timecode is "SMPTE ST 12-1 timecode".

8.2.4. BlockAddIDExtraData

BlockAddIDExtraData is unused within this block additional mapping.

9. Normative References

[DolbyVisionWithinIso] Dolby, "Dolby Vision Streams Within the ISO Base MediaFile Format", 7 February 2020, <https://www.dolby.com/us/en/technologies/dolby-vision/dolby-vision-bitstreams-within-the-iso-base-media-file-format-v2.1.2.pdf>.

[IEEE.1857-4] IEEE, "IEEE Standard for Second-Generation IEEE 1857 Video Coding", 23 October 2018, <https://standards.ieee.org/standard/1857_4-2018.html>.

[IEEE.754] IEEE, "IEEE Standard for Binary Floating-Point Arithmetic", 13 June 2019, <https://standards.ieee.org/standard/754-2019.html>.

[ISO.14496-15] International Organization for Standardization, "Information technology — Coding of audio-visual objects — Part 15: Carriage of network abstraction layer (NAL) unit structured video in ISO base media file format", ISO Standard 14496, 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC6386] Bankoski, J., Koleszar, J., Quillio, L., Salonen, J., Wilkins, P., and Y. Xu, "VP8 Data Format and Decoding Guide", RFC 6386, DOI 10.17487/RFC6386, November 2011, <https://www.rfc-editor.org/info/rfc6386>.

[RFC6648] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", BCP 178, RFC 6648, DOI 10.17487/RFC6648, June 2012, <https://www.rfc-editor.org/info/rfc6648>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[ST12] SMPTE, "Time and Control Code", ST ST 12-1:2014, DOI 10.5594/SMPTE.ST12-1.2014, 20 February 2014, <http://ieeexplore.ieee.org/document/7291029/>.

10. Informative References

[RFC5484] Singer, D., "Associating Time-Codes with RTP Streams", RFC 5484, DOI 10.17487/RFC5484, March 2009, <https://www.rfc-editor.org/info/rfc5484>.

Authors' Addresses

Steve Lhomme

Email: slhomme@matroska.org

Moritz Bunkus

Email: moritz@bunkus.org

Dave Rice

Email: dave@dericed.com