Workgroup: cellar
Internet-Draft: draft-ietf-cellar-matroska-11
Published: 3 July 2022
Intended Status: Standards Track
Expires: 4 January 2023
Authors: S. Lhomme M. Bunkus D. Rice

Matroska Media Container Format Specifications

Abstract

This document defines the Matroska audiovisual container, including definitions of its structural elements, as well as its terminology, vocabulary, and application.

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 4 January 2023.

Table of Contents

1. Introduction

Matroska is a multimedia container format. It was derived from a project called [MCF], but differentiates from it significantly because it is based on EBML (Extensible Binary Meta Language) [RFC8794], a binary derivative of XML. EBML enables significant advantages in terms of future format extensibility, without breaking file support in old parsers.

First, it is essential to clarify exactly "What an Audio/Video container is", to avoid any misunderstandings:

- It is NOT a video or audio compression format (codec)
- It is an envelope for which there can be many audio, video, and subtitles streams, allowing the user to store a complete movie or CD in a single file.

Matroska is designed with the future in mind. It incorporates features like:

- Fast seeking in the file
- Chapter entries
- Full metadata (tags) support
- Selectable subtitle/audio/video streams
- Modularly expandable
- Error resilience (can recover playback even when the stream is damaged)
- Streamable over the internet and local networks (HTTP, CIFS, FTP, etc)
- Menus (like DVDs have [DVD-Video])

Matroska is an open standards project published as an IETF Standard Track RFC. As per the terms of BCP 78 [RFC5378], the technical specifications describing the bitstream are open to everybody. This specification falls under the terms of BCP 79 [RFC8179] with respect to IPR claims. While there are patent claims associated with some codecs that can be contained within a Matroska container, the container itself is free of all such claims.

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. But since it's quite complete it is used as a reference for the development of libmatroska.

This document covers Matroska versions 1, 2, 3 and 4. Matroska v4 is the current version. Matroska 1 to 3 are no longer maintained. No new elements are expected in files with version numbers 1, 2, or 3.

3. 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.

This document defines specific terms in order to define the format and application of Matroska. Specific terms are defined below:

Matroska: A multimedia container format based on EBML (Extensible Binary Meta Language).

Matroska Reader: A data parser that interprets the semantics of a Matroska document and creates a way for programs to use Matroska.

Matroska Player: A Matroska Reader with a primary purpose of playing audiovisual files, including Matroska documents.

4. Matroska Overview

4.1. Principles

Matroska is a Document Type of EBML (Extensible Binary Meta Language). This specification is dependent on the EBML Specification [RFC8794]. For an understanding of Matroska's EBML Schema, see in particular the sections of the EBML Specification covering EBML Element Types (Section 7), EBML Schema (Section 11.1), and EBML Structure (Section 3).

4.2. Added EBML Constraints

As an EBML Document Type, Matroska adds the following constraints to the EBML specification.

*The docType of the EBML Header **MUST** be "matroska".

*The EBMLMaxIDLength of the EBML Header **MUST** be "4".

*The EBMLMaxSizeLength of the EBML Header **MUST** be between "1" and "8" inclusive.

4.3. Design Rules

The Root Element and all Top-Levels Elements use 4 octets for their EBML Element ID -- i.e. Segment and direct children of Segment.

Legacy EBML/Matroska parsers did not handle Empty Elements properly, elements present in the file but with a length of zero. They always

assumed the value was 0 for integers/dates and 0x0p+0 for floats, no matter the default value of the element which should have been used instead. Therefore Matroska writers **MUST NOT** use EBML Empty Elements, if the element has a default value that is not 0 for integers/dates and 0x0p+0 for floats.

When adding new elements to Matroska, these rules **MUST** be followed:

- A non-mandatory integer/date Element **MUST NOT** have a default value other than 0.
- A non-mandatory float Element **MUST NOT** have a default value other than 0x0p+0.
- A non-mandatory string Element **MUST NOT** have a default value, as empty string cannot be defined in the XML Schema.

4.4. Data Layout

A Matroska file **MUST** be composed of at least one EBML Document using the Matroska Document Type. Each EBML Document **MUST** start with an EBML Header and **MUST** be followed by the EBML Root Element, defined as Segment in Matroska. Matroska defines several Top Level Elements which **MAY** occur within the Segment.

As an example, a simple Matroska file consisting of a single EBML Document could be represented like this:

- EBML Header
- Segment

A more complex Matroska file consisting of an EBML Stream (consisting of two EBML Documents) could be represented like this:

- EBML Header
- Segment
- EBML Header
- Segment

The following diagram represents a simple Matroska file, comprised of an EBML Document with an EBML Header, a Segment Element (the Root Element), and all eight Matroska Top Level Elements. In the following diagrams of this section, horizontal spacing expresses a parent-child relationship between Matroska Elements (e.g., the Info Element is contained within the Segment Element) whereas vertical alignment represents the storage order within the file.

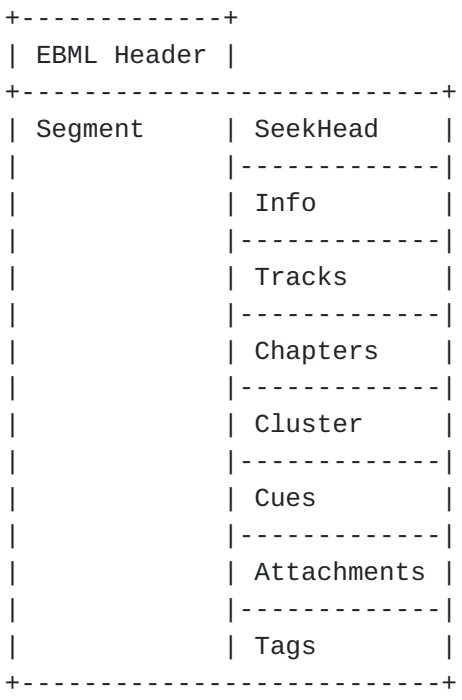

Figure 1: Basic layout of a Matroska file.

The Matroska EBML Schema defines eight Top Level Elements:

- *SeekHead [Section 6.3],
- *Info [Section 6.5],
- *Tracks [Section 18],
- *Chapters [Section 20],
- *Cluster [Section 10],
- *Cues [Section 22],
- *Attachments [Section 21],
- *and Tags [Section 6.8].

The SeekHead Element (also known as MetaSeek) contains an index of Top Level Elements locations within the Segment. Use of the SeekHead Element is **RECOMMENDED**. Without a SeekHead Element, a Matroska parser would have to search the entire file to find all of the other Top Level Elements. This is due to Matroska's flexible ordering requirements; for instance, it is acceptable for the Chapters Element to be stored after the Cluster Elements.

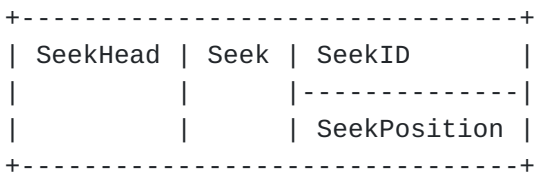

Figure 2: Representation of a SeekHead Element.

The Info Element contains vital information for identifying the whole Segment. This includes the title for the Segment, a randomly generated unique identifier, and the unique identifier(s) of any linked Segment Elements.

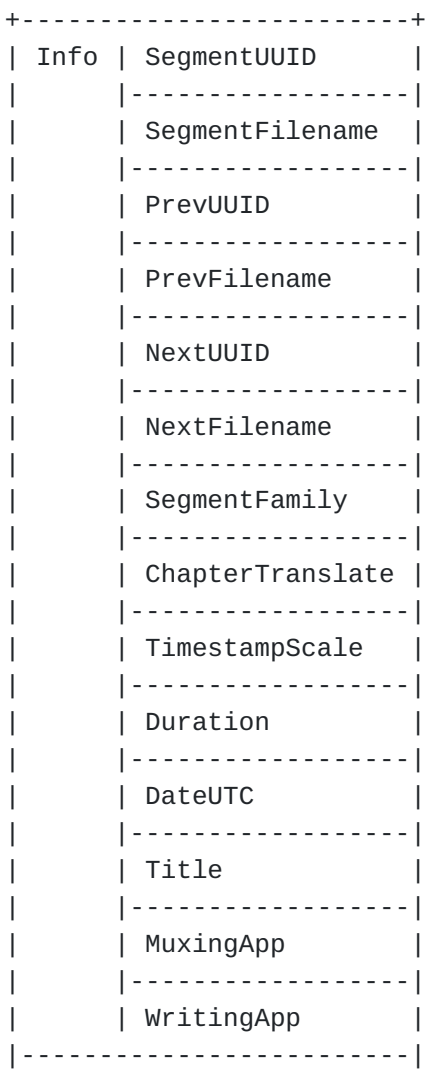

Figure 3: Representation of an Info Element and its Child Elements.

The Tracks Element defines the technical details for each track and can store the name, number, unique identifier, language, and type (audio, video, subtitles, etc.) of each track. For example, the

Tracks Element **MAY** store information about the resolution of a video track or sample rate of an audio track.

The Tracks Element **MUST** identify all the data needed by the codec to decode the data of the specified track. However, the data required is contingent on the codec used for the track. For example, a Track Element for uncompressed audio only requires the audio bit rate to be present. A codec such as AC-3 would require that the CodecID Element be present for all tracks, as it is the primary way to identify which codec to use to decode the track.

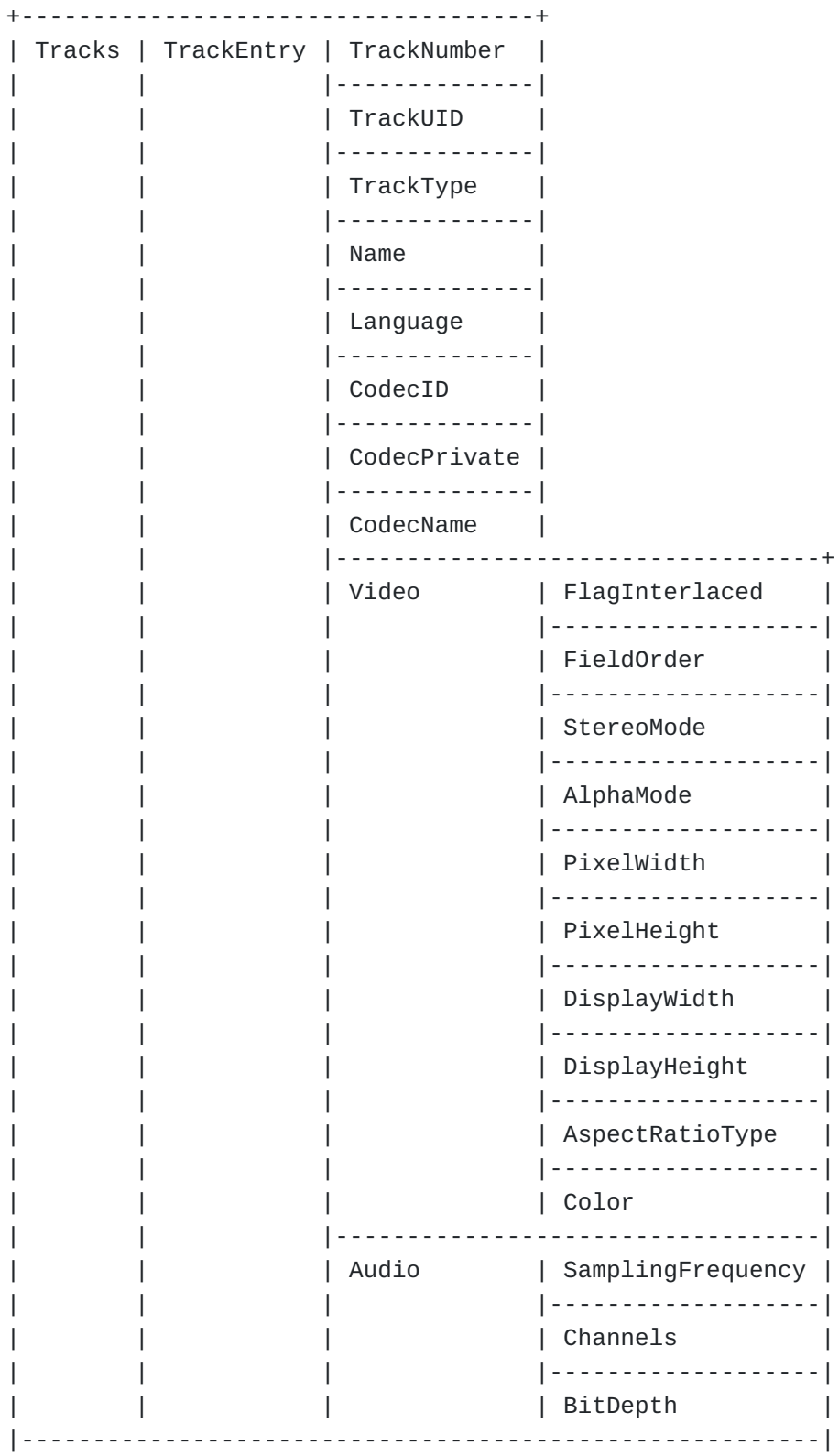

Figure 4: Representation of the Tracks Element and a selection of its Descendant Elements.

The Chapters Element lists all of the chapters. Chapters are a way to set predefined points to jump to in video or audio.

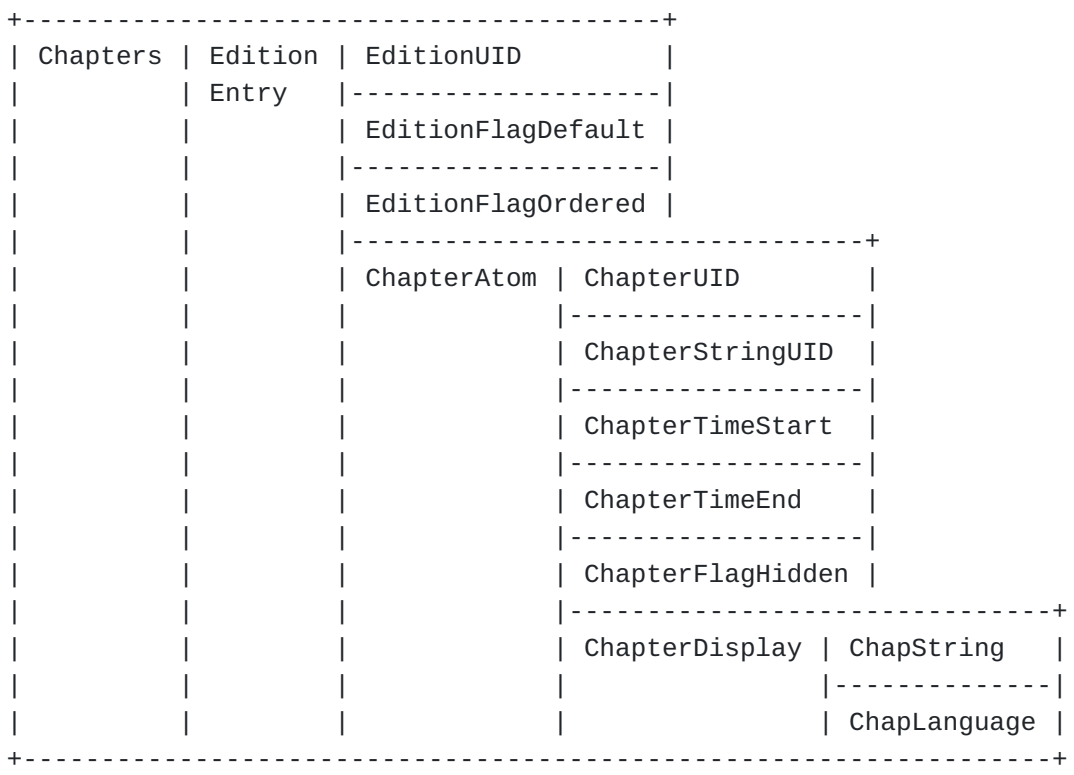

Figure 5:
Representation of the Chapters Element and a selection of its Descendant Elements.

Cluster Elements contain the content for each track, e.g., video frames. A Matroska file **SHOULD** contain at least one Cluster Element. The Cluster Element helps to break up SimpleBlock or BlockGroup Elements and helps with seeking and error protection. Every Cluster Element **MUST** contain a Timestamp Element. This **SHOULD** be the Timestamp Element used to play the first Block in the Cluster Element. Cluster Elements contain one or more block element, such as BlockGroup or SimpleBlock elements. In some situation, a Cluster Element **MAY NOT** contain any block element, for example in a live recording when no data has been collected.

A BlockGroup Element **MAY** contain a Block of data and any information relating directly to that Block.

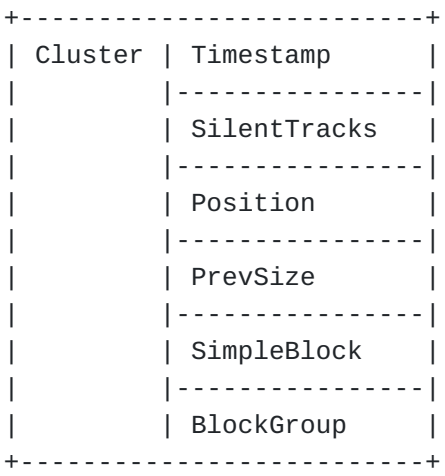

Figure 6: Representation of a Cluster Element and its immediate Child Elements.

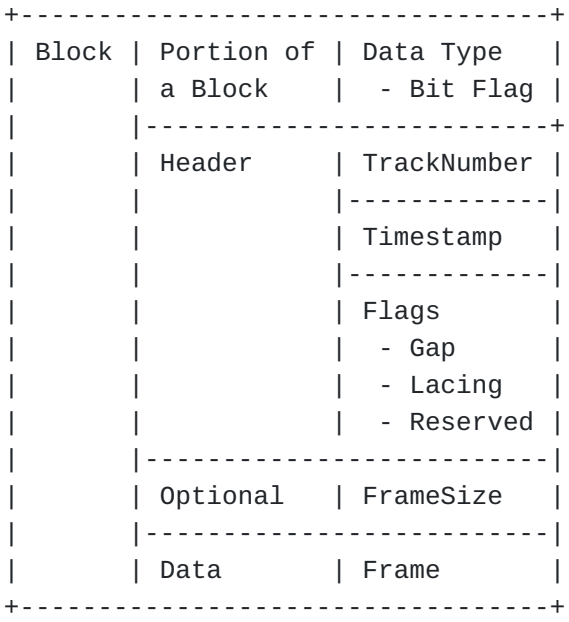

Figure 7: Representation of the Block Element structure.

Each Cluster **MUST** contain exactly one Timestamp Element. The Timestamp Element value **MUST** be stored once per Cluster. The Timestamp Element in the Cluster is relative to the entire Segment. The Timestamp Element **SHOULD** be the first Element in the Cluster it belongs to, or the second if that Cluster contains a CRC-32 element (Section 6.2)

Additionally, the Block contains an offset that, when added to the Cluster's Timestamp Element value, yields the Block's effective timestamp. Therefore, timestamp in the Block itself is relative to the Timestamp Element in the Cluster. For example, if the Timestamp Element in the Cluster is set to 10 seconds and a Block in that

Cluster is supposed to be played 12 seconds into the clip, the timestamp in the Block would be set to 2 seconds.

The ReferenceBlock in the BlockGroup is used instead of the basic "P-frame"/"B-frame" description. Instead of simply saying that this Block depends on the Block directly before, or directly afterwards, the Timestamp of the necessary Block is used. Because there can be as many ReferenceBlock Elements as necessary for a Block, it allows for some extremely complex referencing.

The Cues Element is used to seek when playing back a file by providing a temporal index for some of the Tracks. It is similar to the SeekHead Element, but used for seeking to a specific time when playing back the file. It is possible to seek without this element, but it is much more difficult because a Matroska Reader would have to 'hunt and peck' through the file looking for the correct timestamp.

The Cues Element **SHOULD** contain at least one CuePoint Element. Each CuePoint Element stores the position of the Cluster that contains the BlockGroup or SimpleBlock Element. The timestamp is stored in the CueTime Element and location is stored in the CueTrackPositions Element.

The Cues Element is flexible. For instance, Cues Element can be used to index every single timestamp of every Block or they can be indexed selectively.

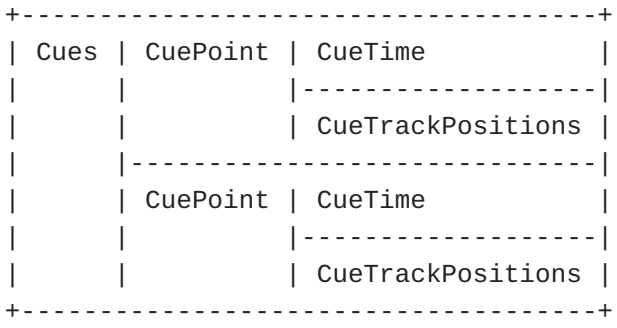

Figure 8: Representation of a Cues Element and two levels of its Descendant Elements.

The Attachments Element is for attaching files to a Matroska file such as pictures, fonts, webpages, etc.

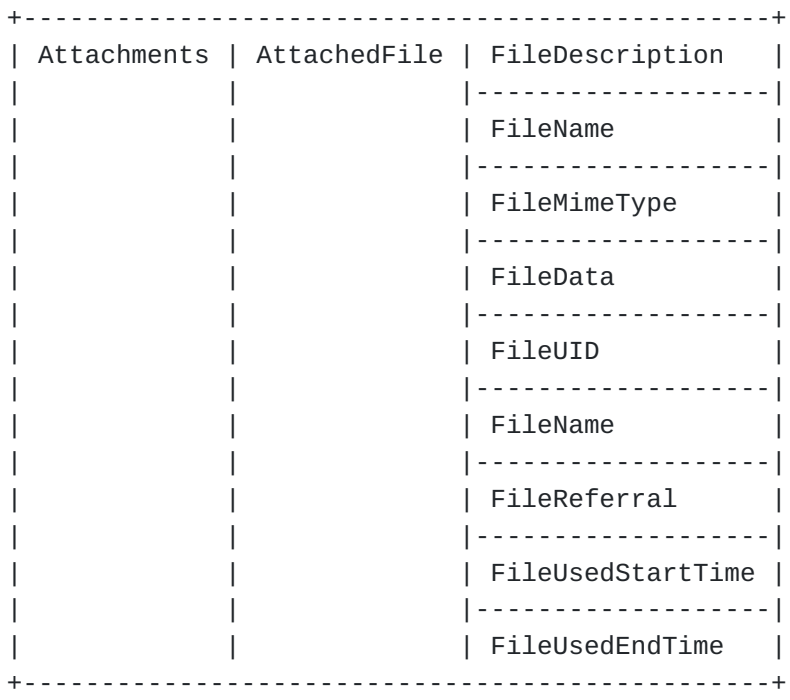

Figure 9: Representation of a Attachments Element.

The Tags Element contains metadata that describes the Segment and potentially its Tracks, Chapters, and Attachments. Each Track or Chapter that those tags applies to has its UID listed in the Tags. The Tags contain all extra information about the file: scriptwriter, singer, actors, directors, titles, edition, price, dates, genre, comments, etc. Tags can contain their values in multiple languages. For example, a movie's "title" Tag might contain both the original English title as well as the title it was released as in Germany.

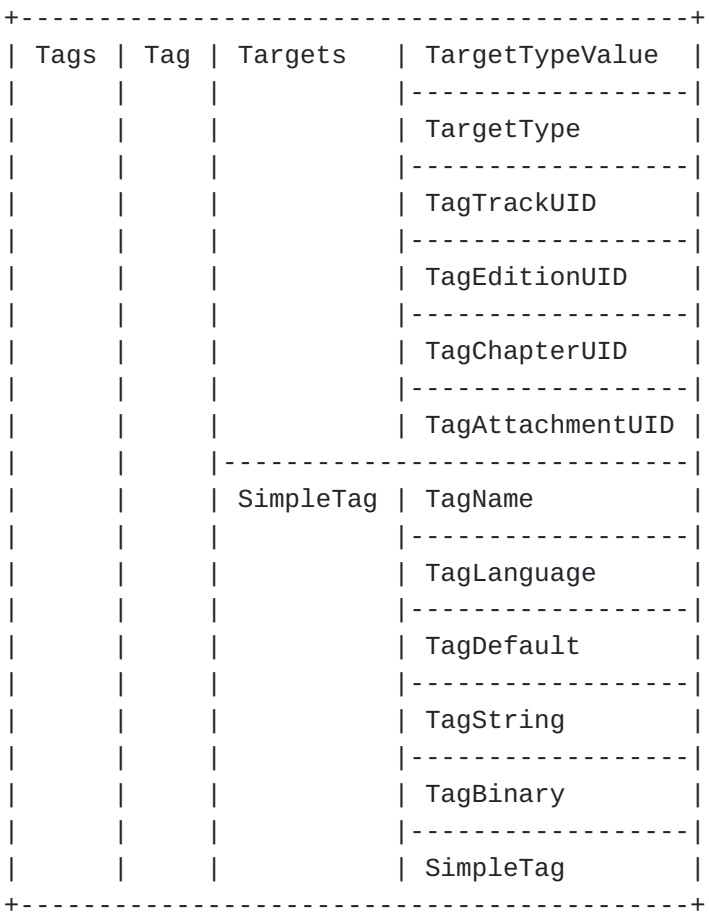

Figure 10: Representation of a Tags Element and three levels of its Children Elements.

5. Matroska Schema

This specification includes an EBML Schema, which defines the Elements and structure of Matroska as an EBML Document Type. The EBML Schema defines every valid Matroska element in a manner defined by the EBML specification.

Here the definition of each Matroska Element is provided.

5.1. Segment Element

name / type / id: Segment / master / 0x18538067

path: \Segment

minOccurs - maxOccurs: 1 - 1

unknownsizeallowed: True

definition: The Root Element that contains all other Top-Level Elements; see Section 4.4.

5.1.1. SeekHead Element

name / type / id: SeekHead / master / 0x114D9B74

path: \Segment\SeekHead

maxOccurs: 2

definition: Contains seeking information of Top-Level Elements; see Section 4.4.

5.1.1.1. Seek Element

name / type / id: Seek / master / 0x4DBB

path: \Segment\SeekHead\Seek

minOccurs: 1

definition: Contains a single seek entry to an EBML Element.

5.1.1.1.1. SeekID Element

name / type / id: SeekID / binary / 0x53AB

path: \Segment\SeekHead\Seek\SeekID

minOccurs - maxOccurs: 1 - 1

definition: The binary EBML ID of a Top-Level Element.

5.1.1.1.2. SeekPosition Element

name / type / id: SeekPosition / uinteger / 0x53AC

path: \Segment\SeekHead\Seek\SeekPosition

minOccurs - maxOccurs: 1 - 1

definition: The Segment Position (Section 16) of a Top-Level Element.

5.1.2. Info Element

name / type / id: Info / master / 0x1549A966

path: \Segment\Info

minOccurs - maxOccurs: 1 - 1

recurring: True

definition:
Contains general information about the Segment.

5.1.2.1. SegmentUUID Element

name / type / id: SegmentUUID / binary / 0x73A4

path: \Segment\Info\SegmentUUID

maxOccurs: 1

range: not 0

definition: A randomly generated unique ID to identify the Segment amongst many others (128 bits). It is effectively a Universally Unique IDentifier stored in binary form [RFC4122].

usage notes: If the Segment is a part of a Linked Segment, then this Element is **REQUIRED**.

5.1.2.2. SegmentFilename Element

name / type / id: SegmentFilename / utf-8 / 0x7384

path: \Segment\Info\SegmentFilename

maxOccurs: 1

definition: A filename corresponding to this Segment.

5.1.2.3. PrevUUID Element

name / type / id: PrevUUID / binary / 0x3CB923

path: \Segment\Info\PrevUUID

maxOccurs: 1

definition: A unique ID to identify the previous Segment of a Linked Segment (128 bits). Like the SegmentUUID, it is a Universally Unique IDentifier stored in binary form [RFC4122].

usage notes: If the Segment is a part of a Linked Segment that uses Hard Linking (Section 17.1), then either the PrevUUID or the NextUUID Element is **REQUIRED**. If a Segment contains a PrevUUID but not a NextUUID, then it **MAY** be considered as the last Segment of the Linked Segment. The PrevUUID **MUST NOT** be equal to the SegmentUUID.

5.1.2.4. PrevFilename Element

name / type / id:
PrevFilename / utf-8 / 0x3C83AB

path: \Segment\Info\PrevFilename

maxOccurs: 1

definition: A filename corresponding to the file of the previous Linked Segment.

usage notes: Provision of the previous filename is for display convenience, but PrevUUID **SHOULD** be considered authoritative for identifying the previous Segment in a Linked Segment.

5.1.2.5. NextUUID Element

name / type / id: NextUUID / binary / 0x3EB923

path: \Segment\Info\NextUUID

maxOccurs: 1

definition: A unique ID to identify the next Segment of a Linked Segment (128 bits). Like the SegmentUUID, it is a Universally Unique IDentifier stored in binary form [RFC4122].

usage notes: If the Segment is a part of a Linked Segment that uses Hard Linking (Section 17.1), then either the PrevUUID or the NextUUID Element is **REQUIRED**. If a Segment contains a NextUUID but not a PrevUUID, then it **MAY** be considered as the first Segment of the Linked Segment. The NextUUID **MUST NOT** be equal to the SegmentUUID.

5.1.2.6. NextFilename Element

name / type / id: NextFilename / utf-8 / 0x3E83BB

path: \Segment\Info\NextFilename

maxOccurs: 1

definition: A filename corresponding to the file of the next Linked Segment.

usage notes: Provision of the next filename is for display convenience, but NextUUID **SHOULD** be considered authoritative for identifying the Next Segment.

5.1.2.7. SegmentFamily Element

name / type / id:
SegmentFamily / binary / 0x4444

path: \Segment\Info\SegmentFamily

definition: A randomly generated unique ID that all Segments of a Linked Segment **MUST** share (128 bits). It is effectively a Universally Unique IDentifier stored in binary form [RFC4122].

usage notes: If the Segment Info contains a ChapterTranslate element, this Element is **REQUIRED**.

5.1.2.8. ChapterTranslate Element

name / type / id: ChapterTranslate / master / 0x6924

path: \Segment\Info\ChapterTranslate

definition: The mapping between this Segment and a segment value in the given Chapter Codec.

rationale: Chapter Codec may need to address different segments, but they may not know of the way to identify such segment when stored in Matroska. This element and its child elements add a way to map the internal segments known to the Chapter Codec to the Segment IDs in Matroska. This allows remuxing a file with Chapter Codec without changing the content of the codec data, just the Segment mapping.

5.1.2.8.1. ChapterTranslateID Element

name / type / id: ChapterTranslateID / binary / 0x69A5

path: \Segment\Info\ChapterTranslate\ChapterTranslateID

minOccurs - maxOccurs: 1 - 1

definition: The binary value used to represent this Segment in the chapter codec data. The format depends on the ChapProcessCodecID used; see Section 5.1.7.1.4.15.

5.1.2.8.2. ChapterTranslateCodec Element

name / type / id: ChapterTranslateCodec / uinteger / 0x69BF

path: \Segment\Info\ChapterTranslate\ChapterTranslateCodec

minOccurs - maxOccurs: 1 - 1

definition: This ChapterTranslate applies to this chapter codec of the given chapter edition(s); see Section 5.1.7.1.4.15.

defined values:

value	label	definition
0	Matroska Script	Chapter commands using the Matroska Script codec.
1	DVD-menu	Chapter commands using the DVD-like codec.

Table 1: ChapterTranslateCodec values

5.1.2.8.3. ChapterTranslateEditionUID Element

name / type / id: ChapterTranslateEditionUID / uinteger / 0x69FC

path: \Segment\Info\ChapterTranslate\ChapterTranslateEditionUID

definition: Specify a chapter edition UID on which this ChapterTranslate applies.

usage notes: When no ChapterTranslateEditionUID is specified in the ChapterTranslate, the ChapterTranslate applied to all chapter editions found in the Segment using the given ChapterTranslateCodec.

5.1.2.9. TimestampScale Element

name / type / id: TimestampScale / uinteger / 0x2AD7B1

path: \Segment\Info\TimestampScale

minOccurs - maxOccurs: 1 - 1

range: not 0

default: 1000000

definition: Base unit for Segment Ticks and Track Ticks, in nanoseconds. A TimestampScale value of 1.000.000 means scaled timestamps in the Segment are expressed in milliseconds; see Section 11 on how to interpret timestamps.

5.1.2.10. Duration Element

name / type / id: Duration / float / 0x4489

path: \Segment\Info\Duration

maxOccurs: 1

range: > 0x0p+0

definition: Duration of the Segment, expressed in Segment Ticks which is based on TimestampScale; see Section 11.1.

5.1.2.11. DateUTC Element

name / type / id: DateUTC / date / 0x4461

path: \Segment\Info\DateUTC

maxOccurs: 1

definition: The date and time that the Segment was created by the muxing application or library.

5.1.2.12. Title Element

name / type / id: Title / utf-8 / 0x7BA9

path: \Segment\Info\Title

maxOccurs: 1

definition: General name of the Segment.

5.1.2.13. MuxingApp Element

name / type / id: MuxingApp / utf-8 / 0x4D80

path: \Segment\Info\MuxingApp

minOccurs - maxOccurs: 1 - 1

definition: Muxing application or library (example: "libmatroska-0.4.3").

usage notes: Include the full name of the application or library followed by the version number.

5.1.2.14. WritingApp Element

name / type / id: WritingApp / utf-8 / 0x5741

path:
\Segment\Info\WritingApp

minOccurs - maxOccurs: 1 - 1

definition: Writing application (example: "mkvmerge-0.3.3").

usage notes: Include the full name of the application followed by the version number.

5.1.3. Cluster Element

name / type / id: Cluster / master / 0x1F43B675

path: \Segment\Cluster

unknownsizeallowed: True

definition: The Top-Level Element containing the (monolithic) Block structure.

5.1.3.1. Timestamp Element

name / type / id: Timestamp / uinteger / 0xE7

path: \Segment\Cluster\Timestamp

minOccurs - maxOccurs: 1 - 1

definition: Absolute timestamp of the cluster, expressed in Segment Ticks which is based on TimestampScale; see Section 11.1.

usage notes: This element **SHOULD** be the first child element of the Cluster it belongs to, or the second if that Cluster contains a CRC-32 element (Section 6.2).

5.1.3.2. PrevSize Element

name / type / id: PrevSize / uinteger / 0xAB

path: \Segment\Cluster\PrevSize

maxOccurs: 1

definition: Size of the previous Cluster, in octets. Can be useful for backward playing.

5.1.3.3. SimpleBlock Element

name / type / id: SimpleBlock / binary / 0xA3

path:
\Segment\Cluster\SimpleBlock

minver: 2

definition: Similar to Block, see Section 10, but without all the extra information, mostly used to reduced overhead when no extra feature is needed; see Section 10.3 on SimpleBlock Structure.

5.1.3.4. BlockGroup Element

name / type / id: BlockGroup / master / 0xA0

path: \Segment\Cluster\BlockGroup

definition: Basic container of information containing a single Block and information specific to that Block.

5.1.3.4.1. Block Element

name / type / id: Block / binary / 0xA1

path: \Segment\Cluster\BlockGroup\Block

minOccurs - maxOccurs: 1 - 1

definition: Block containing the actual data to be rendered and a timestamp relative to the Cluster Timestamp; see Section 10 on Block Structure.

5.1.3.4.2. BlockAdditions Element

name / type / id: BlockAdditions / master / 0x75A1

path: \Segment\Cluster\BlockGroup\BlockAdditions

maxOccurs: 1

definition: Contain additional binary data to complete the main one; see Codec BlockAdditions section of [MatroskaCodec] for more information. An EBML parser that has no knowledge of the Block structure could still see and use/skip these data.

5.1.3.4.2.1. BlockMore Element

name / type / id: BlockMore / master / 0xA6

path: \Segment\Cluster\BlockGroup\BlockAdditions\BlockMore

minOccurs: 1

definition:
Contain the BlockAdditional and some parameters.

5.1.3.4.2.2. BlockAdditional Element

name / type / id: BlockAdditional / binary / 0xA5

path:
\Segment\Cluster\BlockGroup\BlockAdditions\BlockMore\BlockAdditional

minOccurs - maxOccurs: 1 - 1

definition: Interpreted by the codec as it wishes (using the BlockAddID).

5.1.3.4.2.3. BlockAddID Element

name / type / id: BlockAddID / uinteger / 0xEE

path:
\Segment\Cluster\BlockGroup\BlockAdditions\BlockMore\BlockAddID

minOccurs - maxOccurs: 1 - 1

range: not 0

default: 1

definition: An ID to identify how to interpret the BlockAdditional data; see Codec BlockAdditions section of [MatroskaCodec] for more information. A value of 1 indicates that the meaning of the BlockAdditional data is defined by the codec. Any other value indicates the meaning of the BlockAdditional data is found in the BlockAddIDType found in the TrackEntry.

usage notes: Each BlockAddID value **MUST** be unique between all BlockMore elements found in a BlockAdditions.

usage notes: To keep MaxBlockAdditionID as low as possible, small values **SHOULD** be used.

5.1.3.4.3. BlockDuration Element

name / type / id: BlockDuration / uinteger / 0x9B

path: \Segment\Cluster\BlockGroup\BlockDuration

minOccurs - maxOccurs: see implementation notes - 1

default: see implementation notes

definition:

The duration of the Block, expressed in Track Ticks; see Section 11.1. The BlockDuration Element can be useful at the end of a Track to define the duration of the last frame (as there is no subsequent Block available), or when there is a break in a track like for subtitle tracks.

notes:

attribute	note
minOccurs	BlockDuration **MUST** be set (minOccurs=1) if the associated TrackEntry stores a DefaultDuration value.
default	When not written and with no DefaultDuration, the value is assumed to be the difference between the timestampof this Block and the timestamp of the next Block in "display" order (not coding order).

Table 2: BlockDuration implementation notes

5.1.3.4.4. ReferencePriority Element

name / type / id: ReferencePriority / uinteger / 0xFA

path: \Segment\Cluster\BlockGroup\ReferencePriority

minOccurs - maxOccurs: 1 - 1

default: 0

definition: This frame is referenced and has the specified cache priority. In cache only a frame of the same or higher priority can replace this frame. A value of 0 means the frame is not referenced.

5.1.3.4.5. ReferenceBlock Element

name / type / id: ReferenceBlock / integer / 0xFB

path: \Segment\Cluster\BlockGroup\ReferenceBlock

definition: A timestamp value, relative to the timestamp of the Block in this BlockGroup, expressed in Track Ticks; see Section 11.1. This is used to reference other frames necessary to decode this frame. The relative value **SHOULD** correspond to a valid Block this Block depends on. Historically Matroska Writer didn't write the actual Block(s) this Block depends on, but *some* Block in the past.

The value "0" **MAY** also be used to signify this Block cannot be decoded on its own, but without knownledge of which Block is

necessary. In this case, other ReferenceBlock **MUST NOT** be found in the same BlockGroup.

If the BlockGroup doesn't have any ReferenceBlock element, then the Block it contains can be decoded without using any other Block data.

5.1.3.4.6. CodecState Element

name / type / id: CodecState / binary / 0xA4

path: \Segment\Cluster\BlockGroup\CodecState

maxOccurs: 1

minver: 2

definition: The new codec state to use. Data interpretation is private to the codec. This information **SHOULD** always be referenced by a seek entry.

5.1.3.4.7. DiscardPadding Element

name / type / id: DiscardPadding / integer / 0x75A2

path: \Segment\Cluster\BlockGroup\DiscardPadding

maxOccurs: 1

minver: 4

definition: Duration of the silent data added to the Block, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1 (padding at the end of the Block for positive value, at the beginning of the Block for negative value). The duration of DiscardPadding is not calculated in the duration of the TrackEntry and **SHOULD** be discarded during playback.

5.1.4. Tracks Element

name / type / id: Tracks / master / 0x1654AE6B

path: \Segment\Tracks

maxOccurs: 1

recurring: True

definition: A Top-Level Element of information with many tracks described.

5.1.4.1. TrackEntry Element

name / type / id:
TrackEntry / master / 0xAE

path: \Segment\Tracks\TrackEntry

minOccurs: 1

definition: Describes a track with all Elements.

5.1.4.1.1. TrackNumber Element

name / type / id: TrackNumber / uinteger / 0xD7

path: \Segment\Tracks\TrackEntry\TrackNumber

minOccurs - maxOccurs: 1 - 1

range: not 0

definition: The track number as used in the Block Header (using more than 127 tracks is not encouraged, though the design allows an unlimited number).

5.1.4.1.2. TrackUID Element

name / type / id: TrackUID / uinteger / 0x73C5

path: \Segment\Tracks\TrackEntry\TrackUID

minOccurs - maxOccurs: 1 - 1

range: not 0

stream copy: True (Section 8)

definition: A unique ID to identify the Track.

5.1.4.1.3. TrackType Element

name / type / id: TrackType / uinteger / 0x83

path: \Segment\Tracks\TrackEntry\TrackType

minOccurs - maxOccurs: 1 - 1

stream copy: True (Section 8)

definition: The TrackType defines the type of each frame found in the Track. The value **SHOULD** be stored on 1 octet.

defined values:

value	label	each frame contains
1	video	An image.
2	audio	Audio samples.
3	complex	A mix of different other TrackType. The codec needs to define how the Matroska Player should interpret such data.
16	logo	An image to be rendered over the video track(s).
17	subtitle	Subtitle or closed caption data to be rendered over the video track(s).
18	buttons	Interactive button(s) to be rendered over the video track(s).
32	control	Metadata used to control the player of the Matroska Player.
33	metadata	Timed metadata that can be passed on to the Matroska Player.

Table 3: TrackType values

5.1.4.1.4. FlagEnabled Element

name / type / id: FlagEnabled / uinteger / 0xB9

path: \Segment\Tracks\TrackEntry\FlagEnabled

minOccurs - maxOccurs: 1 - 1

range: 0-1

default: 1

minver: 2

definition: Set to 1 if the track is usable. It is possible to turn a not usable track into a usable track using chapter codecs or control tracks.

5.1.4.1.5. FlagDefault Element

name / type / id: FlagDefault / uinteger / 0x88

path: \Segment\Tracks\TrackEntry\FlagDefault

minOccurs - maxOccurs: 1 - 1

range: 0-1

default: 1

definition: Set if that track (audio, video or subs) **SHOULD** be eligible for automatic selection by the player; see Section 19 for more details.

5.1.4.1.6. FlagForced Element

name / type / id: FlagForced / uinteger / 0x55AA

path: \Segment\Tracks\TrackEntry\FlagForced

minOccurs - maxOccurs: 1 - 1

range: 0-1

default: 0

definition: Applies only to subtitles. Set if that track **SHOULD** be eligible for automatic selection by the player if it matches the user's language preference, even if the user's preferences would normally not enable subtitles with the selected audio track; this can be used for tracks containing only translations of foreign-language audio or onscreen text. See Section 19 for more details.

5.1.4.1.7. FlagHearingImpaired Element

name / type / id: FlagHearingImpaired / uinteger / 0x55AB

path: \Segment\Tracks\TrackEntry\FlagHearingImpaired

maxOccurs: 1

range: 0-1

minver: 4

definition: Set to 1 if that track is suitable for users with hearing impairments, set to 0 if it is unsuitable for users with hearing impairments.

5.1.4.1.8. FlagVisualImpaired Element

name / type / id: FlagVisualImpaired / uinteger / 0x55AC

path: \Segment\Tracks\TrackEntry\FlagVisualImpaired

maxOccurs: 1

range: 0-1

minver: 4

definition: Set to 1 if that track is suitable for users with visual impairments, set to 0 if it is unsuitable for users with visual impairments.

5.1.4.1.9. FlagTextDescriptions Element

name / type / id: FlagTextDescriptions / uinteger / 0x55AD

path: \Segment\Tracks\TrackEntry\FlagTextDescriptions

maxOccurs: 1

range: 0-1

minver: 4

definition: Set to 1 if that track contains textual descriptions of video content, set to 0 if that track does not contain textual descriptions of video content.

5.1.4.1.10. FlagOriginal Element

name / type / id: FlagOriginal / uinteger / 0x55AE

path: \Segment\Tracks\TrackEntry\FlagOriginal

maxOccurs: 1

range: 0-1

minver: 4

definition: Set to 1 if that track is in the content's original language, set to 0 if it is a translation.

5.1.4.1.11. FlagCommentary Element

name / type / id: FlagCommentary / uinteger / 0x55AF

path: \Segment\Tracks\TrackEntry\FlagCommentary

maxOccurs: 1

range: 0-1

minver: 4

definition: Set to 1 if that track contains commentary, set to 0 if it does not contain commentary.

5.1.4.1.12. FlagLacing Element

name / type / id: FlagLacing / uinteger / 0x9C

path: \Segment\Tracks\TrackEntry\FlagLacing

minOccurs - maxOccurs:
1 - 1

range: 0-1

default: 1

definition: Set to 1 if the track **MAY** contain blocks using lacing. When set to 0 all blocks **MUST** have their lacing flags set to No lacing; see Section 10.4 on Block Lacing.

5.1.4.1.13. MinCache Element

name / type / id: MinCache / uinteger / 0x6DE7

path: \Segment\Tracks\TrackEntry\MinCache

minOccurs - maxOccurs: 1 - 1

default: 0

definition: The minimum number of frames a player **SHOULD** be able to cache during playback. If set to 0, the reference pseudo-cache system is not used.

5.1.4.1.14. MaxCache Element

name / type / id: MaxCache / uinteger / 0x6DF8

path: \Segment\Tracks\TrackEntry\MaxCache

maxOccurs: 1

definition: The maximum cache size necessary to store referenced frames in and the current frame. 0 means no cache is needed.

5.1.4.1.15. DefaultDuration Element

name / type / id: DefaultDuration / uinteger / 0x23E383

path: \Segment\Tracks\TrackEntry\DefaultDuration

maxOccurs: 1

range: not 0

stream copy: True (Section 8)

definition: Number of nanoseconds per frame, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1 (frame in the Matroska sense -- one Element put into a (Simple)Block).

5.1.4.1.16. DefaultDecodedFieldDuration Element

name / type / id: DefaultDecodedFieldDuration / uinteger / 0x234E7A

path: \Segment\Tracks\TrackEntry\DefaultDecodedFieldDuration

maxOccurs: 1

range: not 0

stream copy: True (Section 8)

minver: 4

definition: The period between two successive fields at the output of the decoding process, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1. see Section 9 for more information

5.1.4.1.17. TrackTimestampScale Element

name / type / id: TrackTimestampScale / float / 0x23314F

path: \Segment\Tracks\TrackEntry\TrackTimestampScale

minOccurs - maxOccurs: 1 - 1

range: > 0x0p+0

default: 0x1p+0

stream copy: True (Section 8)

maxver: 3

definition: The scale to apply on this track to work at normal speed in relation with other tracks (mostly used to adjust video speed when the audio length differs).

5.1.4.1.18. MaxBlockAdditionID Element

name / type / id: MaxBlockAdditionID / uinteger / 0x55EE

path: \Segment\Tracks\TrackEntry\MaxBlockAdditionID

minOccurs - maxOccurs: 1 - 1

default: 0

definition: The maximum value of BlockAddID (Section 5.1.3.4.2.3). A value 0 means there is no BlockAdditions (Section 5.1.3.4.2) for this track.

5.1.4.1.19. BlockAdditionMapping Element

name / type / id: BlockAdditionMapping / master / 0x41E4

path: \Segment\Tracks\TrackEntry\BlockAdditionMapping

minver: 4

definition: Contains elements that extend the track format, by adding content either to each frame, with BlockAddID (Section 5.1.3.4.2.3), or to the track as a whole with BlockAddIDExtraData.

5.1.4.1.19.1. BlockAddIDValue Element

name / type / id: BlockAddIDValue / uinteger / 0x41F0

path:
\Segment\Tracks\TrackEntry\BlockAdditionMapping\BlockAddIDValue

maxOccurs: 1

range: >=2

minver: 4

definition: If the track format extension needs content beside frames, the value refers to the BlockAddID (Section 5.1.3.4.2.3), value being described.

usage notes: To keep MaxBlockAdditionID as low as possible, small values **SHOULD** be used.

5.1.4.1.19.2. BlockAddIDName Element

name / type / id: BlockAddIDName / string / 0x41A4

path:
\Segment\Tracks\TrackEntry\BlockAdditionMapping\BlockAddIDName

maxOccurs: 1

minver: 4

definition: A human-friendly name describing the type of BlockAdditional data, as defined by the associated Block Additional Mapping.

5.1.4.1.19.3. BlockAddIDType Element

name / type / id: BlockAddIDType / uinteger / 0x41E7

path:
\Segment\Tracks\TrackEntry\BlockAdditionMapping\BlockAddIDType

minOccurs - maxOccurs: 1 - 1

default: 0

minver: 4

definition: Stores the registered identifier of the Block Additional Mapping to define how the BlockAdditional data should be handled.

usage notes: If BlockAddIDType is 0, the BlockAddIDValue and corresponding BlockAddID values **MUST** be 1.

5.1.4.1.19.4. BlockAddIDExtraData Element

name / type / id: BlockAddIDExtraData / binary / 0x41ED

path:
\Segment\Tracks\TrackEntry\BlockAdditionMapping\BlockAddIDExtraData

maxOccurs: 1

minver: 4

definition: Extra binary data that the BlockAddIDType can use to interpret the BlockAdditional data. The interpretation of the binary data depends on the BlockAddIDType value and the corresponding Block Additional Mapping.

5.1.4.1.20. Name Element

name / type / id: Name / utf-8 / 0x536E

path: \Segment\Tracks\TrackEntry\Name

maxOccurs: 1

definition: A human-readable track name.

5.1.4.1.21. Language Element

name / type / id: Language / string / 0x22B59C

path: \Segment\Tracks\TrackEntry\Language

minOccurs - maxOccurs: 1 - 1

default:
eng

definition: Specifies the language of the track in the Matroska languages form; see Section 12 on language codes. This Element **MUST** be ignored if the LanguageBCP47 Element is used in the same TrackEntry.

5.1.4.1.22. LanguageBCP47 Element

name / type / id: LanguageBCP47 / string / 0x22B59D

path: \Segment\Tracks\TrackEntry\LanguageBCP47

maxOccurs: 1

minver: 4

definition: Specifies the language of the track according to [BCP47] and using the IANA Language Subtag Registry [IANALangRegistry]. If this Element is used, then any Language Elements used in the same TrackEntry **MUST** be ignored.

5.1.4.1.23. CodecID Element

name / type / id: CodecID / string / 0x86

path: \Segment\Tracks\TrackEntry\CodecID

minOccurs - maxOccurs: 1 - 1

stream copy: True (Section 8)

definition: An ID corresponding to the codec, see [MatroskaCodec] for more info.

5.1.4.1.24. CodecPrivate Element

name / type / id: CodecPrivate / binary / 0x63A2

path: \Segment\Tracks\TrackEntry\CodecPrivate

maxOccurs: 1

stream copy: True (Section 8)

definition: Private data only known to the codec.

5.1.4.1.25. CodecName Element

name / type / id: CodecName / utf-8 / 0x258688

path:
\Segment\Tracks\TrackEntry\CodecName

maxOccurs: 1

definition: A human-readable string specifying the codec.

5.1.4.1.26. AttachmentLink Element

name / type / id: AttachmentLink / uinteger / 0x7446

path: \Segment\Tracks\TrackEntry\AttachmentLink

maxOccurs: 1

range: not 0

maxver: 3

definition: The UID of an attachment that is used by this codec.

usage notes: The value **MUST** match the FileUID value of an attachment found in this Segment.

5.1.4.1.27. TrackOverlay Element

name / type / id: TrackOverlay / uinteger / 0x6FAB

path: \Segment\Tracks\TrackEntry\TrackOverlay

definition: Specify that this track is an overlay track for the Track specified (in the u-integer). That means when this track has a gap, see Section 27.1 on SilentTracks, the overlay track **SHOULD** be used instead. The order of multiple TrackOverlay matters, the first one is the one that **SHOULD** be used. If not found it **SHOULD** be the second, etc.

5.1.4.1.28. CodecDelay Element

name / type / id: CodecDelay / uinteger / 0x56AA

path: \Segment\Tracks\TrackEntry\CodecDelay

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 4

definition:

CodecDelay is The codec-built-in delay, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1. It represents the amount of codec samples that will be discarded by the decoder during playback. This timestamp value **MUST** be subtracted from each frame timestamp in order to get the timestamp that will be actually played. The value **SHOULD** be small so the muxing of tracks with the same actual timestamp are in the same Cluster.

5.1.4.1.29. SeekPreRoll Element

name / type / id: SeekPreRoll / uinteger / 0x56BB

path: \Segment\Tracks\TrackEntry\SeekPreRoll

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 4

definition: After a discontinuity, SeekPreRoll is the duration of the data the decoder **MUST** decode before the decoded data is valid, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1.

5.1.4.1.30. TrackTranslate Element

name / type / id: TrackTranslate / master / 0x6624

path: \Segment\Tracks\TrackEntry\TrackTranslate

definition: The mapping between this TrackEntry and a track value in the given Chapter Codec.

rationale: Chapter Codec may need to address content in specific track, but they may not know of the way to identify tracks in Matroska. This element and its child elements add a way to map the internal tracks known to the Chapter Codec to the track IDs in Matroska. This allows remuxing a file with Chapter Codec without changing the content of the codec data, just the track mapping.

5.1.4.1.30.1. TrackTranslateTrackID Element

name / type / id: TrackTranslateTrackID / binary / 0x66A5

path:
\Segment\Tracks\TrackEntry\TrackTranslate\TrackTranslateTrackID

minOccurs - maxOccurs: 1 - 1

definition: The binary value used to represent this TrackEntry in the chapter codec data. The format depends on the ChapProcessCodecID used; see Section 5.1.7.1.4.15.

5.1.4.1.30.2. TrackTranslateCodec Element

name / type / id: TrackTranslateCodec / uinteger / 0x66BF

path: \Segment\Tracks\TrackEntry\TrackTranslate\TrackTranslateCodec

minOccurs - maxOccurs: 1 - 1

definition: This TrackTranslate applies to this chapter codec of the given chapter edition(s); see Section 5.1.7.1.4.15.

defined values:

value	label	definition
0	Matroska Script	Chapter commands using the Matroska Script codec.
1	DVD-menu	Chapter commands using the DVD-like codec.

Table 4: TrackTranslateCodec values

5.1.4.1.30.3. TrackTranslateEditionUID Element

name / type / id: TrackTranslateEditionUID / uinteger / 0x66FC

path:
\Segment\Tracks\TrackEntry\TrackTranslate\TrackTranslateEditionUID

definition: Specify a chapter edition UID on which this TrackTranslate applies.

usage notes: When no TrackTranslateEditionUID is specified in the TrackTranslate, the TrackTranslate applies to all chapter editions found in the Segment using the given TrackTranslateCodec.

5.1.4.1.31. Video Element

name / type / id:

Video / master / 0xE0

path: \Segment\Tracks\TrackEntry\Video

maxOccurs: 1

definition: Video settings.

5.1.4.1.31.1. FlagInterlaced Element

name / type / id: FlagInterlaced / uinteger / 0x9A

path: \Segment\Tracks\TrackEntry\Video\FlagInterlaced

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 2

definition: Specify whether the video frames in this track are interlaced or not.

defined values:

value	label	definition
0	undetermined	Unknown status.This value **SHOULD** be avoided.
1	interlaced	Interlaced frames.
2	progressive	No interlacing.

Table 5: FlagInterlaced values

5.1.4.1.31.2. FieldOrder Element

name / type / id: FieldOrder / uinteger / 0x9D

path: \Segment\Tracks\TrackEntry\Video\FieldOrder

minOccurs - maxOccurs: 1 - 1

default: 2

stream copy: True (Section 8)

minver: 4

definition: Specify the field ordering of video frames in this track.

defined values:

value	label	definition
0	progressive	Interlaced frames.This value **SHOULD** be avoided, setting FlagInterlaced to 2 is sufficient.
1	tff	Top field displayed first. Top field stored first.
2	undetermined	Unknown field order.This value **SHOULD** be avoided.
6	bff	Bottom field displayed first. Bottom field stored first.
9	bff(swapped)	Top field displayed first. Fields are interleaved in storage with the top line of the top field stored first.
14	tff(swapped)	Bottom field displayed first. Fields are interleaved in storage with the top line of the top field stored first.

Table 6: FieldOrder values

usage notes: If FlagInterlaced is not set to 1, this Element **MUST** be ignored.

5.1.4.1.31.3. StereoMode Element

name / type / id: StereoMode / uinteger / 0x53B8

path: \Segment\Tracks\TrackEntry\Video\StereoMode

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 3

definition: Stereo-3D video mode. There are some more details in Section 18.10.

restrictions:

value	label
0	mono
1	side by side (left eye first)

value	label
2	top - bottom (right eye is first)
3	top - bottom (left eye is first)
4	checkboard (right eye is first)
5	checkboard (left eye is first)
6	row interleaved (right eye is first)
7	row interleaved (left eye is first)
8	column interleaved (right eye is first)
9	column interleaved (left eye is first)
10	anaglyph (cyan/red)
11	side by side (right eye first)
12	anaglyph (green/magenta)
13	both eyes laced in one Block (left eye is first)
14	both eyes laced in one Block (right eye is first)

Table 7: StereoMode values

5.1.4.1.31.4. AlphaMode Element

name / type / id: AlphaMode / uinteger / 0x53C0

path: \Segment\Tracks\TrackEntry\Video\AlphaMode

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 3

definition: Indicate whether the BlockAdditional Element with BlockAddID of "1" contains Alpha data, as defined by to the Codec Mapping for the CodecID. Undefined values **SHOULD NOT** be used as the behavior of known implementations is different (considered either as 0 or 1).

defined values:

value	label	definition
0	none	The BlockAdditional Element with BlockAddID of "1" does not exist or **SHOULD NOT** be considered as containing such data.
1	present	The BlockAdditional Element with BlockAddID of "1" contains alpha channel data.

Table 8: AlphaMode values

5.1.4.1.31.5. OldStereoMode Element

name / type / id: OldStereoMode / uinteger / 0x53B9

path:
\Segment\Tracks\TrackEntry\Video\OldStereoMode

maxOccurs: 1

maxver: 2

definition: Bogus StereoMode value used in old versions of libmatroska.

restrictions:

value	label
0	mono
1	right eye
2	left eye
3	both eyes

Table 9: OldStereoMode values

usage notes: This Element **MUST NOT** be used. It was an incorrect value used in libmatroska up to 0.9.0.

5.1.4.1.31.6. PixelWidth Element

name / type / id: PixelWidth / uinteger / 0xB0

path: \Segment\Tracks\TrackEntry\Video\PixelWidth

minOccurs - maxOccurs: 1 - 1

range: not 0

stream copy: True (Section 8)

definition: Width of the encoded video frames in pixels.

5.1.4.1.31.7. PixelHeight Element

name / type / id: PixelHeight / uinteger / 0xBA

path: \Segment\Tracks\TrackEntry\Video\PixelHeight

minOccurs - maxOccurs: 1 - 1

range: not 0

stream copy: True (Section 8)

definition:
Height of the encoded video frames in pixels.

5.1.4.1.31.8. PixelCropBottom Element

name / type / id: PixelCropBottom / uinteger / 0x54AA

path: \Segment\Tracks\TrackEntry\Video\PixelCropBottom

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

definition: The number of video pixels to remove at the bottom of the image.

5.1.4.1.31.9. PixelCropTop Element

name / type / id: PixelCropTop / uinteger / 0x54BB

path: \Segment\Tracks\TrackEntry\Video\PixelCropTop

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

definition: The number of video pixels to remove at the top of the image.

5.1.4.1.31.10. PixelCropLeft Element

name / type / id: PixelCropLeft / uinteger / 0x54CC

path: \Segment\Tracks\TrackEntry\Video\PixelCropLeft

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

definition: The number of video pixels to remove on the left of the image.

5.1.4.1.31.11. PixelCropRight Element

name / type / id: PixelCropRight / uinteger / 0x54DD

path:
\Segment\Tracks\TrackEntry\Video\PixelCropRight

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

definition: The number of video pixels to remove on the right of the image.

5.1.4.1.31.12. DisplayWidth Element

name / type / id: DisplayWidth / uinteger / 0x54B0

path: \Segment\Tracks\TrackEntry\Video\DisplayWidth

maxOccurs: 1

range: not 0

default: see implementation notes

stream copy: True (Section 8)

definition: Width of the video frames to display. Applies to the video frame after cropping (PixelCrop* Elements).

notes:

attribute	note
default	If the DisplayUnit of the same TrackEntry is 0, then the default value for DisplayWidth is equal toPixelWidth - PixelCropLeft - PixelCropRight, else there is no default value.

Table 10: DisplayWidth implementation notes

5.1.4.1.31.13. DisplayHeight Element

name / type / id: DisplayHeight / uinteger / 0x54BA

path: \Segment\Tracks\TrackEntry\Video\DisplayHeight

maxOccurs: 1

range: not 0

default: see implementation notes

stream copy: True (Section 8)

definition:

Height of the video frames to display. Applies to the video frame after cropping (PixelCrop* Elements).

notes:

attribute	note
default	If the DisplayUnit of the same TrackEntry is 0, then the default value for DisplayHeight is equal toPixelHeight - PixelCropTop - PixelCropBottom, else there is no default value.

Table 11: DisplayHeight implementation notes

5.1.4.1.31.14. DisplayUnit Element

name / type / id: DisplayUnit / uinteger / 0x54B2

path: \Segment\Tracks\TrackEntry\Video\DisplayUnit

minOccurs - maxOccurs: 1 - 1

default: 0

definition: How DisplayWidth & DisplayHeight are interpreted.

restrictions:

value	label
0	pixels
1	centimeters
2	inches
3	display aspect ratio
4	unknown

Table 12: DisplayUnit values

5.1.4.1.31.15. UncompressedFourCC Element

name / type / id: UncompressedFourCC / binary / 0x2EB524

path: \Segment\Tracks\TrackEntry\Video\UncompressedFourCC

minOccurs - maxOccurs: see implementation notes - 1

stream copy: True (Section 8)

definition: Specify the uncompressed pixel format used for the Track's data as a FourCC. This value is similar in scope to the biCompression value of AVI's BITMAPINFO [AVIFormat]. See the YUV video formats [FourCC-YUV] and RGB video formats [FourCC-RGB] for common values.

notes:

attribute	note
minOccurs	UncompressedFourCC **MUST** be set (minOccurs=1) in TrackEntry, when the CodecID Element of the TrackEntry is set to "V_UNCOMPRESSED".

Table 13: UncompressedFourCC implementation notes

5.1.4.1.31.16. Colour Element

name / type / id: Colour / master / 0x55B0

path: \Segment\Tracks\TrackEntry\Video\Colour

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: Settings describing the colour format.

5.1.4.1.31.17. MatrixCoefficients Element

name / type / id: MatrixCoefficients / uinteger / 0x55B1

path: \Segment\Tracks\TrackEntry\Video\Colour\MatrixCoefficients

minOccurs - maxOccurs: 1 - 1

default: 2

stream copy: True (Section 8)

minver: 4

definition: The Matrix Coefficients of the video used to derive luma and chroma values from red, green, and blue color primaries. For clarity, the value and meanings for MatrixCoefficients are adopted from Table 4 of ISO/IEC 23001-8:2016 or ITU-T H.273.

restrictions:

value	label
0	Identity
1	ITU-R BT.709
2	unspecified
3	reserved
4	US FCC 73.682

value	label
5	ITU-R BT.470BG
6	SMPTE 170M
7	SMPTE 240M
8	YCoCg
9	BT2020 Non-constant Luminance
10	BT2020 Constant Luminance
11	SMPTE ST 2085
12	Chroma-derived Non-constant Luminance
13	Chroma-derived Constant Luminance
14	ITU-R BT.2100-0

Table 14: MatrixCoefficients values

5.1.4.1.31.18. BitsPerChannel Element

name / type / id: BitsPerChannel / uinteger / 0x55B2

path: \Segment\Tracks\TrackEntry\Video\Colour\BitsPerChannel

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 4

definition: Number of decoded bits per channel. A value of 0 indicates that the BitsPerChannel is unspecified.

5.1.4.1.31.19. ChromaSubsamplingHorz Element

name / type / id: ChromaSubsamplingHorz / uinteger / 0x55B3

path: \Segment\Tracks\TrackEntry\Video\Colour\ChromaSubsamplingHorz

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: The amount of pixels to remove in the Cr and Cb channels for every pixel not removed horizontally. Example: For video with 4:2:0 chroma subsampling, the ChromaSubsamplingHorz **SHOULD** be set to 1.

5.1.4.1.31.20. ChromaSubsamplingVert Element

name / type / id: ChromaSubsamplingVert / uinteger / 0x55B4

path:
\Segment\Tracks\TrackEntry\Video\Colour\ChromaSubsamplingVert

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: The amount of pixels to remove in the Cr and Cb channels for every pixel not removed vertically. Example: For video with 4:2:0 chroma subsampling, the ChromaSubsamplingVert **SHOULD** be set to 1.

5.1.4.1.31.21. CbSubsamplingHorz Element

name / type / id: CbSubsamplingHorz / uinteger / 0x55B5

path: \Segment\Tracks\TrackEntry\Video\Colour\CbSubsamplingHorz

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: The amount of pixels to remove in the Cb channel for every pixel not removed horizontally. This is additive with ChromaSubsamplingHorz. Example: For video with 4:2:1 chroma subsampling, the ChromaSubsamplingHorz **SHOULD** be set to 1 and CbSubsamplingHorz **SHOULD** be set to 1.

5.1.4.1.31.22. CbSubsamplingVert Element

name / type / id: CbSubsamplingVert / uinteger / 0x55B6

path: \Segment\Tracks\TrackEntry\Video\Colour\CbSubsamplingVert

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: The amount of pixels to remove in the Cb channel for every pixel not removed vertically. This is additive with ChromaSubsamplingVert.

5.1.4.1.31.23. ChromaSitingHorz Element

name / type / id: ChromaSitingHorz / uinteger / 0x55B7

path:
\Segment\Tracks\TrackEntry\Video\Colour\ChromaSitingHorz

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 4

definition: How chroma is subsampled horizontally.

restrictions:

value	label
0	unspecified
1	left collocated
2	half

Table 15: ChromaSitingHorz values

5.1.4.1.31.24. ChromaSitingVert Element

name / type / id: ChromaSitingVert / uinteger / 0x55B8

path: \Segment\Tracks\TrackEntry\Video\Colour\ChromaSitingVert

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 4

definition: How chroma is subsampled vertically.

restrictions:

value	label
0	unspecified
1	top collocated
2	half

Table 16: ChromaSitingVert values

5.1.4.1.31.25. Range Element

name / type / id: Range / uinteger / 0x55B9

path:
\Segment\Tracks\TrackEntry\Video\Colour\Range

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 4

definition: Clipping of the color ranges.

restrictions:

value	label
0	unspecified
1	broadcast range
2	full range (no clipping)
3	defined by MatrixCoefficients / TransferCharacteristics

Table 17: Range values

5.1.4.1.31.26. TransferCharacteristics Element

name / type / id: TransferCharacteristics / uinteger / 0x55BA

path:
\Segment\Tracks\TrackEntry\Video\Colour\TransferCharacteristics

minOccurs - maxOccurs: 1 - 1

default: 2

stream copy: True (Section 8)

minver: 4

definition: The transfer characteristics of the video. For clarity, the value and meanings for TransferCharacteristics are adopted from Table 3 of ISO/IEC 23091-4 or ITU-T H.273.

restrictions:

value	label
0	reserved
1	ITU-R BT.709
2	unspecified
3	reserved
4	Gamma 2.2 curve - BT.470M

value	label
5	Gamma 2.8 curve - BT.470BG
6	SMPTE 170M
7	SMPTE 240M
8	Linear
9	Log
10	Log Sqrt
11	IEC 61966-2-4
12	ITU-R BT.1361 Extended Colour Gamut
13	IEC 61966-2-1
14	ITU-R BT.2020 10 bit
15	ITU-R BT.2020 12 bit
16	ITU-R BT.2100 Perceptual Quantization
17	SMPTE ST 428-1
18	ARIB STD-B67 (HLG)

Table 18: TransferCharacteristics values

5.1.4.1.31.27. Primaries Element

name / type / id: Primaries / uinteger / 0x55BB

path: \Segment\Tracks\TrackEntry\Video\Colour\Primaries

minOccurs - maxOccurs: 1 - 1

default: 2

stream copy: True (Section 8)

minver: 4

definition: The colour primaries of the video. For clarity, the value and meanings for Primaries are adopted from Table 2 of ISO/IEC 23091-4 or ITU-T H.273.

restrictions:

value	label
0	reserved
1	ITU-R BT.709
2	unspecified
3	reserved
4	ITU-R BT.470M
5	ITU-R BT.470BG - BT.601 625
6	ITU-R BT.601 525 - SMPTE 170M
7	SMPTE 240M
8	FILM
9	ITU-R BT.2020

value	label
10	SMPTE ST 428-1
11	SMPTE RP 432-2
12	SMPTE EG 432-2
22	EBU Tech. 3213-E - JEDEC P22 phosphors

Table 19: Primaries values

5.1.4.1.31.28. MaxCLL Element

name / type / id: MaxCLL / uinteger / 0x55BC

path: \Segment\Tracks\TrackEntry\Video\Colour\MaxCLL

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: Maximum brightness of a single pixel (Maximum Content Light Level) in candelas per square meter (cd/m^2).

5.1.4.1.31.29. MaxFALL Element

name / type / id: MaxFALL / uinteger / 0x55BD

path: \Segment\Tracks\TrackEntry\Video\Colour\MaxFALL

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: Maximum brightness of a single full frame (Maximum Frame-Average Light Level) in candelas per square meter (cd/m^2).

5.1.4.1.31.30. MasteringMetadata Element

name / type / id: MasteringMetadata / master / 0x55D0

path: \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: SMPTE 2086 mastering data.

5.1.4.1.31.31. PrimaryRChromaticityX Element

name / type / id:
PrimaryRChromaticityX / float / 0x55D1

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\Primary RChromaticityX

maxOccurs: 1

range: 0-1

stream copy: True (Section 8)

minver: 4

definition: Red X chromaticity coordinate, as defined by [CIE-1931].

5.1.4.1.31.32. PrimaryRChromaticityY Element

name / type / id: PrimaryRChromaticityY / float / 0x55D2

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\Primary RChromaticityY

maxOccurs: 1

range: 0-1

stream copy: True (Section 8)

minver: 4

definition: Red Y chromaticity coordinate, as defined by [CIE-1931]].

5.1.4.1.31.33. PrimaryGChromaticityX Element

name / type / id: PrimaryGChromaticityX / float / 0x55D3

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\Primary GChromaticityX

maxOccurs: 1

range: 0-1

stream copy: True (Section 8)

minver:
4

definition: Green X chromaticity coordinate, as defined by [CIE-1931]].

5.1.4.1.31.34. PrimaryGChromaticityY Element

name / type / id: PrimaryGChromaticityY / float / 0x55D4

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\PrimaryGChromaticityY

maxOccurs: 1

range: 0-1

stream copy: True (Section 8)

minver: 4

definition: Green Y chromaticity coordinate, as defined by [CIE-1931]].

5.1.4.1.31.35. PrimaryBChromaticityX Element

name / type / id: PrimaryBChromaticityX / float / 0x55D5

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\PrimaryBChromaticityX

maxOccurs: 1

range: 0-1

stream copy: True (Section 8)

minver: 4

definition: Blue X chromaticity coordinate, as defined by [CIE-1931]].

5.1.4.1.31.36. PrimaryBChromaticityY Element

name / type / id: PrimaryBChromaticityY / float / 0x55D6

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\PrimaryBChromaticityY

maxOccurs:
1

range: 0-1

stream copy: True (Section 8)

minver: 4

definition: Blue Y chromaticity coordinate, as defined by [CIE-1931]].

5.1.4.1.31.37. WhitePointChromaticityX Element

name / type / id: WhitePointChromaticityX / float / 0x55D7

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\WhitePointChromaticityX

maxOccurs: 1

range: 0-1

stream copy: True (Section 8)

minver: 4

definition: White X chromaticity coordinate, as defined by [CIE-1931]].

5.1.4.1.31.38. WhitePointChromaticityY Element

name / type / id: WhitePointChromaticityY / float / 0x55D8

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\WhitePointChromaticityY

maxOccurs: 1

range: 0-1

stream copy: True (Section 8)

minver: 4

definition: White Y chromaticity coordinate, as defined by [CIE-1931]].

5.1.4.1.31.39. LuminanceMax Element

name / type / id:
LuminanceMax / float / 0x55D9

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\LuminanceMax

maxOccurs: 1

range: >= 0x0p+0

stream copy: True (Section 8)

minver: 4

definition: Maximum luminance. Represented in candelas per square meter (cd/m^2).

5.1.4.1.31.40. LuminanceMin Element

name / type / id: LuminanceMin / float / 0x55DA

path:
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\LuminanceMin

maxOccurs: 1

range: >= 0x0p+0

stream copy: True (Section 8)

minver: 4

definition: Minimum luminance. Represented in candelas per square meter (cd/m^2).

5.1.4.1.31.41. Projection Element

name / type / id: Projection / master / 0x7670

path: \Segment\Tracks\TrackEntry\Video\Projection

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: Describes the video projection details. Used to render spherical, VR videos or flipping videos horizontally/vertically.

5.1.4.1.31.42. ProjectionType Element

name / type / id: ProjectionType / uinteger / 0x7671

path: \Segment\Tracks\TrackEntry\Video\Projection\ProjectionType

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

minver: 4

definition: Describes the projection used for this video track.

restrictions:

value	label
0	rectangular
1	equirectangular
2	cubemap
3	mesh

Table 20: ProjectionType values

5.1.4.1.31.43. ProjectionPrivate Element

name / type / id: ProjectionPrivate / binary / 0x7672

path: \Segment\Tracks\TrackEntry\Video\Projection\ProjectionPrivate

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: Private data that only applies to a specific projection.

*If ProjectionType equals 0 (Rectangular), then this element must not be present.

*If ProjectionType equals 1 (Equirectangular), then this element must be present and contain the same binary data that would be stored inside an ISOBMFF Equirectangular Projection Box ('equi').

*If ProjectionType equals 2 (Cubemap), then this element must be present and contain the same binary data that would be stored inside an ISOBMFF Cubemap Projection Box ('cbmp').

*If ProjectionType equals 3 (Mesh), then this element must be present and contain the same binary data that would be stored inside an ISOBMFF Mesh Projection Box ('mshp').

usage notes: ISOBMFF box size and fourcc fields are not included in the binary data, but the FullBox version and flag fields are. This is to avoid redundant framing information while preserving versioning and semantics between the two container formats.

5.1.4.1.31.44. ProjectionPoseYaw Element

name / type / id: ProjectionPoseYaw / float / 0x7673

path: \Segment\Tracks\TrackEntry\Video\Projection\ProjectionPoseYaw

minOccurs - maxOccurs: 1 - 1

range: >= -0xB4p+0, <= 0xB4p+0

default: 0x0p+0

stream copy: True (Section 8)

minver: 4

definition: Specifies a yaw rotation to the projection.

Value represents a clockwise rotation, in degrees, around the up vector. This rotation must be applied before any ProjectionPosePitch or ProjectionPoseRoll rotations. The value of this element **MUST** be in the -180 to 180 degree range, both included.

Setting ProjectionPoseYaw to 180 or -180 degrees, with the ProjectionPoseRoll and ProjectionPosePitch set to 0 degrees flips the image horizontally.

5.1.4.1.31.45. ProjectionPosePitch Element

name / type / id: ProjectionPosePitch / float / 0x7674

path:
\Segment\Tracks\TrackEntry\Video\Projection\ProjectionPosePitch

minOccurs - maxOccurs: 1 - 1

range: >= -0x5Ap+0, <= 0x5Ap+0

default:
0x0p+0

stream copy: True (Section 8)

minver: 4

definition: Specifies a pitch rotation to the projection.

Value represents a counter-clockwise rotation, in degrees, around the right vector. This rotation must be applied after the ProjectionPoseYaw rotation and before the ProjectionPoseRoll rotation. The value of this element **MUST** be in the -90 to 90 degree range, both included.

5.1.4.1.31.46. ProjectionPoseRoll Element

name / type / id: ProjectionPoseRoll / float / 0x7675

path:
\Segment\Tracks\TrackEntry\Video\Projection\ProjectionPoseRoll

minOccurs - maxOccurs: 1 - 1

range: >= -0xB4p+0, <= 0xB4p+0

default: 0x0p+0

stream copy: True (Section 8)

minver: 4

definition: Specifies a roll rotation to the projection.

Value represents a counter-clockwise rotation, in degrees, around the forward vector. This rotation must be applied after the ProjectionPoseYaw and ProjectionPosePitch rotations. The value of this element **MUST** be in the -180 to 180 degree range, both included.

Setting ProjectionPoseRoll to 180 or -180 degrees, the ProjectionPoseYaw to 180 or -180 degrees with ProjectionPosePitch set to 0 degrees flips the image vertically.

Setting ProjectionPoseRoll to 180 or -180 degrees, with the ProjectionPoseYaw and ProjectionPosePitch set to 0 degrees flips the image horizontally and vertically.

5.1.4.1.32. Audio Element

name / type / id: Audio / master / 0xE1

path:
\Segment\Tracks\TrackEntry\Audio

maxOccurs: 1

definition: Audio settings.

5.1.4.1.32.1. SamplingFrequency Element

name / type / id: SamplingFrequency / float / 0xB5

path: \Segment\Tracks\TrackEntry\Audio\SamplingFrequency

minOccurs - maxOccurs: 1 - 1

range: > 0x0p+0

default: 0x1.f4p+12

stream copy: True (Section 8)

definition: Sampling frequency in Hz.

5.1.4.1.32.2. OutputSamplingFrequency Element

name / type / id: OutputSamplingFrequency / float / 0x78B5

path: \Segment\Tracks\TrackEntry\Audio\OutputSamplingFrequency

maxOccurs: 1

range: > 0x0p+0

default: see implementation notes

definition: Real output sampling frequency in Hz (used for SBR techniques).

notes:

attribute	note
default	The default value for OutputSamplingFrequency of the same TrackEntry is equal to the SamplingFrequency.

Table 21: OutputSamplingFrequency implementation notes

5.1.4.1.32.3. Channels Element

name / type / id: Channels / uinteger / 0x9F

path: \Segment\Tracks\TrackEntry\Audio\Channels

minOccurs - maxOccurs:
1 - 1

range: not 0

default: 1

stream copy: True (Section 8)

definition: Numbers of channels in the track.

5.1.4.1.32.4. BitDepth Element

name / type / id: BitDepth / uinteger / 0x6264

path: \Segment\Tracks\TrackEntry\Audio\BitDepth

maxOccurs: 1

range: not 0

stream copy: True (Section 8)

definition: Bits per sample, mostly used for PCM.

5.1.4.1.33. TrackOperation Element

name / type / id: TrackOperation / master / 0xE2

path: \Segment\Tracks\TrackEntry\TrackOperation

maxOccurs: 1

stream copy: True (Section 8)

minver: 3

definition: Operation that needs to be applied on tracks to create this virtual track. For more details look at Section 18.8.

5.1.4.1.33.1. TrackCombinePlanes Element

name / type / id: TrackCombinePlanes / master / 0xE3

path: \Segment\Tracks\TrackEntry\TrackOperation\TrackCombinePlanes

maxOccurs: 1

stream copy: True (Section 8)

minver: 3

definition:
Contains the list of all video plane tracks that need to be combined to create this 3D track

5.1.4.1.33.2. TrackPlane Element

name / type / id: TrackPlane / master / 0xE4

path:
\Segment\Tracks\TrackEntry\TrackOperation\TrackCombinePlanes\TrackPlane

minOccurs: 1

stream copy: True (Section 8)

minver: 3

definition: Contains a video plane track that need to be combined to create this 3D track

5.1.4.1.33.3. TrackPlaneUID Element

name / type / id: TrackPlaneUID / uinteger / 0xE5

path:
\Segment\Tracks\TrackEntry\TrackOperation\TrackCombinePlanes\TrackPlane\TrackPlaneUID

minOccurs - maxOccurs: 1 - 1

range: not 0

stream copy: True (Section 8)

minver: 3

definition: The trackUID number of the track representing the plane.

5.1.4.1.33.4. TrackPlaneType Element

name / type / id: TrackPlaneType / uinteger / 0xE6

path:
\Segment\Tracks\TrackEntry\TrackOperation\TrackCombinePlanes\TrackPlane\TrackPlaneType

minOccurs - maxOccurs: 1 - 1

stream copy: True (Section 8)

minver:
3

definition: The kind of plane this track corresponds to.

restrictions:

value	label
0	left eye
1	right eye
2	background

Table 22: TrackPlaneType values

5.1.4.1.33.5. TrackJoinBlocks Element

name / type / id: TrackJoinBlocks / master / 0xE9

path: \Segment\Tracks\TrackEntry\TrackOperation\TrackJoinBlocks

maxOccurs: 1

stream copy: True (Section 8)

minver: 3

definition: Contains the list of all tracks whose Blocks need to be combined to create this virtual track

5.1.4.1.33.6. TrackJoinUID Element

name / type / id: TrackJoinUID / uinteger / 0xED

path:
\Segment\Tracks\TrackEntry\TrackOperation\TrackJoinBlocks\TrackJoinUID

minOccurs: 1

range: not 0

stream copy: True (Section 8)

minver: 3

definition: The trackUID number of a track whose blocks are used to create this virtual track.

5.1.4.1.34. ContentEncodings Element

name / type / id:
ContentEncodings / master / 0x6D80

path: \Segment\Tracks\TrackEntry\ContentEncodings

maxOccurs: 1

stream copy: True (Section 8)

definition: Settings for several content encoding mechanisms like compression or encryption.

5.1.4.1.34.1. ContentEncoding Element

name / type / id: ContentEncoding / master / 0x6240

path: \Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding

minOccurs: 1

stream copy: True (Section 8)

definition: Settings for one content encoding like compression or encryption.

5.1.4.1.34.2. ContentEncodingOrder Element

name / type / id: ContentEncodingOrder / uinteger / 0x5031

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncodingOrder

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

definition: Tell in which order to apply each ContentEncoding of the ContentEncodings. The decoder/demuxer **MUST** start with the ContentEncoding with the highest ContentEncodingOrder and work its way down to the ContentEncoding with the lowest ContentEncodingOrder. This value **MUST** be unique over for each ContentEncoding found in the ContentEncodings of this TrackEntry.

5.1.4.1.34.3. ContentEncodingScope Element

name / type / id: ContentEncodingScope / uinteger / 0x5032

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncodingScope

minOccurs - maxOccurs: 1 - 1

default: 1

stream copy: True (Section 8)

definition: A bit field that describes which Elements have been modified in this way. Values (big-endian) can be OR'ed.

defined values:

value	label	definition
1	Block	All frame contents, excluding lacing data.
2	Private	The track's private data.
4	Next	The next ContentEncoding (next ContentEncodingOrder. Either the data inside ContentCompression and/or ContentEncryption).This value **SHOULD NOT** be used.

Table 23: ContentEncodingScope values

5.1.4.1.34.4. ContentEncodingType Element

name / type / id: ContentEncodingType / uinteger / 0x5033

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncodingType

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

definition: A value describing what kind of transformation is applied.

restrictions:

value	label
0	Compression
1	Encryption

Table 24:
ContentEncodingType
values

5.1.4.1.34.5. ContentCompression Element

name / type / id: ContentCompression / master / 0x5034

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentCompression

maxOccurs: 1

stream copy: True (Section 8)

definition: Settings describing the compression used. This Element **MUST** be present if the value of ContentEncodingType is 0 and absent otherwise. Each block **MUST** be decompressable even if no previous block is available in order not to prevent seeking.

5.1.4.1.34.6. ContentCompAlgo Element

name / type / id: ContentCompAlgo / uinteger / 0x4254

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentCompression\ContentCompAlgo

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

definition: The compression algorithm used.

defined values:

value	label	definition
0	zlib	zlib compression [RFC1950].
1	bzlib	bzip2 compression [BZIP2], **SHOULD NOT** be used; see usage notes.
2	lzo1x	Lempel-Ziv-Oberhumer compression [LZO], **SHOULD NOT** be used; see usage notes.
3	Header Stripping	Octets in ContentCompSettings (Section 5.1.4.1.34.7) have been stripped from each frame.

Table 25: ContentCompAlgo values

usage notes:

Compression method "1" (bzlib) and "2" (lzo1x) are lacking proper documentation on the format which limits implementation possibilities. Due to licensing conflicts on commonly available libraries compression methods "2" (lzo1x) does not offer widespread interoperability. Decoding implementations **MAY** support methods "1" and "2" as possible. The use of these compression methods **SHOULD NOT** be used as a default.

5.1.4.1.34.7. ContentCompSettings Element

name / type / id: ContentCompSettings / binary / 0x4255

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentCompression\ContentCompSettings

maxOccurs: 1

stream copy: True (Section 8)

definition: Settings that might be needed by the decompressor. For Header Stripping (ContentCompAlgo=3), the bytes that were removed from the beginning of each frames of the track.

5.1.4.1.34.8. ContentEncryption Element

name / type / id: ContentEncryption / master / 0x5035

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption

maxOccurs: 1

stream copy: True (Section 8)

definition: Settings describing the encryption used. This Element **MUST** be present if the value of ContentEncodingType is 1 (encryption) and **MUST** be ignored otherwise.

5.1.4.1.34.9. ContentEncAlgo Element

name / type / id: ContentEncAlgo / uinteger / 0x47E1

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\ContentEncAlgo

minOccurs - maxOccurs: 1 - 1

default: 0

stream copy: True (Section 8)

definition: The encryption algorithm used.

defined values:

value	label	definition
0	Not encrypted	The data are not encrypted.
1	DES	Data Encryption Standard (DES) [FIPS.46-3].
2	3DES	Triple Data Encryption Algorithm [SP.800-67].
3	Twofish	Twofish Encryption Algorithm [Twofish].
4	Blowfish	Blowfish Encryption Algorithm [Blowfish].
5	AES	Advanced Encryption Standard (AES) [FIPS.197].

Table 26: ContentEncAlgo values

5.1.4.1.34.10. ContentEncKeyID Element

name / type / id: ContentEncKeyID / binary / 0x47E2

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\ContentEncKeyID

maxOccurs: 1

stream copy: True (Section 8)

definition: For public key algorithms this is the ID of the public key the the data was encrypted with.

5.1.4.1.34.11. ContentEncAESSettings Element

name / type / id: ContentEncAESSettings / master / 0x47E7

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\ContentEncAESSettings

maxOccurs: 1

stream copy: True (Section 8)

minver: 4

definition: Settings describing the encryption algorithm used.

notes:

attribute	note
maxOccurs	ContentEncAESSettings **MUST NOT** be set (maxOccurs=0) if ContentEncAlgo is not AES (5).

Table 27: ContentEncAESSettings implementation notes

5.1.4.1.34.12. AESSettingsCipherMode Element

name / type / id: AESSettingsCipherMode / uinteger / 0x47E8

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\ContentEncAESSettings\AESSettingsCipherMode

minOccurs - maxOccurs: 1 - 1

stream copy: True (Section 8)

minver: 4

definition: The AES cipher mode used in the encryption.

defined values:

value	label	definition
1	AES-CTR	Counter [SP.800-38A].
2	AES-CBC	Cipher Block Chaining [SP.800-38A].

Table 28: AESSettingsCipherMode values

notes:

attribute	note
maxOccurs	AESSettingsCipherMode **MUST NOT** be set (maxOccurs=0) if ContentEncAlgo is not AES (5).

Table 29: AESSettingsCipherMode implementation notes

5.1.5. Cues Element

name / type / id: Cues / master / 0x1C53BB6B

path: \Segment\Cues

minOccurs - maxOccurs: see implementation notes - 1

definition: A Top-Level Element to speed seeking access. All entries are local to the Segment.

notes:

attribute	note
minOccurs	This Element **SHOULD** be set when the Segment is not transmitted as a live stream (see #livestreaming).

Table 30: Cues implementation notes

5.1.5.1. CuePoint Element

name / type / id: CuePoint / master / 0xBB

path: \Segment\Cues\CuePoint

minOccurs: 1

definition: Contains all information relative to a seek point in the Segment.

5.1.5.1.1. CueTime Element

name / type / id: CueTime / uinteger / 0xB3

path: \Segment\Cues\CuePoint\CueTime

minOccurs - maxOccurs: 1 - 1

definition: Absolute timestamp of the seek point, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1.

5.1.5.1.2. CueTrackPositions Element

name / type / id: CueTrackPositions / master / 0xB7

path: \Segment\Cues\CuePoint\CueTrackPositions

minOccurs: 1

definition: Contain positions for different tracks corresponding to the timestamp.

5.1.5.1.2.1. CueTrack Element

name / type / id: CueTrack / uinteger / 0xF7

path: \Segment\Cues\CuePoint\CueTrackPositions\CueTrack

minOccurs - maxOccurs: 1 - 1

range: not 0

definition: The track for which a position is given.

5.1.5.1.2.2. CueClusterPosition Element

name / type / id:
CueClusterPosition / uinteger / 0xF1

path: \Segment\Cues\CuePoint\CueTrackPositions\CueClusterPosition

minOccurs - maxOccurs: 1 - 1

definition: The Segment Position (Section 16) of the Cluster containing the associated Block.

5.1.5.1.2.3. CueRelativePosition Element

name / type / id: CueRelativePosition / uinteger / 0xF0

path: \Segment\Cues\CuePoint\CueTrackPositions\CueRelativePosition

maxOccurs: 1

minver: 4

definition: The relative position inside the Cluster of the referenced SimpleBlock or BlockGroup with 0 being the first possible position for an Element inside that Cluster.

5.1.5.1.2.4. CueDuration Element

name / type / id: CueDuration / uinteger / 0xB2

path: \Segment\Cues\CuePoint\CueTrackPositions\CueDuration

maxOccurs: 1

minver: 4

definition: The duration of the block, expressed in Segment Ticks which is based on TimestampScale; see Section 11.1. If missing, the track's DefaultDuration does not apply and no duration information is available in terms of the cues.

5.1.5.1.2.5. CueBlockNumber Element

name / type / id: CueBlockNumber / uinteger / 0x5378

path: \Segment\Cues\CuePoint\CueTrackPositions\CueBlockNumber

maxOccurs: 1

range: not 0

definition: Number of the Block in the specified Cluster.

5.1.5.1.2.6. CueCodecState Element

name / type / id: CueCodecState / uinteger / 0xEA

path: \Segment\Cues\CuePoint\CueTrackPositions\CueCodecState

minOccurs - maxOccurs: 1 - 1

default: 0

minver: 2

definition: The Segment Position (Section 16) of the Codec State corresponding to this Cue Element. 0 means that the data is taken from the initial Track Entry.

5.1.5.1.2.7. CueReference Element

name / type / id: CueReference / master / 0xDB

path: \Segment\Cues\CuePoint\CueTrackPositions\CueReference

minver: 2

definition: The Clusters containing the referenced Blocks.

5.1.5.1.2.8. CueRefTime Element

name / type / id: CueRefTime / uinteger / 0x96

path: \Segment\Cues\CuePoint\CueTrackPositions\CueReference\CueRefTime

minOccurs - maxOccurs: 1 - 1

minver: 2

definition: Timestamp of the referenced Block, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1.

5.1.6. Attachments Element

name / type / id: Attachments / master / 0x1941A469

path: \Segment\Attachments

maxOccurs: 1

definition: Contain attached files.

5.1.6.1. AttachedFile Element

name / type / id:
AttachedFile / master / 0x61A7

path: \Segment\Attachments\AttachedFile

minOccurs: 1

definition: An attached file.

5.1.6.1.1. FileDescription Element

name / type / id: FileDescription / utf-8 / 0x467E

path: \Segment\Attachments\AttachedFile\FileDescription

maxOccurs: 1

definition: A human-friendly name for the attached file.

5.1.6.1.2. FileName Element

name / type / id: FileName / utf-8 / 0x466E

path: \Segment\Attachments\AttachedFile\FileName

minOccurs - maxOccurs: 1 - 1

definition: Filename of the attached file.

5.1.6.1.3. FileMimeType Element

name / type / id: FileMimeType / string / 0x4660

path: \Segment\Attachments\AttachedFile\FileMimeType

minOccurs - maxOccurs: 1 - 1

stream copy: True (Section 8)

definition: MIME type of the file.

5.1.6.1.4. FileData Element

name / type / id: FileData / binary / 0x465C

path: \Segment\Attachments\AttachedFile\FileData

minOccurs - maxOccurs: 1 - 1

stream copy: True (Section 8)

definition:
The data of the file.

5.1.6.1.5. FileUID Element

name / type / id: FileUID / uinteger / 0x46AE

path: \Segment\Attachments\AttachedFile\FileUID

minOccurs - maxOccurs: 1 - 1

range: not 0

stream copy: True (Section 8)

definition: Unique ID representing the file, as random as possible.

5.1.7. Chapters Element

name / type / id: Chapters / master / 0x1043A770

path: \Segment\Chapters

maxOccurs: 1

recurring: True

definition: A system to define basic menus and partition data. For more detailed information, look at the Chapters explanation in Section 20.

5.1.7.1. EditionEntry Element

name / type / id: EditionEntry / master / 0x45B9

path: \Segment\Chapters\EditionEntry

minOccurs: 1

definition: Contains all information about a Segment edition.

5.1.7.1.1. EditionUID Element

name / type / id: EditionUID / uinteger / 0x45BC

path: \Segment\Chapters\EditionEntry\EditionUID

maxOccurs: 1

range: not 0

stream copy: True (Section 8)

definition: A unique ID to identify the edition. It's useful for tagging an edition.

5.1.7.1.2. EditionFlagDefault Element

name / type / id: EditionFlagDefault / uinteger / 0x45DB

path: \Segment\Chapters\EditionEntry\EditionFlagDefault

minOccurs - maxOccurs: 1 - 1

range: 0-1

default: 0

definition: Set to 1 if the edition **SHOULD** be used as the default one.

5.1.7.1.3. EditionFlagOrdered Element

name / type / id: EditionFlagOrdered / uinteger / 0x45DD

path: \Segment\Chapters\EditionEntry\EditionFlagOrdered

minOccurs - maxOccurs: 1 - 1

range: 0-1

default: 0

definition: Set to 1 if the chapters can be defined multiple times and the order to play them is enforced; see Section 20.1.3.

5.1.7.1.4. ChapterAtom Element

name / type / id: ChapterAtom / master / 0xB6

path: \Segment\Chapters\EditionEntry\+ChapterAtom

minOccurs: 1

recursive: True

definition: Contains the atom information to use as the chapter atom (apply to all tracks).

5.1.7.1.4.1. ChapterUID Element

name / type / id: ChapterUID / uinteger / 0x73C4

path:
\Segment\Chapters\EditionEntry\+ChapterAtom\ChapterUID

minOccurs - maxOccurs: 1 - 1

range: not 0

stream copy: True (Section 8)

definition: A unique ID to identify the Chapter.

5.1.7.1.4.2. ChapterStringUID Element

name / type / id: ChapterStringUID / utf-8 / 0x5654

path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterStringUID

maxOccurs: 1

minver: 3

definition: A unique string ID to identify the Chapter. Use for WebVTT cue identifier storage [WebVTT].

5.1.7.1.4.3. ChapterTimeStart Element

name / type / id: ChapterTimeStart / uinteger / 0x91

path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterTimeStart

minOccurs - maxOccurs: 1 - 1

definition: Timestamp of the start of Chapter, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1.

5.1.7.1.4.4. ChapterTimeEnd Element

name / type / id: ChapterTimeEnd / uinteger / 0x92

path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterTimeEnd

minOccurs - maxOccurs: see implementation notes - 1

definition: Timestamp of the end of Chapter timestamp excluded, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1. The value **MUST** be greater than or equal to the ChapterTimeStart of the same ChapterAtom.

usage notes: The ChapterTimeEnd timestamp value being excluded, it **MUST** take in account the duration of the last frame it includes,

especially for the ChapterAtom using the last frames of the Segment.

notes:

attribute	note
minOccurs	ChapterTimeEnd **MUST** be set (minOccurs=1) if the Edition is an ordered edition; see Section 20.1.3, unless it's a Parent Chapter; see Section 20.2.3

Table 31: ChapterTimeEnd implementation notes

5.1.7.1.4.5. ChapterFlagHidden Element

name / type / id: ChapterFlagHidden / uinteger / 0x98

path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterFlagHidden

minOccurs - maxOccurs: 1 - 1

range: 0-1

default: 0

definition: Set to 1 if a chapter is hidden. Hidden chapters **SHOULD NOT** be available to the user interface (but still to Control Tracks; see Section 20.2.5 on Chapter flags).

5.1.7.1.4.6. ChapterSegmentUUID Element

name / type / id: ChapterSegmentUUID / binary / 0x6E67

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapterSegmentUUID

minOccurs - maxOccurs: see implementation notes - 1

range: >0

definition: The SegmentUUID of another Segment to play during this chapter (128 bits). Like the SegmentUUID, it is a Universally Unique IDentifier stored in binary form [RFC4122].

usage notes: The value **MUST NOT** be the SegmentUUID value of the Segment it belongs to.

notes:

attribute	note
minOccurs	

attribute	note
	ChapterSegmentUUID **MUST** be set (minOccurs=1) if ChapterSegmentEditionUID is used; see Section 17.2 on medium-linking Segments.

Table 32: ChapterSegmentUUID implementation notes

5.1.7.1.4.7. ChapterSegmentEditionUID Element

name / type / id: ChapterSegmentEditionUID / uinteger / 0x6EBC

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapterSegmentEditionUID

maxOccurs: 1

range: not 0

definition: The EditionUID to play from the Segment linked in ChapterSegmentUUID. If ChapterSegmentEditionUID is undeclared, then no Edition of the linked Segment is used; see Section 17.2 on medium-linking Segments.

5.1.7.1.4.8. ChapterPhysicalEquiv Element

name / type / id: ChapterPhysicalEquiv / uinteger / 0x63C3

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapterPhysicalEquiv

maxOccurs: 1

definition: Specify the physical equivalent of this ChapterAtom like "DVD" (60) or "SIDE" (50); see Section 20.4 for a complete list of values.

5.1.7.1.4.9. ChapterDisplay Element

name / type / id: ChapterDisplay / master / 0x80

path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterDisplay

definition: Contains all possible strings to use for the chapter display.

5.1.7.1.4.10. ChapString Element

name / type / id: ChapString / utf-8 / 0x85

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapterDisplay\ChapString

minOccurs - maxOccurs: 1 - 1

definition:
Contains the string to use as the chapter atom.

5.1.7.1.4.11. ChapLanguage Element

name / type / id: ChapLanguage / string / 0x437C

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapterDisplay\ChapLanguage

minOccurs: 1

default: eng

definition: A language corresponding to the string, in the bibliographic ISO-639-2 form [ISO639-2]. This Element **MUST** be ignored if a ChapLanguageBCP47 Element is used within the same ChapterDisplay Element.

5.1.7.1.4.12. ChapLanguageBCP47 Element

name / type / id: ChapLanguageBCP47 / string / 0x437D

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapterDisplay\ChapLanguageBCP47

minver: 4

definition: Specifies a language corresponding to the ChapString in the format defined in [BCP47] and using the IANA Language Subtag Registry [IANALangRegistry]. If a ChapLanguageBCP47 Element is used, then any ChapLanguage and ChapCountry Elements used in the same ChapterDisplay **MUST** be ignored.

5.1.7.1.4.13. ChapCountry Element

name / type / id: ChapCountry / string / 0x437E

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapterDisplay\ChapCountry

definition: A country corresponding to the string, using the same 2 octets country-codes as in Internet domains [IANADomains] based on [ISO3166-1] alpha-2 codes. This Element **MUST** be ignored if a ChapLanguageBCP47 Element is used within the same ChapterDisplay Element.

5.1.7.1.4.14. ChapProcess Element

name / type / id: ChapProcess / master / 0x6944

path:
\Segment\Chapters\EditionEntry\+ChapterAtom\ChapProcess

definition: Contains all the commands associated to the Atom.

5.1.7.1.4.15. ChapProcessCodecID Element

name / type / id: ChapProcessCodecID / uinteger / 0x6955

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapProcess\ChapProcessCodecID

minOccurs - maxOccurs: 1 - 1

default: 0

definition: Contains the type of the codec used for the processing. A value of 0 means native Matroska processing (to be defined), a value of 1 means the DVD command set is used; see Section 20.3 on DVD menus. More codec IDs can be added later.

5.1.7.1.4.16. ChapProcessPrivate Element

name / type / id: ChapProcessPrivate / binary / 0x450D

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapProcess\ChapProcessPrivate

maxOccurs: 1

definition: Some optional data attached to the ChapProcessCodecID information. For ChapProcessCodecID = 1, it is the "DVD level" equivalent; see Section 20.3 on DVD menus.

5.1.7.1.4.17. ChapProcessCommand Element

name / type / id: ChapProcessCommand / master / 0x6911

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapProcess\ChapProcessCommand

definition: Contains all the commands associated to the Atom.

5.1.7.1.4.18. ChapProcessTime Element

name / type / id: ChapProcessTime / uinteger / 0x6922

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapProcess\ChapProcessCommand\ChapProcessTime

minOccurs - maxOccurs: 1 - 1

definition:
Defines when the process command **SHOULD** be handled

restrictions:

value	label
0	during the whole chapter
1	before starting playback
2	after playback of the chapter

Table 33: ChapProcessTime values

5.1.7.1.4.19. ChapProcessData Element

name / type / id: ChapProcessData / binary / 0x6933

path: \Segment\Chapters\EditionEntry\
+ChapterAtom\ChapProcess\ChapProcessCommand\ChapProcessData

minOccurs - maxOccurs: 1 - 1

definition: Contains the command information. The data **SHOULD** be interpreted depending on the ChapProcessCodecID value. For ChapProcessCodecID = 1, the data correspond to the binary DVD cell pre/post commands; see Section 20.3 on DVD menus.

5.1.8. Tags Element

name / type / id: Tags / master / 0x1254C367

path: \Segment\Tags

definition: Element containing metadata describing Tracks, Editions, Chapters, Attachments, or the Segment as a whole. A list of valid tags can be found in [MatroskaTags].

5.1.8.1. Tag Element

name / type / id: Tag / master / 0x7373

path: \Segment\Tags\Tag

minOccurs: 1

definition: A single metadata descriptor.

5.1.8.1.1. Targets Element

name / type / id: Targets / master / 0x63C0

path: \Segment\Tags\Tag\Targets

minOccurs - maxOccurs:
1 - 1

definition: Specifies which other elements the metadata represented by the Tag applies to. If empty or not present, then the Tag describes everything in the Segment.

5.1.8.1.1.1. TargetTypeValue Element

name / type / id: TargetTypeValue / uinteger / 0x68CA

path: \Segment\Tags\Tag\Targets\TargetTypeValue

minOccurs - maxOccurs: 1 - 1

default: 50

definition: A number to indicate the logical level of the target.

defined values:

value	label	definition
70	COLLECTION	The highest hierarchical level that tags can describe.
60	EDITION / ISSUE / VOLUME / OPUS / SEASON / SEQUEL	A list of lower levels grouped together.
50	ALBUM / OPERA / CONCERT / MOVIE / EPISODE	The most common grouping level of music and video (equals to an episode for TV series).
40	PART / SESSION	When an album or episode has different logical parts.
30	TRACK / SONG / CHAPTER	The common parts of an album or movie.
20	SUBTRACK / PART / MOVEMENT / SCENE	Corresponds to parts of a track for audio (like a movement).
10	SHOT	The lowest hierarchy found in music or movies.

Table 34: TargetTypeValue values

5.1.8.1.1.2. TargetType Element

name / type / id: TargetType / string / 0x63CA

path: \Segment\Tags\Tag\Targets\TargetType

maxOccurs: 1

definition: An informational string that can be used to display the logical level of the target like "ALBUM", "TRACK", "MOVIE", "CHAPTER", etc ; see Section 6.4 of [MatroskaTags].

restrictions:

value	label
COLLECTION	TargetTypeValue 70
EDITION	TargetTypeValue 60
ISSUE	TargetTypeValue 60
VOLUME	TargetTypeValue 60
OPUS	TargetTypeValue 60
SEASON	TargetTypeValue 60
SEQUEL	TargetTypeValue 60
ALBUM	TargetTypeValue 50
OPERA	TargetTypeValue 50
CONCERT	TargetTypeValue 50
MOVIE	TargetTypeValue 50
EPISODE	TargetTypeValue 50
PART	TargetTypeValue 40
SESSION	TargetTypeValue 40
TRACK	TargetTypeValue 30
SONG	TargetTypeValue 30
CHAPTER	TargetTypeValue 30
SUBTRACK	TargetTypeValue 20
PART	TargetTypeValue 20
MOVEMENT	TargetTypeValue 20
SCENE	TargetTypeValue 20
SHOT	TargetTypeValue 10

Table 35: TargetType values

5.1.8.1.1.3. TagTrackUID Element

name / type / id: TagTrackUID / uinteger / 0x63C5

path: \Segment\Tags\Tag\Targets\TagTrackUID

default: 0

definition: A unique ID to identify the Track(s) the tags belong to.

usage notes:
If the value is 0 at this level, the tags apply to all tracks in the Segment. If set to any other value, it **MUST** match the TrackUID value of a track found in this Segment.

5.1.8.1.1.4. TagEditionUID Element

name / type / id: TagEditionUID / uinteger / 0x63C9

path: \Segment\Tags\Tag\Targets\TagEditionUID

default: 0

definition: A unique ID to identify the EditionEntry(s) the tags belong to.

usage notes: If the value is 0 at this level, the tags apply to all editions in the Segment. If set to any other value, it **MUST** match the EditionUID value of an edition found in this Segment.

5.1.8.1.1.5. TagChapterUID Element

name / type / id: TagChapterUID / uinteger / 0x63C4

path: \Segment\Tags\Tag\Targets\TagChapterUID

default: 0

definition: A unique ID to identify the Chapter(s) the tags belong to.

usage notes: If the value is 0 at this level, the tags apply to all chapters in the Segment. If set to any other value, it **MUST** match the ChapterUID value of a chapter found in this Segment.

5.1.8.1.1.6. TagAttachmentUID Element

name / type / id: TagAttachmentUID / uinteger / 0x63C6

path: \Segment\Tags\Tag\Targets\TagAttachmentUID

default: 0

definition: A unique ID to identify the Attachment(s) the tags belong to.

usage notes: If the value is 0 at this level, the tags apply to all the attachments in the Segment. If set to any other value, it **MUST** match the FileUID value of an attachment found in this Segment.

5.1.8.1.2. SimpleTag Element

name / type / id: SimpleTag / master / 0x67C8

path: \Segment\Tags\Tag\+SimpleTag

minOccurs: 1

recursive: True

definition: Contains general information about the target.

5.1.8.1.2.1. TagName Element

name / type / id: TagName / utf-8 / 0x45A3

path: \Segment\Tags\Tag\+SimpleTag\TagName

minOccurs - maxOccurs: 1 - 1

definition: The name of the Tag that is going to be stored.

5.1.8.1.2.2. TagLanguage Element

name / type / id: TagLanguage / string / 0x447A

path: \Segment\Tags\Tag\+SimpleTag\TagLanguage

minOccurs - maxOccurs: 1 - 1

default: und

definition: Specifies the language of the tag specified, in the Matroska languages form; see Section 12 on language codes. This Element **MUST** be ignored if the TagLanguageBCP47 Element is used within the same SimpleTag Element.

5.1.8.1.2.3. TagLanguageBCP47 Element

name / type / id: TagLanguageBCP47 / string / 0x447B

path: \Segment\Tags\Tag\+SimpleTag\TagLanguageBCP47

maxOccurs: 1

minver: 4

definition: Specifies the language used in the TagString according to [BCP47] and using the IANA Language Subtag Registry [IANALangRegistry]. If this Element is used, then any TagLanguage Elements used in the same SimpleTag **MUST** be ignored.

5.1.8.1.2.4. TagDefault Element

name / type / id: TagDefault / uinteger / 0x4484

path: \Segment\Tags\Tag\+SimpleTag\TagDefault

minOccurs - maxOccurs: 1 - 1

range: 0-1

default: 1

definition: A boolean value to indicate if this is the default/ original language to use for the given tag.

5.1.8.1.2.5. TagString Element

name / type / id: TagString / utf-8 / 0x4487

path: \Segment\Tags\Tag\+SimpleTag\TagString

maxOccurs: 1

definition: The value of the Tag.

5.1.8.1.2.6. TagBinary Element

name / type / id: TagBinary / binary / 0x4485

path: \Segment\Tags\Tag\+SimpleTag\TagBinary

maxOccurs: 1

definition: The values of the Tag, if it is binary. Note that this cannot be used in the same SimpleTag as TagString.

6. Matroska Element Ordering

Except for the EBML Header and the CRC-32 Element, the EBML specification does not require any particular storage order for Elements. This specification however defines mandates and recommendations for ordering certain Elements in order to facilitate better playback, seeking, and editing efficiency. This section describes and offers rationale for ordering requirements and recommendations for Matroska.

6.1. Top-Level Elements

The Info Element is the only **REQUIRED** Top-Level Element in a Matroska file. To be playable, Matroska **MUST** also contain at least one Tracks Element and Cluster Element. The first Info Element and

the first Tracks Element **MUST** either be stored before the first Cluster Element or both **SHALL** be referenced by a SeekHead Element occurring before the first Cluster Element.

When using Medium Linking, chapters are used to reference other Segments to play in a given order Section 17.2. In that case the Segment containing in these Chapters do no required a Track Element or a Cluster Element.

It is possible to edit a Matroska file after it has been created. For example, chapters, tags, or attachments can be added. When new Top-Level Elements are added to a Matroska file, the SeekHead Element(s) **MUST** be updated so that the SeekHead Element(s) itemize the identity and position of all Top-Level Elements.

Editing, removing, or adding Elements to a Matroska file often requires that some existing Elements be voided or extended. Transforming the existing Elements into Void Elements as padding can be used as a method to avoid moving large amounts of data around.

6.2. CRC-32

As noted by the EBML specification, if a CRC-32 Element is used, then the CRC-32 Element **MUST** be the first ordered Element within its Parent Element.

In Matroska all Top-Level Elements of an EBML Document **SHOULD** include a CRC-32 Element as a their first Child Element. The Segment Element, which is the Root Element, **SHOULD NOT** have a CRC-32 Element.

6.3. SeekHead

If used, the first SeekHead Element **MUST** be the first non-CRC-32 Child Element of the Segment Element. If a second SeekHead Element is used, then the first SeekHead Element **MUST** reference the identity and position of the second SeekHead.

Additionally, the second SeekHead Element **MUST** only reference Cluster Elements and not any other Top-Level Element already contained within the first SeekHead Element.

The second SeekHead Element **MAY** be stored in any order relative to the other Top-Level Elements. Whether one or two SeekHead Element(s) are used, the SeekHead Element(s) **MUST** collectively reference the identity and position of all Top-Level Elements except for the first SeekHead Element.

6.4. Cues (index)

The Cues Element is **RECOMMENDED** to optimize seeking access in Matroska. It is programmatically simpler to add the Cues Element after all Cluster Elements have been written because this does not require a prediction of how much space to reserve before writing the Cluster Elements. However, storing the Cues Element before the Cluster Elements can provide some seeking advantages. If the Cues Element is present, then it **SHOULD** either be stored before the first Cluster Element or be referenced by a SeekHead Element.

6.5. Info

The first Info Element **SHOULD** occur before the first Tracks Element and first Cluster Element except when referenced by a SeekHead Element.

6.6. Chapters Element

The Chapters Element **SHOULD** be placed before the Cluster Element(s). The Chapters Element can be used during playback even if the user does not need to seek. It immediately gives the user information about what section is being read and what other sections are available. In the case of Ordered Chapters it is **RECOMMENDED** to evaluate the logical linking even before playing. The Chapters Element **SHOULD** be placed before the first Tracks Element and after the first Info Element.

6.7. Attachments

The Attachments Element is not intended to be used by default when playing the file, but could contain information relevant to the content, such as cover art or fonts. Cover art is useful even before the file is played and fonts could be needed before playback starts for initialization of subtitles. The Attachments Element **MAY** be placed before the first Cluster Element; however if the Attachments Element is likely to be edited, then it **SHOULD** be placed after the last Cluster Element.

6.8. Tags

The Tags Element is most subject to changes after the file was originally created. For easier editing, the Tags Element **SHOULD** be placed at the end of the Segment Element, even after the Attachments Element. On the other hand, it is inconvenient to have to seek in the Segment for tags, especially for network streams. So it's better if the Tags Element is found early in the stream. When editing the Tags Element, the original Tags Element at the beginning can be overwritten with a Void Element and a new Tags Element written at

the end of the Segment Element. The file size will only marginally change.

7. Matroska versioning

Matroska is based upon the principle that a reading application does not have to support 100% of the specifications in order to be able to play the file. A Matroska file therefore contains version indicators that tell a reading application what to expect.

It is possible and valid to have the version fields indicate that the file contains Matroska Elements from a higher specification version number while signaling that a reading application **MUST** only support a lower version number properly in order to play it back (possibly with a reduced feature set).

The EBML Header of each Matroska document informs the reading application on what version of Matroska to expect. The Elements within EBML Header with jurisdiction over this information are DocTypeVersion and DocTypeReadVersion.

DocTypeVersion **MUST** be equal to or greater than the highest Matroska version number of any Element present in the Matroska file. For example, a file using the SimpleBlock Element (Section 5.1.3.3) **MUST** have a DocTypeVersion equal to or greater than 2. A file containing CueRelativePosition Elements (Section 5.1.5.1.2.3) **MUST** have a DocTypeVersion equal to or greater than 4.

The DocTypeReadVersion **MUST** contain the minimum version number that a reading application can minimally support in order to play the file back -- optionally with a reduced feature set. For example, if a file contains only Elements of version 2 or lower except for CueRelativePosition (which is a version 4 Matroska Element), then DocTypeReadVersion **SHOULD** still be set to 2 and not 4 because evaluating CueRelativePosition is not necessary for standard playback -- it makes seeking more precise if used.

A reading application supporting Matroska version V **MUST NOT** refuse to read an application with DocReadTypeVersion equal to or lower than V even if DocTypeVersion is greater than V.

A reading application supporting at least Matroska version V reading a file whose DocTypeReadVersion field is equal to or lower than V **MUST** skip Matroska/EBML Elements it encounters but does not know about if that unknown element fits into the size constraints set by the current Parent Element.

8. Stream Copy

It is sometimes necessary to create a Matroska file from another Matroska file, for example to add subtitles in a language or to edit out a portion of the content. Some values from the original Matroska file needs to be kept the same in the destination file. For example the SamplingFrequency of an audio track wouldn't change between the two files. Some other values may change between the two files, for example the TrackNumber of an audio track when another track has been added.

Elements are marked with a property: stream copy: True when the values need to be kept between the source and destination file. If that property is not set, elements may or may not keep the same value between the source and destination.

9. DefaultDecodedFieldDuration

The DefaultDecodedFieldDuration Element can signal to the displaying application how often fields of a video sequence will be available for displaying. It can be used for both interlaced and progressive content. If the video sequence is signaled as interlaced, then the period between two successive fields at the output of the decoding process equals DefaultDecodedFieldDuration.

For video sequences signaled as progressive, it is twice the value of DefaultDecodedFieldDuration.

These values are valid at the end of the decoding process before post-processing (such as deinterlacing or inverse telecine) is applied.

Examples:

*Blu-ray movie: 1000000000ns/(48/1.001) = 20854167ns

*PAL broadcast/DVD: 1000000000ns/(50/1.000) = 20000000ns

*N/ATSC broadcast: 1000000000ns/(60/1.001) = 16683333ns

*hard-telecined DVD: 1000000000ns/(60/1.001) = 16683333ns (60 encoded interlaced fields per second)

*soft-telecined DVD: 1000000000ns/(60/1.001) = 16683333ns (48 encoded interlaced fields per second, with "repeat*first*field = 1")

10. Block Structure

Bit 0 is the most significant bit.

Frames using references **SHOULD** be stored in "coding order". That means the references first, and then the frames referencing them. A consequence is that timestamps might not be consecutive. But a frame with a past timestamp **MUST** reference a frame already known, otherwise it's considered bad/void.

10.1. Block Header

Offset	Player	Description
0x00+	**MUST**	Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits set to increase the range).
0x01+	**MUST**	Timestamp (relative to Cluster timestamp, signed int16)

Table 36: Block Header base parts

10.2. Block Header Flags

Offset	Bit	Player	Description
0x03+	0-3	-	Reserved, set to 0
0x03+	4	-	Invisible, the codec **SHOULD** decode this frame but not display it
0x03+	5-6	**MUST**	Lacing
			* 00 : no lacing
			* 01 : Xiph lacing
			* 11 : EBML lacing
			* 10 : fixed-size lacing
0x03+	7	-	not used

Table 37: Block Header flags part

10.3. SimpleBlock Structure

The SimpleBlock is inspired by the Block structure; see Section 10. The main differences are the added Keyframe flag and Discardable flag. Otherwise everything is the same.

Bit 0 is the most significant bit.

Frames using references **SHOULD** be stored in "coding order". That means the references first, and then the frames referencing them. A consequence is that timestamps might not be consecutive. But a frame with a past timestamp **MUST** reference a frame already known, otherwise it's considered bad/void.

10.3.1. SimpleBlock Header

Offset	Player	Description
0x00+	**MUST**	Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits set to increase the range).
0x01+	**MUST**	Timestamp (relative to Cluster timestamp, signed int16)

Table 38: SimpleBlock Header base parts

10.3.2. SimpleBlock Header Flags

Offset	Bit	Player	Description
0x03+	0	-	Keyframe, set when the Block contains only keyframes
0x03+	1-3	-	Reserved, set to 0
0x03+	4	-	Invisible, the codec **SHOULD** decode this frame but not display it
0x03+	5-6	**MUST**	Lacing
			* 00 : no lacing
			* 01 : Xiph lacing
			* 11 : EBML lacing
			* 10 : fixed-size lacing
0x03+	7	-	Discardable, the frames of the Block can be discarded during playing if needed

Table 39: SimpleBlock Header flags part

10.4. Block Lacing

Lacing is a mechanism to save space when storing data. It is typically used for small blocks of data (referred to as frames in Matroska). It packs multiple frames into a single Block or SimpleBlock.

Lacing **MUST NOT** be used to store a single frame in a Block or SimpleBlock.

There are 3 types of lacing:

1. Xiph, inspired by what is found in the Ogg container [RFC3533]

2. EBML, which is the same with sizes coded differently

3. fixed-size, where the size is not coded

When lacing is not used, i.e. to store a single frame, the lacing bits 5 and 6 of the Block or SimpleBlock **MUST** be set to zero.

For example, a user wants to store 3 frames of the same track. The first frame is 800 octets long, the second is 500 octets long and the third is 1000 octets long. As these data are small, they can be stored in a lace to save space.

It is possible not to use lacing at all and just store a single frame without any extra data. When the FlagLacing -- Section 5.1.4.1.12 -- is set to "0" all blocks of that track **MUST NOT** use lacing.

10.4.1. No lacing

When no lacing is used, the number of frames in the lace is ommitted and only one frame can be stored in the Block. The bits 5-6 of the Block Header flags are set to 00.

The Block for a 800 octets frame is as follows:

Block Octets	Value	Description
4-803	<frame>	Single frame data

Table 40: No lacing

When a Block contains a single frame, it **MUST** use this No lacing mode.

10.4.2. Xiph lacing

The Xiph lacing uses the same coding of size as found in the Ogg container [RFC3533]. The bits 5-6 of the Block Header flags are set to 01.

The Block data with laced frames is stored as follows:

- *Lacing Head on 1 Octet: Number of frames in the lace minus 1.
- *Lacing size of each frame except the last one.
- *Binary data of each frame consecutively.

The lacing size is split into 255 values, stored as unsigned octets -- for example, 500 is coded 255;245 or [0xFF 0xF5]. A frame with a size multiple of 255 is coded with a 0 at the end of the size -- for example, 765 is coded 255;255;255;0 or [0xFF 0xFF 0xFF 0x00].

The size of the last frame is deduced from the size remaining in the Block after the other frames.

Because large sizes result in large coding of the sizes, it is **RECOMMENDED** to use Xiph lacing only with small frames.

In our example, the 800, 500 and 1000 frames are stored with Xiph lacing in a Block as follows:

Block Octet	Value	Description
4	0x02	Number of frames minus 1
5-8	0xFF 0xFF 0xFF 0x23	Size of the first frame (255;255;255;35)
9-10	0xFF 0xF5	Size of the second frame (255;245)
11-810		First frame data
811-1310		Second frame data
1311-2310		Third frame data

Table 41: Xiph lacing example

The Block is 2311 octets large and the last frame starts at 1311, so we can deduce the size of the last frame is 2311 - 1311 = 1000.

10.4.3. EBML lacing

The EBML lacing encodes the frame size with an EBML-like encoding [RFC8794]. The bits 5-6 of the Block Header flags are set to 11.

The Block data with laced frames is stored as follows:

*Lacing Head on 1 Octet: Number of frames in the lace minus 1.

*Lacing size of each frame except the last one.

*Binary data of each frame consecutively.

The first frame size is encoded as an EBML Unsigned Integer Element value. The other frame sizes are encoded as a difference with the previous frame size as EBML Signed Integer Element values. That corresponds to an EBML Data Size Values with two's complement notation with the leftmost bit being the sign bit as found in [RFC8794], giving this range of values:

Bit Representation	Value
1xxx xxxx	value $-(2^6-1)$ to 2^6-1 (ie 0 to 2^7-2 minus 2^6-1, half of the range)
01xx xxxx xxxx xxxx	value $-(2^{13}-1)$ to $2^{13}-1$
001x xxxx xxxx xxxx xxxx xxxx	value $-(2^{20}-1)$ to $2^{20}-1$
0001 xxxx xxxx xxxx xxxx xxxx xxxx xxxx	value $-(2^{27}-1)$ to $2^{27}-1$
0000 1xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	value $-(2^{34}-1)$ to $2^{34}-1$

Bit Representation	Value
0000 01xx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	value -(2^{41}-1) to 2^{41}-1
0000 001x xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	value -(2^{48}-1) to 2^{48}-1

Table 42: EBML Lacing bits usage

In our example, the 800, 500 and 1000 frames are stored with EBML lacing in a Block as follows:

Block Octets	Value	Description
4	0x02	Number of frames minus 1
5-6	0x43 0x20	Size of the first frame (800 = 0x320 + 0x4000)
7-8	0x5E 0xD3	Size of the second frame (500 - 800 = -300 = -0x12C + 0x1FFF + 0x4000)
8-807	<frame1>	First frame data
808-1307	<frame2>	Second frame data
1308-2307	<frame3>	Third frame data

Table 43: EBML lacing example

The Block is 2308 octets large and the last frame starts at 1308, so we can deduce the size of the last frame is 2308 - 1308 = 1000.

10.4.4. Fixed-size lacing

The Fixed-size lacing doesn't store the frame size, only the number of frames in the lace. Each frame **MUST** have the same size. The frame size of each frame is deduced from the total size of the Block. The bits 5-6 of the Block Header flags are set to 10.

The Block data with laced frames is stored as follows:

- Lacing Head on 1 Octet: Number of frames in the lace minus 1.
- Binary data of each frame consecutively.

For example, for 3 frames of 800 octets each:

Block Octets	Value	Description
4	0x02	Number of frames minus 1
5-804	<frame1>	First frame data
805-1604	<frame2>	Second frame data
1605-2404	<frame3>	Third frame data

Table 44: Fixed-size lacing example

This gives a Block of 2405 octets. When reading the Block we find that there are 3 frames (Octet 4). The data start at Octet 5, so the size of each frame is (2405 - 5) / 3 = 800.

10.4.5. Laced Frames Timestamp

A Block only contains a single timestamp value. But when lacing is used, it contains more than one frame. Each frame originally has its own timestamp, or Presentation Timestamp (PTS). That timestamp applies to the first frame in the lace.

In the lace, each frame after the first one has an underdetermined timestamp. But each of these frames **MUST** be contiguous -- i.e. the decoded data **MUST NOT** contain any gap between them. If there is a gap in the stream, the frames around the gap **MUST NOT** be in the same Block.

Lacing is only useful for small contiguous data to save space. This is usually the case for audio tracks and not the case for video -- which use a lot of data -- or subtitle tracks -- which have long gaps. For audio, there is usually a fixed output sampling frequency for the whole track. So the decoder should be able to recover the timestamp of each sample, knowing each output sample is contiguous with a fixed frequency. For subtitles this is usually not the case so lacing **SHOULD NOT** be used.

10.5. Random Access Points

Random Access Points (RAP) are positions where the parser can seek to and start playback without decoding of what was before. In Matroska BlockGroups and SimpleBlocks can be RAPs. To seek to these elements it is still necessary to seek to the Cluster containing them, read the Cluster Timestamp and start playback from the BlockGroup or SimpleBlock that is a RAP.

Because a Matroska File is usually composed of multiple tracks playing at the same time -- video, audio and subtitles -- to seek properly to a RAP, each selected track must be taken in account. Usually all audio and subtitle BlockGroup or SimpleBlock are RAP. They are independent of each other and can be played randomly.

Video tracks on the other hand often use references to previous and future frames for better coding efficiency. Frames with such reference **MUST** either contain one or more ReferenceBlock Elements in their BlockGroup or **MUST** be marked as non-keyframe in a SimpleBlock; see Section 10.3.2.

*BlockGroup with a frame that references another frame, with the EBML tree shown as XML:

```
<Cluster>
  <Timestamp>123456</Timestamp>
  <BlockGroup>
    <!-- References a Block 40 Track Ticks before this one -->
    <ReferenceBlock>-40</ReferenceBlock>
    <Block/>
  </BlockGroup>
  ...
</Cluster>
```

*SimpleBlock with a frame that references another frame, with the EBML tree shown as XML:

```
<Cluster>
  <Timestamp>123456</Timestamp>
  <SimpleBlock/> (octet 3 bit 0 not set)
  ...
</Cluster>
```

Frames that are RAP -- i.e. they don't depend on other frames -- **MUST** set the keyframe flag if they are in a SimpleBlock or their parent BlockGroup **MUST NOT** contain a ReferenceBlock.

*BlockGroup with a frame that references no other frame, with the EBML tree shown as XML:

```
<Cluster>
  <Timestamp>123456</Timestamp>
  <BlockGroup>
    <!-- No ReferenceBlock allowed in this BlockGroup -->
    <Block/>
  </BlockGroup>
  ...
</Cluster>
```

*SimpleBlock with a frame that references no other frame, with the EBML tree shown as XML:

```
<Cluster>
  <Timestamp>123456</Timestamp>
  <SimpleBlock/> (octet 3 bit 0 set)
  ...
</Cluster>
```

There may be cases where the use of BlockGroup is necessary, as the frame may need a BlockDuration, BlockAdditions, CodecState or a DiscardPadding element. For thoses cases a SimpleBlock **MUST NOT** be used, the reference information **SHOULD** be recovered for non-RAP frames.

*SimpleBlock with a frame that references another frame, with the EBML tree shown as XML:

```
<Cluster>
  <Timestamp>123456</Timestamp>
  <SimpleBlock/> (octet 3 bit 0 not set)
  ...
</Cluster>
```

*Same frame that references another frame put inside a BlockGroup to add BlockDuration, with the EBML tree shown as XML:

```
<Cluster>
  <Timestamp>123456</Timestamp>
  <BlockGroup>
    <!-- ReferenceBlock value recovered based on the codec -->
    <ReferenceBlock>-40</ReferenceBlock>
    <BlockDuration>20<BlockDuration>
    <Block/>
  </BlockGroup>
  ...
</Cluster>
```

When a frame in a BlockGroup is not a RAP, all references **SHOULD** be listed as a ReferenceBlock, at least some of them, even if not accurate, or one ReferenceBlock with the value "0" corresponding to a self or unknown reference. The lack of ReferenceBlock would mean such a frame is a RAP and seeking on that frame that actually depends on other frames **MAY** create bogus output or even crash.

*Same frame that references another frame put inside a BlockGroup but the reference could not be recovered, with the EBML tree shown as XML:

```
<Cluster>
  <Timestamp>123456</Timestamp>
  <BlockGroup>
    <!-- ReferenceBlock value not recovered from the codec -->
    <ReferenceBlock>0</ReferenceBlock>
    <BlockDuration>20<BlockDuration>
    <Block/>
  </BlockGroup>
  ...
</Cluster>
```

*BlockGroup with a frame that references two other frames, with the EBML tree shown as XML:

```
<Cluster>
  <Timestamp>123456</Timestamp>
  <BlockGroup>
    <!-- References a Block 80 Track Ticks before this one -->
    <ReferenceBlock>-80</ReferenceBlock>
    <!-- References a Block 40 Track Ticks after this one -->
    <ReferenceBlock>40</ReferenceBlock>
    <Block/>
  </BlockGroup>
  ...
</Cluster>
```

Intra-only video frames, such as the ones found in AV1 or VP9, can be decoded without any other frame, but they don't reset the codec state. So seeking to these frames is not possible as the next frames may need frames that are not known from this seeking point. Such intra-only frames **MUST NOT** be considered as keyframes so the keyframe flag **MUST NOT** be set in the SimpleBlock or a ReferenceBlock **MUST** be used to signify the frame is not a RAP. The timestamp value of the ReferenceBlock **MUST** be "0", meaning it's referencing itself.

*Intra-only frame not an RAP, with the EBML tree shown as XML:

```
<Cluster>
  <Timestamp>123456</Timestamp>
  <BlockGroup>
    <!-- References itself to mark it should not be used as RAP -->
    <ReferenceBlock>0</ReferenceBlock>
    <Block/>
  </BlockGroup>
  ...
</Cluster>
```

Because a video SimpleBlock has less references information than a video BlockGroup, it is possible to remux a video track using BlockGroup into a SimpleBlock, as long as it doesn't use any other BlockGroup features than ReferenceBlock.

11. Timestamps

Historically timestamps in Matroska were mistakenly called timecodes. The Timestamp Element was called Timecode, the TimestampScale Element was called TimecodeScale, the TrackTimestampScale Element was called TrackTimecodeScale and the ReferenceTimestamp Element was called ReferenceTimeCode.

11.1. Timestamp Ticks

All timestamp values in Matroska are expressed in multiples of a tick. They are usually stored as integers. There are three types of ticks possible:

11.1.1. Matroska Ticks

For such elements, the timestamp value is stored directly in nanoseconds.

The elements storing values in Matroska Ticks/nanoseconds are:

*TrackEntry\DefaultDuration; defined in Section 5.1.4.1.15

*TrackEntry\DefaultDecodedFieldDuration; defined in Section 5.1.4.1.16

*TrackEntry\SeekPreRoll; defined in Section 5.1.4.1.29

*TrackEntry\CodecDelay; defined in Section 5.1.4.1.28

*BlockGroup\DiscardPadding; defined in Section 5.1.3.4.7

*ChapterAtom\ChapterTimeStart; defined in Section 5.1.7.1.4.3

*ChapterAtom\ChapterTimeEnd; defined in Section 5.1.7.1.4.4

*CuePoint\CueTime; defined in Section 5.1.5.1.1

*CueReference\CueRefTime; defined in Section 5.1.5.1.1

11.1.2. Segment Ticks

Elements in Segment Ticks involve the use of the TimestampScale Element of the Segment to get the timestamp in nanoseconds of the element, with the following formula:

```
timestamp in nanosecond = element value * TimestampScale
```

This allows storing smaller integer values in the elements.

When using the default value of TimestampScale of "1,000,000", one Segment Tick represents one millisecond.

The elements storing values in Segment Ticks are:

*Cluster\Timestamp; defined in Section 5.1.3.1

*Info\Duration is stored as a floating point but the same formula applies; defined in Section 5.1.2.10

*CuePoint\CueTrackPositions\CueDuration; defined in Section 5.1.5.1.2.4

11.1.3. Track Ticks

Elements in Track Ticks involve the use of the TimestampScale Element of the Segment and the TrackTimestampScale Element of the Track to get the timestamp in nanoseconds of the element, with the following formula:

```
timestamp in nanoseconds =
    element value * TrackTimestampScale * TimestampScale
```

This allows storing smaller integer values in the elements. The resulting floating point values of the timestamps are still expressed in nanoseconds.

When using the default values for TimestampScale and TrackTimestampScale of "1,000,000" and of "1.0" respectively, one Track Tick represents one millisecond.

The elements storing values in Track Ticks are:

- Cluster\BlockGroup\Block and Cluster\SimpleBlock timestamps; detailed in Section 11.2

- Cluster\BlockGroup\BlockDuration; defined in Section 5.1.3.4.3

- Cluster\BlockGroup\ReferenceBlock; defined in Section 5.1.3.4.5

When the TrackTimestampScale is interpreted as "1.0", Track Ticks are equivalent to Segment Ticks and give an integer value in nanoseconds. This is the most common case as TrackTimestampScale is usually omitted.

A value of TrackTimestampScale other than "1.0" **MAY** be used to scale the timestamps more in tune with each Track sampling frequency. For historical reasons, a lot of Matroska readers don't take the TrackTimestampScale value in account. So using a value other than "1.0" **MAY** not work in many places.

11.2. Block Timestamps

A Block Element and SimpleBlock Element timestamp is the time when the decoded data of the first frame in the Block/SimpleBlock **MUST** be presented, if the track of that Block/SimpleBlock is selected for playback. This is also known as the Presentation Timestamp (PTS).

The Block Element and SimpleBlock Element store their timestamps as signed integers, relative to the Cluster\Timestamp value of the Cluster they are stored in. To get the timestamp of a Block or SimpleBlock in nanoseconds you have to use the following formula:

```
( Cluster\Timestamp + ( block timestamp * TrackTimestampScale ) ) *
TimestampScale
```

The Block Element and SimpleBlock Element store their timestamps as 16bit signed integers, allowing a range from "-32768" to "+32767" Track Ticks. Although these values can be negative, when added to the Cluster\Timestamp, the resulting frame timestamp **SHOULD NOT** be negative.

When a CodecDelay Element is set, its value **MUST** be substracted from each Block timestamp of that track. To get the timestamp in nanoseconds of the first frame in a Block or SimpleBlock, the formula becomes:

```
( ( Cluster\Timestamp + ( block timestamp * TrackTimestampScale ) ) *
  TimestampScale ) - CodecDelay
```

The resulting frame timestamp **SHOULD NOT** be negative.

During playback, when a frame has a negative timestamp, the content **MUST** be decoded by the decoder but not played to the user.

11.3. TimestampScale Rounding

The default Track Tick duration is one millisecond.

The TimestampScale is a floating value, which is usually 1.0. But when it's not, the multiplied Block Timestamp is a floating values in nanoseconds. The Matroska Reader **SHOULD** use the nearest rounding value in nanosecond to get the proper nanosecond timestamp of a Block. This allows some clever TimestampScale values to have more refined timestampt precision per frame.

12. Language Codes

Matroska from version 1 through 3 uses language codes that can be either the 3 letters bibliographic ISO-639-2 form [ISO639-2] (like "fre" for french), or such a language code followed by a dash and a country code for specialities in languages (like "fre-ca" for Canadian French). The ISO 639-2 Language Elements are "Language Element", "TagLanguage Element", and "ChapLanguage Element".

Starting in Matroska version 4, either [ISO639-2] or [BCP47] **MAY** be used, although BCP 47 is **RECOMMENDED**. The BCP 47 Language Elements are "LanguageBCP47 Element", "TagLanguageBCP47 Element", and "ChapLanguageBCP47 Element". If a BCP 47 Language Element and an ISO 639-2 Language Element are used within the same Parent Element, then the ISO 639-2 Language Element **MUST** be ignored and precedence given to the BCP 47 Language Element.

Country codes are the same 2 octets country-codes as in Internet domains [IANADomains] based on [ISO3166-1] alpha-2 codes.

13. Encryption

Encryption in Matroska is designed in a very generic style to allow people to implement whatever form of encryption is best for them. It is possible to use the encryption framework in Matroska as a type of DRM (Digital Rights Management).

Because encryption occurs within the Block Element, it is possible to manipulate encrypted streams without decrypting them. The streams could potentially be copied, deleted, cut, appended, or any number of other possible editing techniques without decryption. The data can be used without having to expose it or go through the decrypting process.

Encryption can also be layered within Matroska. This means that two completely different types of encryption can be used, requiring two separate keys to be able to decrypt a stream.

Encryption information is stored in the ContentEncodings Element under the ContentEncryption Element.

For encryption systems sharing public/private keys, the creation of the keys and the exchange of keys are not covered by this document. They have to be handled by the system using Matroska.

The ContentEncodingScope Element gives an idea of which part of the track are encrypted. But each ContentEncAlgo Element and its sub elements like AESSettingsCipherMode really define how the encrypted should be exactly interpreted.

The AES-CTR system, which corresponds to ContentEncAlgo = 5 (Section 5.1.4.1.34.9) and AESSettingsCipherMode = 1 (Section 5.1.4.1.34.12), is defined in the [WebM-Enc] document.

14. Image Presentation

14.1. Cropping

The PixelCrop Elements (PixelCropTop, PixelCropBottom, PixelCropRight, and PixelCropLeft) indicate when, and by how much, encoded videos frames **SHOULD** be cropped for display. These Elements allow edges of the frame that are not intended for display, such as the sprockets of a full-frame film scan or the VANC area of a digitized analog videotape, to be stored but hidden. PixelCropTop and PixelCropBottom store an integer of how many rows of pixels **SHOULD** be cropped from the top and bottom of the image (respectively). PixelCropLeft and PixelCropRight store an integer of how many columns of pixels **SHOULD** be cropped from the left and right of the image (respectively). For example, a pillar-boxed video that stores a 1440x1080 visual image within the center of a padded 1920x1080 encoded image **MAY** set both PixelCropLeft and PixelCropRight to "240", so that a Matroska Player **SHOULD** crop off 240 columns of pixels from the left and right of the encoded image to present the image with the pillar-boxes hidden.

Cropping has to be performed before resizing and the display dimensions given by DisplayWidth, DisplayHeight and DisplayUnit apply to the already cropped image.

14.2. Rotation

The ProjectionPoseRoll Element (see Section 5.1.4.1.31.46) can be used to indicate that the image from the associated video track **SHOULD** be rotated for presentation. For instance, the following

representation of the Projection Element Section 5.1.4.1.31.41) and the ProjectionPoseRoll Element represents a video track where the image **SHOULD** be presented with a 90 degree counter-clockwise rotation, with the EBML tree shown as XML :

```
<Projection>
  <ProjectionPoseRoll>90</ProjectionPoseRoll>
</Projection>
```

Figure 11: Rotation example.

15. File Extensions

The file extensions for Matroska files are usually as follows:

*".mkv" for files containing video tracks

*".mka" for files containing audio tracks with no video tracks

*".mk3d" for files containing some stereoscopic video tracks

16. Segment Position

The Segment Position of an Element refers to the position of the first octet of the Element ID of that Element, measured in octets, from the beginning of the Element Data section of the containing Segment Element. In other words, the Segment Position of an Element is the distance in octets from the beginning of its containing Segment Element minus the size of the Element ID and Element Data Size of that Segment Element. The Segment Position of the first Child Element of the Segment Element is 0. An Element which is not stored within a Segment Element, such as the Elements of the EBML Header, do not have a Segment Position.

16.1. Segment Position Exception

Elements that are defined to store a Segment Position **MAY** define reserved values to indicate a special meaning.

16.2. Example of Segment Position

This table presents an example of Segment Position by showing a hexadecimal representation of a very small Matroska file with labels to show the offsets in octets. The file contains a Segment Element with an Element ID of "0x18538067" and a MuxingApp Element with an Element ID of "0x4D80".

```
   0                           1                           2
   0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  0
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 0 |1A|45|DF|A3|8B|42|82|88|6D|61|74|72|6F|73|6B|61|
   ^ EBML Header
 0 |                                                 |18|53|80|67|
                                                     ^ Segment ID
20 |93|
   ^ Segment Data Size
20 |  |15|49|A9|66|8E|4D|80|84|69|65|74|66|57|41|84|69|65|74|66|
      ^ Start of Segment data
20 |                 |4D|80|84|69|65|74|66|57|41|84|69|65|74|66|
                     ^ MuxingApp start
```

In the above example, the Element ID of the Segment Element is stored at offset 16, the Element Data Size of the Segment Element is stored at offset 20, and the Element Data of the Segment Element is stored at offset 21.

The MuxingApp Element is stored at offset 26. Since the Segment Position of an Element is calculated by subtracting the position of the Element Data of the containing Segment Element from the position of that Element, the Segment Position of MuxingApp Element in the above example is '26 - 21' or '5'.

17. Linked Segments

Matroska provides several methods to link two or more Segment Elements together to create a Linked Segment. A Linked Segment is a set of multiple Segments linked together into a single presentation by using Hard Linking or Medium Linking.

All Segments within a Linked Segment **MUST** have a SegmentUUID.

All Segments within a Linked Segment **SHOULD** be stored within the same directory or be accessible quickly based on their SegmentUUID in order to have seamless transition between segments.

All Segments within a Linked Segment **MAY** set a SegmentFamily with a common value to make it easier for a Matroska Player to know which Segments are meant to be played together.

The SegmentFilename, PrevFilename and NextFilename elements **MAY** also give hints on the original filenames that were used when the Segment links were created, in case some SegmentUUID are damaged.

17.1. Hard Linking

Hard Linking, also called splitting, is the process of creating a Linked Segment by linking multiple Segment Elements using the NextUUID and PrevUUID Elements.

All Segments within a Hard Linked Segment **MUST** use the same Tracks list and TimestampScale.

Within a Linked Segment, the timestamps of Block and SimpleBlock **MUST** follow consecutively the timestamps of Block and SimpleBlock from the previous Segment in linking order.

With Hard Linking, the chapters of any Segment within the Linked Segment **MUST** only reference the current Segment. The NextUUID and PrevUUID reference the respective SegmentUUID values of the next and previous Segments.

The first Segment of a Linked Segment **MUST NOT** have a PrevUUID Element. The last Segment of a Linked Segment **MUST NOT** have a NextUUID Element.

For each node of the chain of Segments of a Linked Segment at least one Segment **MUST** reference the other Segment of the node.

In a chain of Segments of a Linked Segment the NextUUID always takes precedence over the PrevUUID. So if SegmentA has a NextUUID to SegmentB and SegmentB has a PrevUUID to SegmentC, the link to use is NextUUID between SegmentA and SegmentB, SegmentC is not part of the Linked Segment.

If SegmentB has a PrevUUID to SegmentA but SegmentA has no NextUUID, then the Matroska Player **MAY** consider these two Segments linked as SegmentA followed by SegmentB.

As an example, three Segments can be Hard Linked as a Linked Segment through cross-referencing each other with SegmentUUID, PrevUUID, and NextUUID, as in this table:

file name	SegmentUUID	PrevUUID	NextUUID
start.mkv	71000c23cd310998 53fbc94dd984a5dd	Invalid	a77b3598941cb803 eac0fcdafe44fac9
middle.mkv	a77b3598941cb803 eac0fcdafe44fac9	71000c23cd310998 53fbc94dd984a5dd	6c92285fa6d3e827 b198d120ea3ac674
end.mkv	6c92285fa6d3e827 b198d120ea3ac674	a77b3598941cb803 eac0fcdafe44fac9	Invalid

Table 45: Usual Hard Linking UIDs

An other example where only the NextUUID Element is used:

file name	SegmentUUID	PrevUUID	NextUUID
start.mkv	71000c23cd310998 53fbc94dd984a5dd	Invalid	a77b3598941cb803 eac0fcdafe44fac9
middle.mkv	a77b3598941cb803 eac0fcdafe44fac9	n/a	6c92285fa6d3e827 b198d120ea3ac674
end.mkv	6c92285fa6d3e827 b198d120ea3ac674	n/a	Invalid

Table 46: Hard Linking without PrevUUID

An example where only the PrevUUID Element is used:

file name	SegmentUUID	PrevUUID	NextUUID
start.mkv	71000c23cd310998 53fbc94dd984a5dd	Invalid	n/a
middle.mkv	a77b3598941cb803 eac0fcdafe44fac9	71000c23cd310998 53fbc94dd984a5dd	n/a
end.mkv	6c92285fa6d3e827 b198d120ea3ac674	a77b3598941cb803 eac0fcdafe44fac9	Invalid

Table 47: Hard Linking without NextUUID

In this example only the middle.mkv is using the PrevUUID and NextUUID Elements:

file name	SegmentUUID	PrevUUID	NextUUID
start.mkv	71000c23cd310998 53fbc94dd984a5dd	Invalid	n/a
middle.mkv	a77b3598941cb803 eac0fcdafe44fac9	71000c23cd310998 53fbc94dd984a5dd	6c92285fa6d3e827 b198d120ea3ac674
end.mkv	6c92285fa6d3e827 b198d120ea3ac674	n/a	Invalid

Table 48: Hard Linking with mixed UID links

17.2. Medium Linking

Medium Linking creates relationships between Segments using Ordered Chapters (Section 20.1.3) and the ChapterSegmentUUID Element. A Chapter Edition with Ordered Chapters **MAY** contain Chapter elements that reference timestamp ranges from other Segments. The Segment referenced by the Ordered Chapter via the ChapterSegmentUUID Element **SHOULD** be played as part of a Linked Segment.

The timestamps of Segment content referenced by Ordered Chapters **MUST** be adjusted according to the cumulative duration of the the previous Ordered Chapters.

As an example a file named intro.mkv could have a SegmentUUID of "0xb16a58609fc7e60653a60c984fc11ead". Another file called program.mkv could use a Chapter Edition that contains two Ordered Chapters. The first chapter references the Segment of intro.mkv with the use of a ChapterSegmentUUID, ChapterSegmentEditionUID, ChapterTimeStart, and optionally a ChapterTimeEnd element. The second chapter references content within the Segment of program.mkv. A Matroska Player **SHOULD** recognize the Linked Segment created by the use of ChapterSegmentUUID in an enabled Edition and present the reference content of the two Segments as a single presentation.

The ChapterSegmentUUID represents the Segment that holds the content to play in place of the Linked Chapter. The ChapterSegmentUUID **MUST NOT** be the SegmentUUID of its own Segment.

There are 2 ways to use a chapter link: * Linked-Duration linking, * Linked-Edition linking

17.2.1. Linked-Duration

A Matroska Player **MUST** play the content of the linked Segment from the ChapterTimeStart until ChapterTimeEnd timestamp in place of the Linked Chapter.

ChapterTimeStart and ChapterTimeEnd represent timestamps in the Linked Segment matching the value of ChapterSegmentUUID. Their values **MUST** be in the range of the linked Segment duration.

The ChapterTimeEnd value **MUST** be set when using linked-duration chapter linking. ChapterSegmentEditionUID **MUST NOT** be set.

17.2.2. Linked-Edition

A Matroska Player **MUST** play the whole linked Edition of the linked Segment in place of the Linked Chapter.

ChapterSegmentEditionUID represents a valid Edition from the Linked Segment matching the value of ChapterSegmentUUID.

When using linked-edition chapter linking. ChapterTimeEnd is **OPTIONAL**.

18. Track Flags

18.1. Default flag

The "default track" flag is a hint for a Matroska Player indicating that a given track **SHOULD** be eligible to be automatically selected as the default track for a given language. If no tracks in a given language have the default track flag set, then all tracks in that

language are eligible for automatic selection. This can be used to indicate that a track provides "regular service" suitable for users with default settings, as opposed to specialized services, such as commentary, hearing-impaired captions, or descriptive audio.

The Matroska Player **MAY** override the "default track" flag for any reason, including user preferences to prefer tracks providing accessibility services.

18.2. Forced flag

The "forced" flag tells the Matroska Player that it **SHOULD** display this subtitle track, even if user preferences usually would not call for any subtitles to be displayed alongside the current selected audio track. This can be used to indicate that a track contains translations of onscreen text, or of dialogue spoken in a different language than the track's primary one.

18.3. Hearing-impaired flag

The "hearing impaired" flag tells the Matroska Player that it **SHOULD** prefer this track when selecting a default track for a hearing-impaired user, and that it **MAY** prefer to select a different track when selecting a default track for a non-hearing-impaired user.

18.4. Visual-impaired flag

The "visual impaired" flag tells the Matroska Player that it **SHOULD** prefer this track when selecting a default track for a visually-impaired user, and that it **MAY** prefer to select a different track when selecting a default track for a non-visually-impaired user.

18.5. Descriptions flag

The "descriptions" flag tells the Matroska Player that this track is suitable to play via a text-to-speech system for a visually-impaired user, and that it **SHOULD NOT** automatically select this track when selecting a default track for a non-visually-impaired user.

18.6. Original flag

The "original" flag tells the Matroska Player that this track is in the original language, and that it **SHOULD** prefer it if configured to prefer original-language tracks of this track's type.

18.7. Commentary flag

The "commentary" flag tells the Matroska Player that this track contains commentary on the content.

18.8. Track Operation

TrackOperation allows combining multiple tracks to make a virtual one. It uses two separate system to combine tracks. One to create a 3D "composition" (left/right/background planes) and one to simplify join two tracks together to make a single track.

A track created with TrackOperation is a proper track with a UID and all its flags. However the codec ID is meaningless because each "sub" track needs to be decoded by its own decoder before the "operation" is applied. The Cues Elements corresponding to such a virtual track **SHOULD** be the sum of the Cues Elements for each of the tracks it's composed of (when the Cues are defined per track).

In the case of TrackJoinBlocks, the Block Elements (from BlockGroup and SimpleBlock) of all the tracks **SHOULD** be used as if they were defined for this new virtual Track. When two Block Elements have overlapping start or end timestamps, it's up to the underlying system to either drop some of these frames or render them the way they overlap. This situation **SHOULD** be avoided when creating such tracks as you can never be sure of the end result on different platforms.

18.9. Overlay Track

Overlay tracks **SHOULD** be rendered in the same channel as the track its linked to. When content is found in such a track, it **SHOULD** be played on the rendering channel instead of the original track.

18.10. Multi-planar and 3D videos

There are two different ways to compress 3D videos: have each eye track in a separate track and have one track have both eyes combined inside (which is more efficient, compression-wise). Matroska supports both ways.

For the single track variant, there is the StereoMode Element, which defines how planes are assembled in the track (mono or left-right combined). Odd values of StereoMode means the left plane comes first for more convenient reading. The pixel count of the track (PixelWidth/PixelHeight) is the raw amount of pixels, for example 3840x1080 for full HD side by side, and the DisplayWidth/ DisplayHeight in pixels is the amount of pixels for one plane (1920x1080 for that full HD stream). Old stereo 3D were displayed using anaglyph (cyan and red colors separated). For compatibility with such movies, there is a value of the StereoMode that corresponds to AnaGlyph.

There is also a "packed" mode (values 13 and 14) which consists of packing two frames together in a Block using lacing. The first frame

is the left eye and the other frame is the right eye (or vice versa). The frames **SHOULD** be decoded in that order and are possibly dependent on each other (P and B frames).

For separate tracks, Matroska needs to define exactly which track does what. TrackOperation with TrackCombinePlanes do that. For more details look at Section 18.8 on how TrackOperation works.

The 3D support is still in infancy and may evolve to support more features.

The StereoMode used to be part of Matroska v2 but it didn't meet the requirement for multiple tracks. There was also a bug in libmatroska prior to 0.9.0 that would save/read it as 0x53B9 instead of 0x53B8; see OldStereoMode (Section 5.1.4.1.31.5). Matroska Readers may support these legacy files by checking Matroska v2 or 0x53B9. The older values of StereoMode were 0: mono, 1: right eye, 2: left eye, 3: both eyes, the only values that can be found in OldStereoMode. They are not compatible with the StereoMode values found in Matroska v3 and above.

19. Default track selection

This section provides some example sets of Tracks and hypothetical user settings, along with indications of which ones a similarly-configured Matroska Player **SHOULD** automatically select for playback by default in such a situation. A player **MAY** provide additional settings with more detailed controls for more nuanced scenarios. These examples are provided as guidelines to illustrate the intended usages of the various supported Track flags, and their expected behaviors.

Track names are shown in English for illustrative purposes; actual files may have titles in the language of each track, or provide titles in multiple languages.

19.1. Audio Selection

Example track set:

No.	Type	Lang	Layout	Original	Default	Other flags	Name
1	Video	und	N/A	N/A	N/A	None	
2	Audio	eng	5.1	1	1	None	
3	Audio	eng	2.0	1	1	None	
4	Audio	eng	2.0	1	0	Visual-impaired	Descriptive audio
5	Audio	esp	5.1	0	1	None	

No.	Type	Lang	Layout	Original	Default	Other flags	Name
6	Audio	esp	2.0	0	0	Visual-impaired	Descriptive audio
7	Audio	eng	2.0	1	0	Commentary	Director's Commentary
8	Audio	eng	2.0	1	0	None	Karaoke

Table 49: Audio Tracks for default selection

Here we have a file with 7 audio tracks, of which 5 are in English and 2 are in Spanish.

The English tracks all have the Original flag, indicating that English is the original content language.

Generally the player will first consider the track languages: if the player has an option to prefer original-language audio and the user has enabled it, then it should prefer one of the Original-flagged tracks. If configured to specifically prefer audio tracks in English or Spanish, the player should select one of the tracks in the corresponding language. The player may also wish to prefer an Original-flagged track if no tracks matching any of the user's explicitly-preferred languages are available.

Two of the tracks have the Visual-impaired flag. If the player has been configured to prefer such tracks, it should select one; otherwise, it should avoid them if possible.

If selecting an English track, when other settings have left multiple possible options, it may be useful to exclude the tracks that lack the Default flag: here, one provides descriptive service for the visually impaired (which has its own flag and may be automatically selected by user configuration, but is unsuitable for users with default-configured players), one is a commentary track (which has its own flag, which the player may or may not have specialized handling for), and the last contains karaoke versions of the music that plays during the film, which is an unusual specialized audio service that Matroska has no built-in support for indicating, so it's indicated in the track name instead. By not setting the Default flag on these specialized tracks, the file's author hints that they should not be automatically selected by a default-configured player.

Having narrowed its choices down, our example player now may have to select between tracks 2 and 3. The only difference between these tracks is their channel layouts: 2 is 5.1 surround, while 3 is stereo. If the player is aware that the output device is a pair of headphones or stereo speakers, it may wish to prefer the stereo mix

automatically. On the other hand, if it knows that the device is a surround system, it may wish to prefer the surround mix.

If the player finishes analyzing all of the available audio tracks and finds that multiple seem equally and maximally preferable, it **SHOULD** default to the first of the group.

19.2. Subtitle selection

Example track set:

No.	Type	Lang	Original	Default	Forced	Other flags	Name
1	Video	und	N/A	N/A	N/A	None	
2	Audio	fra	1	1	N/A	None	
3	Audio	por	0	1	N/A	None	
4	Subtitles	fra	1	1	0	None	
5	Subtitles	fra	1	0	0	Hearing-impaired	Captions for the hearing-impaired
6	Subtitles	por	0	1	0	None	
7	Subtitles	por	0	0	1	None	Signs
8	Subtitles	por	0	0	0	Hearing-impaired	SDH

Table 50: Subtitle Tracks for default selection

Here we have 2 audio tracks and 5 subtitle tracks. As we can see, French is the original language.

We'll start by discussing the case where the user prefers French (or Original-language) audio (or has explicitly selected the French audio track), and also prefers French subtitles.

In this case, if the player isn't configured to display captions when the audio matches their preferred subtitle languages, the player doesn't need to select a subtitle track at all.

If the user *has* indicated that they want captions to be displayed, the selection simply comes down to whether Hearing-impaired subtitles are preferred.

The situation for a user who prefers Portuguese subtitles starts out somewhat analogous. If they select the original French audio (either by explicit audio language preference, preference for Original-language tracks, or by explicitly selecting that track), then the selection once again comes down to the hearing-impaired preference.

However, the case where the Portuguese audio track is selected has an important catch: a Forced track in Portuguese is present. This may contain translations of onscreen text from the video track, or of portions of the audio that are not translated (music, for instance). This means that even if the user's preferences wouldn't normally call for captions here, the Forced track should be selected nonetheless, rather than selecting no track at all. On the other hand, if the user's preferences *do* call for captions, the non-Forced tracks should be preferred, as the Forced track will not contain captioning for the dialogue.

20. Chapters

The Matroska Chapters system can have multiple Editions and each Edition can consist of Simple Chapters where a chapter start time is used as marker in the timeline only. An Edition can be more complex with Ordered Chapters where a chapter end time stamp is additionally used or much more complex with Linked Chapters. The Matroska Chapters system can also have a menu structure, borrowed from the DVD menu system [DVD-Video], or have it's own Native Matroska menu structure.

20.1. EditionEntry

The EditionEntry is also called an Edition. An Edition contains a set of Edition flags and **MUST** contain at least one ChapterAtom Element. Chapters are always inside an Edition (or a Chapter itself part of an Edition). Multiple Editions are allowed. Some of these Editions **MAY** be ordered and others not.

20.1.1. EditionFlagDefault

Only one Edition **SHOULD** have an EditionFlagDefault flag set to true.

20.1.2. Default Edition

The Default Edition is the Edition that a Matroska Player **SHOULD** use for playback by default.

The first Edition with the EditionFlagDefault flag set to true is the Default Edition.

When all EditionFlagDefault flags are set to false, then the first Edition is the Default Edition.

Edition	FlagDefault	Default Edition
Edition 1	true	X
Edition 2	true	
Edition 3	true	

Table 51: Default edition, all default

Edition	FlagDefault	Default Edition
Edition 1	false	X
Edition 2	false	
Edition 3	false	

Table 52: Default edition, no default

Edition	FlagDefault	Default Edition
Edition 1	false	
Edition 2	true	X
Edition 3	false	

Table 53: Default edition, with default

20.1.3. EditionFlagOrdered

The EditionFlagOrdered Flag is a significant feature as it enables an Edition of Ordered Chapters which defines and arranges a virtual timeline rather than simply labeling points within the timeline. For example, with Editions of Ordered Chapters a single Matroska file can present multiple edits of a film without duplicating content. Alternatively, if a videotape is digitized in full, one Ordered Edition could present the full content (including colorbars, countdown, slate, a feature presentation, and black frames), while another Edition of Ordered Chapters can use Chapters that only mark the intended presentation with the colorbars and other ancillary visual information excluded. If an Edition of Ordered Chapters is enabled, then the Matroska Player **MUST** play those Chapters in their stored order from the timestamp marked in the ChapterTimeStart Element to the timestamp marked in to ChapterTimeEnd Element.

If the EditionFlagOrdered Flag evaluates to "0", Simple Chapters are used and only the ChapterTimeStart of a Chapter is used as chapter mark to jump to the predefined point in the timeline. With Simple Chapters, a Matroska Player **MUST** ignore certain Chapter Elements. In that case these elements are informational only.

The following list shows the different Chapter elements only found in Ordered Chapters.

Ordered Chapter elements
ChapterAtom/ChapterSegmentUUID
ChapterAtom/ChapterSegmentEditionUID
ChapterAtom/ChapterTrack
ChapterAtom/ChapProcess
Info/ChapterTranslate
TrackEntry/TrackTranslate

Table 54: elements only found in ordered chapters

Furthermore there are other EBML Elements which could be used if the EditionFlagOrdered evaluates to "1".

20.1.3.1. Ordered-Edition and Matroska Segment-Linking

*Hard Linking: Ordered-Chapters supersedes the Hard Linking.

*Medium Linking: Ordered Chapters are used in a normal way and can be combined with the ChapterSegmentUUID element which establishes a link to another Segment.

See Section 17 on the Linked Segments for more information about Hard Linking and Medium Linking.

20.2. ChapterAtom

The ChapterAtom is also called a Chapter.

20.2.1. ChapterTimeStart

The timestamp of the start of Chapter with nanosecond accuracy, not scaled by TimestampScale. For Simple Chapters this is the position of the chapter markers in the timeline.

20.2.2. ChapterTimeEnd

The timestamp of the end of Chapter with nanosecond accuracy, not scaled by TimestampScale. The timestamp defined by the ChapterTimeEnd is not part of the Chapter. A Matroska Player calculates the duration of this Chapter using the difference between the ChapterTimeEnd and ChapterTimeStart. The end timestamp **MUST** be greater than or equal to the start timestamp.

When the ChapterTimeEnd timestamp is equal to the ChapterTimeStart timestamp, the timestamps is included in the Chapter. It can be useful to put markers in a file or add chapter commands with ordered chapter commands without having to play anything; see Section 5.1.7.1.4.14.

Chapter	Start timestamp	End timestamp	Duration
Chapter 1	0	1000000000	1000000000
Chapter 2	1000000000	5000000000	4000000000
Chapter 3	6000000000	6000000000	0
Chapter 4	9000000000	8000000000	Invalid (-1000000000)

Table 55: ChapterTimeEnd usage possibilities

20.2.3. Nested Chapters

A ChapterAtom element can contain other ChapterAtom elements. That element is a Parent Chapter and the ChapterAtom elements it contains are Nested Chapters.

Nested Chapters can be useful to tag small parts of a Segment that already have tags or add Chapter Codec commands on smaller parts of a Segment that already have Chapter Codec commands.

The ChapterTimeStart of a Nested Chapter **MUST** be greater than or equal to the ChapterTimeStart its Parent Chapter.

If the Parent Chapter of a Nested Chapter has a ChapterTimeEnd, the ChapterTimeStart of that Nested Chapter **MUST** be smaller than or equal to the ChapterTimeEnd of the Parent Chapter.

20.2.4. Nested Chapters in Ordered Chapters

The ChapterTimeEnd of the lowest level of Nested Chapters **MUST** be set for Ordered Chapters.

When used with Ordered Chapters, the ChapterTimeEnd value of a Parent Chapter is useless for playback as the proper playback sections are described in its Nested Chapters. The ChapterTimeEnd **SHOULD NOT** be set in Parent Chapters and **MUST** be ignored for playback.

20.2.5. ChapterFlagHidden

Each Chapter ChapterFlagHidden flag works independently from parent chapters. A Nested Chapter with a ChapterFlagHidden that evaluates to "0" remains visible in the user interface even if the Parent Chapter ChapterFlagHidden flag is set to "1".

Chapter + Nested Chapter	ChapterFlagHidden	visible
Chapter 1	0	yes
Nested Chapter 1.1	0	yes
Nested Chapter 1.2	1	no
Chapter 2	1	no
Nested Chapter 2.1	0	yes
Nested Chapter 2.2	1	no

Table 56: ChapterFlagHidden nested visibility

20.3. Menu features

The menu features are handled like a chapter codec. That means each codec has a type, some private data and some data in the chapters.

The type of the menu system is defined by the ChapProcessCodecID parameter. For now, only 2 values are supported : 0 matroska script, 1 menu borrowed from the DVD [DVD-Video]. The private data depend on the type of menu system (stored in ChapProcessPrivate), idem for the data in the chapters (stored in ChapProcessData).

The menu system, as well as Chapter Codecs in general, can do actions on the Matroska Player like jumping to another Chapter or Edition, selecting different tracks and possibly more. The scope of all the possibilities of Chapter Codecs is not covered in this document as it depends on the Chapter Codec features and its integration in a Matroska Player.

20.4. Physical Types

Each level can have different meanings for audio and video. The ORIGINAL_MEDIUM tag can be used to specify a string for ChapterPhysicalEquiv = 60. Here is the list of possible levels for both audio and video:

Value	Audio	Video	Comment
70	SET / PACKAGE	SET / PACKAGE	the collection of different media
60	CD / 12" / 10" / 7" / TAPE / MINIDISC / DAT	DVD / VHS / LASERDISC	the physical medium like a CD or a DVD
50	SIDE	SIDE	when the original medium (LP/DVD) has different sides
40	-	LAYER	another physical level on DVDs
30	SESSION	SESSION	as found on CDs and DVDs
20	TRACK	-	as found on audio CDs
10	INDEX	-	the first logical level of the side/medium

Table 57: ChapterPhysicalEquiv meaning per track type

20.5. Chapter Examples

20.5.1. Example 1 : basic chaptering

In this example a movie is split in different chapters. It could also just be an audio file (album) on which each track corresponds to a chapter.

*00000ms - 05000ms : Intro

*05000ms - 25000ms : Before the crime

*25000ms - 27500ms : The crime

*27500ms - 38000ms : The killer arrested

*38000ms - 43000ms : Credits

This would translate in the following matroska form, with the EBML tree shown as XML :

```
<Chapters>
  <EditionEntry>
    <EditionUID>16603393396715046047</EditionUID>
    <ChapterAtom>
      <ChapterUID>1193046</ChapterUID>
      <ChapterTimeStart>0</ChapterTimeStart>
      <ChapterTimeEnd>5000000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>Intro</ChapString>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>2311527</ChapterUID>
      <ChapterTimeStart>5000000000</ChapterTimeStart>
      <ChapterTimeEnd>25000000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>Before the crime</ChapString>
      </ChapterDisplay>
      <ChapterDisplay>
        <ChapString>Avant le crime</ChapString>
        <ChapLanguage>fra</ChapLanguage>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>3430008</ChapterUID>
      <ChapterTimeStart>25000000000</ChapterTimeStart>
      <ChapterTimeEnd>27500000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>The crime</ChapString>
      </ChapterDisplay>
      <ChapterDisplay>
        <ChapString>Le crime</ChapString>
        <ChapLanguage>fra</ChapLanguage>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>4548489</ChapterUID>
      <ChapterTimeStart>27500000000</ChapterTimeStart>
      <ChapterTimeEnd>38000000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>After the crime</ChapString>
      </ChapterDisplay>
      <ChapterDisplay>
        <ChapString>Après le crime</ChapString>
        <ChapLanguage>fra</ChapLanguage>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>5666960</ChapterUID>
```

```
      <ChapterTimeStart>38000000000</ChapterTimeStart>
      <ChapterTimeEnd>43000000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>Credits</ChapString>
      </ChapterDisplay>
      <ChapterDisplay>
        <ChapString>Générique</ChapString>
        <ChapLanguage>fra</ChapLanguage>
      </ChapterDisplay>
    </ChapterAtom>
  </EditionEntry>
</Chapters>
```

Figure 12: Basic Chapters Example.

20.5.2. Example 2 : nested chapters

In this example an (existing) album is split into different chapters, and one of them contain another splitting.

20.5.2.1. The Micronauts "Bleep To Bleep"

*00:00 - 12:28 : Baby Wants To Bleep/Rock

-00:00 - 04:38 : Baby wants to bleep (pt.1)

-04:38 - 07:12 : Baby wants to rock

-07:12 - 10:33 : Baby wants to bleep (pt.2)

-10:33 - 12:28 : Baby wants to bleep (pt.3)

*12:30 - 19:38 : Bleeper_O+2

*19:40 - 22:20 : Baby wants to bleep (pt.4)

*22:22 - 25:18 : Bleep to bleep

*25:20 - 33:35 : Baby wants to bleep (k)

*33:37 - 44:28 : Bleeper

This would translate in the following matroska form, with the EBML tree shown as XML :

```
<Chapters>
  <EditionEntry>
    <EditionUID>1281690858003401414</EditionUID>
    <ChapterAtom>
      <ChapterUID>1</ChapterUID>
      <ChapterTimeStart>0</ChapterTimeStart>
      <ChapterTimeEnd>748000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>Baby wants to Bleep/Rock</ChapString>
      </ChapterDisplay>
      <ChapterAtom>
        <ChapterUID>2</ChapterUID>
        <ChapterTimeStart>0</ChapterTimeStart>
        <ChapterTimeEnd>278000000</ChapterTimeEnd>
        <ChapterDisplay>
          <ChapString>Baby wants to bleep (pt.1)</ChapString>
        </ChapterDisplay>
      </ChapterAtom>
      <ChapterAtom>
        <ChapterUID>3</ChapterUID>
        <ChapterTimeStart>278000000</ChapterTimeStart>
        <ChapterTimeEnd>432000000</ChapterTimeEnd>
        <ChapterDisplay>
          <ChapString>Baby wants to rock</ChapString>
        </ChapterDisplay>
      </ChapterAtom>
      <ChapterAtom>
        <ChapterUID>4</ChapterUID>
        <ChapterTimeStart>432000000</ChapterTimeStart>
        <ChapterTimeEnd>633000000</ChapterTimeEnd>
        <ChapterDisplay>
          <ChapString>Baby wants to bleep (pt.2)</ChapString>
        </ChapterDisplay>
      </ChapterAtom>
      <ChapterAtom>
        <ChapterUID>5</ChapterUID>
        <ChapterTimeStart>633000000</ChapterTimeStart>
        <ChapterTimeEnd>748000000</ChapterTimeEnd>
        <ChapterDisplay>
          <ChapString>Baby wants to bleep (pt.3)</ChapString>
        </ChapterDisplay>
      </ChapterAtom>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>6</ChapterUID>
      <ChapterTimeStart>750000000</ChapterTimeStart>
      <ChapterTimeEnd>1178500000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>Bleeper_O+2</ChapString>
```

```
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>7</ChapterUID>
      <ChapterTimeStart>1180500000</ChapterTimeStart>
      <ChapterTimeEnd>1340000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>Baby wants to bleep (pt.4)</ChapString>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>8</ChapterUID>
      <ChapterTimeStart>1342000000</ChapterTimeStart>
      <ChapterTimeEnd>1518000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>Bleep to bleep</ChapString>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>9</ChapterUID>
      <ChapterTimeStart>1520000000</ChapterTimeStart>
      <ChapterTimeEnd>2015000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>Baby wants to bleep (k)</ChapString>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>10</ChapterUID>
      <ChapterTimeStart>2017000000</ChapterTimeStart>
      <ChapterTimeEnd>2668000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChapString>Bleeper</ChapString>
      </ChapterDisplay>
    </ChapterAtom>
  </EditionEntry>
</Chapters>
```

Figure 13: Nested Chapters Example.

21. Attachments

Matroska supports storage of related files and data in the Attachments Element (a Top-Level Element). Attachment Elements can be used to store related cover art, font files, transcripts, reports, error recovery files, picture, or text-based annotations, copies of specifications, or other ancillary files related to the Segment.

Matroska Readers **MUST NOT** execute files stored as Attachment Elements.

21.1. Cover Art

This section defines a set of guidelines for the storage of cover art in Matroska files. A Matroska Reader **MAY** use embedded cover art to display a representational still-image depiction of the multimedia contents of the Matroska file.

Only JPEG and PNG image formats **SHOULD** be used for cover art pictures.

There can be two different covers for a movie/album: a portrait style (e.g., a DVD case) and a landscape style (e.g., a wide banner ad).

There can be two versions of the same cover, the normal cover and the small cover. The dimension of the normal cover **SHOULD** be 600 pixels on the smallest side -- for example, 960x600 for landscape, 600x800 for portrait, or 600x600 for square. The dimension of the small cover **SHOULD** be 120 pixels on the smallest side -- for example, 192x120 or 120x160.

Versions of cover art can be differentiated by the filename, which is stored in the FileName Element. The default filename of the normal cover in square or portrait mode is cover.(jpg|png). When stored, the normal cover **SHOULD** be the first Attachment in storage order. The small cover **SHOULD** be prefixed with "small_", such as small_cover.(jpg|png). The landscape variant **SHOULD** be suffixed with "_land", such as cover_land.(jpg|png). The filenames are case sensitive.

The following table provides examples of file names for cover art in Attachments.

FileName	Image Orientation	Pixel Length of Smallest Side
cover.jpg	Portrait or square	600
small_cover.png	Portrait or square	120
cover_land.png	Landscape	600
small*cover*land.jpg	Landscape	120

Table 58: Cover Art Filenames

21.2. Font files

Font files **MAY** be added to a Matroska file as Attachments so that the font file may be used to display an associated subtitle track. This allows the presentation of a Matroska file to be consistent in

various environments where the needed fonts might not be available on the local system.

Depending on the font format in question, each font file can contain multiple font variants. Each font variant has a name which will be referred to as Font Name from now on. This Font Name can be different than the Attachment's FileName, even when disregarding the extension. In order to select a font for display, a Matroska player **SHOULD** consider both the Font Name and the base name of the Attachment's FileName, preferring the former when there are multiple matches.

Subtitle codecs, such as SubStation Alpha (SSA/ASS), usually refer to a font by its Font Name, not by its filename. If none of the Attachments are a match for the Font Name, the Matroska player **SHOULD** attempt to find a system font whose Font Name matches the one used in the subtitle track.

Since loading fonts temporarily can take a while, a Matroska player usually loads or installs all the fonts found in attachments so they are ready to be used during playback. Failure to use the font attachment might result in incorrect rendering of the subtitles.

If a selected subtitle track has some AttachmentLink elements, the player **MAY** use only these fonts.

A Matroska player **SHOULD** handle the official font MIME types from [RFC8081] when the system can handle the type: * font/sfnt: Generic SFNT Font Type, * font/ttf: TTF Font Type, * font/otf: OpenType Layout (OTF) Font Type, * font/collection: Collection Font Type, * font/woff: WOFF 1.0, * font/woff2: WOFF 2.0.

Fonts in Matroska existed long before [RFC8081]. A few unofficial MIME types for fonts were used in existing files. Therefore it is **RECOMMENDED** for a Matroska player to support the following legacy MIME types for font attachments:

*application/x-truetype-font: Truetype fonts, equivalent to font/ttf and sometimes font/otf,

*application/x-font-ttf: TTF fonts, equivalent to font/ttf,

*application/vnd.ms-opentype: OpenType Layout fonts, equivalent to font/otf

*application/font-sfnt: Generic SFNT Font Type, equivalent to font/sfnt

*application/font-woff: WOFF 1.0, equivalent to font/woff

There may also be some font attachments with the application/octet-stream MIME type. In that case the Matroska player **MAY** try to guess the font type by checking the file extension of the AttachedFile\FileName string. Common file extensions for fonts are: * .ttf for Truetype fonts, equivalent to font/ttf, * .otf for OpenType Layout fonts, equivalent to font/otf, * .ttc for Collection fonts, equivalent to font/collection The file extension check **MUST** be case insensitive.

Matroska writers **SHOULD** use a valid font MIME type from [RFC8081] in the AttachedFile\FileMimeType of the font attachment. They **MAY** use the MIME types found in older files when compatibility with older players is necessary.

22. Cues

The Cues Element provides an index of certain Cluster Elements to allow for optimized seeking to absolute timestamps within the Segment. The Cues Element contains one or many CuePoint Elements which each **MUST** reference an absolute timestamp (via the CueTime Element), a Track (via the CueTrack Element), and a Segment Position (via the CueClusterPosition Element). Additional non-mandated Elements are part of the CuePoint Element such as CueDuration, CueRelativePosition, CueCodecState and others which provide any Matroska Reader with additional information to use in the optimization of seeking performance.

22.1. Recommendations

The following recommendations are provided to optimize Matroska performance.

*Unless Matroska is used as a live stream, it **SHOULD** contain a Cues Element.

*For each video track, each keyframe **SHOULD** be referenced by a CuePoint Element.

*It is **RECOMMENDED** to not reference non-keyframes of video tracks in Cues unless it references a Cluster Element which contains a CodecState Element but no keyframes.

*For each subtitle track present, each subtitle frame **SHOULD** be referenced by a CuePoint Element with a CueDuration Element.

*References to audio tracks **MAY** be skipped in CuePoint Elements if a video track is present. When included the CuePoint Elements **SHOULD** reference audio keyframes at most once every 500 milliseconds.

* If the referenced frame is not stored within the first SimpleBlock, or first BlockGroup within its Cluster Element, then the CueRelativePosition Element **SHOULD** be written to reference where in the Cluster the reference frame is stored.

* If a CuePoint Element references Cluster Element that includes a CodecState Element, then that CuePoint Element **MUST** use a CueCodecState Element.

* CuePoint Elements **SHOULD** be numerically sorted in storage order by the value of the CueTime Element.

23. Matroska Streaming

In Matroska, there are two kinds of streaming: file access and livestreaming.

23.1. File Access

File access can simply be reading a file located on your computer, but also includes accessing a file from an HTTP (web) server or CIFS (Windows share) server. These protocols are usually safe from reading errors and seeking in the stream is possible. However, when a file is stored far away or on a slow server, seeking can be an expensive operation and **SHOULD** be avoided. The following guidelines, when followed, help reduce the number of seeking operations for regular playback and also have the playback start quickly without a lot of data needed to read first (like a Cues Element, Attachment Element or SeekHead Element).

Matroska, having a small overhead, is well suited for storing music/videos on file servers without a big impact on the bandwidth used. Matroska does not require the index to be loaded before playing, which allows playback to start very quickly. The index can be loaded only when seeking is requested the first time.

23.2. Livestreaming

Livestreaming is the equivalent of television broadcasting on the internet. There are 2 families of servers for livestreaming: RTP/RTSP and HTTP. Matroska is not meant to be used over RTP. RTP already has timing and channel mechanisms that would be wasted if doubled in Matroska. Additionally, having the same information at the RTP and Matroska level would be a source of confusion if they do not match. Livestreaming of Matroska over HTTP (or any other plain protocol based on TCP) is possible.

A live Matroska stream is different from a file because it usually has no known end (only ending when the client disconnects). For this, all bits of the "size" portion of the Segment Element **MUST** be

set to 1. Another option is to concatenate Segment Elements with known sizes, one after the other. This solution allows a change of codec/resolution between each segment. For example, this allows for a switch between 4:3 and 16:9 in a television program.

When Segment Elements are continuous, certain Elements, like SeekHead, Cues, Chapters, and Attachments, **MUST NOT** be used.

It is possible for a Matroska Player to detect that a stream is not seekable. If the stream has neither a SeekHead list or a Cues list at the beginning of the stream, it **SHOULD** be considered non-seekable. Even though it is possible to seek blindly forward in the stream, it is **NOT RECOMMENDED**.

In the context of live radio or web TV, it is possible to "tag" the content while it is playing. The Tags Element can be placed between Clusters each time it is necessary. In that case, the new Tags Element **MUST** reset the previously encountered Tags Elements and use the new values instead.

24. Implementation Recommendations

24.1. Cluster

It is **RECOMMENDED** that the size of each individual Cluster Element be limited to store no more than 5 seconds or 5 megabytes.

24.2. SeekHead

It is **RECOMMENDED** that the first SeekHead Element be followed by a Void Element to allow for the SeekHead Element to be expanded to cover new Top-Level Elements that could be added to the Matroska file, such as Tags, Chapters, and Attachments Elements.

The size of this Void Element should be adjusted depending whether the Matroska file already has Tags, Chapters, and Attachments Elements.

24.3. Cues

For video files, it is **RECOMMENDED** to index at least the keyframes of the video track.

24.4. Optimum Layouts

While there can be Top-Level Elements in any order, some ordering of Elements are better than others. Here are few optimum layouts for different use case:

24.4.1. Optimum layout for a muxer

This is the basic layout muxers should be using for an efficient playback experience.

- *SeekHead
- *Info
- *Tracks
- *Chapters
- *Attachments
- *Tags
- *Clusters
- *Cues

24.4.2. Optimum layout after editing tags

When tags from the previous layout need to be extended, they are moved to the end with the extra information. The location where the old tags were located is voided.

- *SeekHead
- *Info
- *Tracks
- *Chapters
- *Attachments
- *Void
- *Clusters
- *Cues
- *Tags

24.4.3. Optimum layout with Cues at the front

Cues are usually a big chunk of data referencing a lot of locations in the file. For a player that want to seek in the file they need to seek to the end of the file to have these locations. It is often better if they are placed early in the file. On the other hand that

means players that don't intend to seek will have to read/skip these data no matter what.

Because the Cues reference locations further in the file, it's often complicated to allocate the proper space for that element before all the locations are known. Therefore shis layout is rarely used.

- *SeekHead
- *Info
- *Tracks
- *Chapters
- *Attachments
- *Tags
- *Cues
- *Clusters

24.4.4. Optimum layout for livestreaming

In Livestreaming (Section 23.2) only a few elements make sense. SeekHead and Cues are useless for example. All elements other than the Clusters **MUST** be placed before the Clusters.

- *Info
- *Tracks
- *Attachments (rare)
- *Tags
- *Clusters

25. Security Considerations

Matroska inherits security considerations from EBML.

Attacks on a Matroska Reader could include:

- *Storage of a arbitrary and potentially executable data within an Attachment Element. Matroska Readers that extract or use data from Matroska Attachments **SHOULD** check that the data adheres to expectations.
- *A Matroska Attachment with an inaccurate mime-type.

*Damage to the Encryption and Compression fields (Section 13) that would result in bogus binary data interpreted by the decoder.

26. IANA Considerations

26.1. Matroska Element IDs Registry

This document creates a new IANA registry called the "Matroska Element IDs" registry.

To register a new Element ID in this registry, one needs an Element ID, a Change Controller (IESG or email of registrant) and an optional Reference to a document describing the Element ID.

Element IDs are described in Section 5 of [RFC8794]. Element IDs are encoded using the VINT mechanism described in Section 4 of [RFC8794] and can be between one and five octets long. Five-octet-long Element IDs are possible only if declared in the EBML header.

One-octet Element IDs **MUST** be between 0x80 and 0xFE. These items are valuable because they are short, and they need to be used for commonly repeated elements. Element IDs are to be allocated within this range according to the "RFC Required" policy [RFC8126].

The following one-octet Element ID is RESERVED: 0xFF.

Values in the one-octet range of 0x00 to 0x7F are not valid for use as an Element ID.

Two-octet Element IDs **MUST** be between 0x407F and 0x7FFE. Element IDs are to be allocated within this range according to the "Specification Required" policy [RFC8126].

The following two-octet Element ID is RESERVED: 0x7FFF.

Values in the two-octet ranges of 0x0000 to 0x4000 and 0x8000 to 0xFFFF are not valid for use as an Element ID.

Three-octet Element IDs **MUST** be between 0x203FFF and 0x3FFFFE. Element IDs are to be allocated within this range according to the "First Come First Served" policy [RFC8126].

The following three-octet Element ID is RESERVED: 0x3FFFFF.

Values in the three-octet ranges of 0x000000 to 0x200000 and 0x400000 to 0xFFFFFF are not valid for use as an Element ID.

Four-octet Element IDs **MUST** be between 0x101FFFFF and 0x1FFFFFFE. Four-octet Element IDs are somewhat special in that they are useful for resynchronizing to major structures in the event of data

corruption or loss. As such, four-octet Element IDs are split into two categories. Four-octet Element IDs whose lower three octets (as encoded) would make printable 7-bit ASCII values (0x20 to 0x7E, inclusive) **MUST** be allocated by the "Specification Required" policy. Sequential allocation of values is not required: specifications **SHOULD** include a specific request and are encouraged to do early allocations.

To be clear about the above category: four-octet Element IDs always start with hex 0x10 to 0x1F, and that octet may be chosen so that the entire VINT has some desirable property, such as a specific CRC. The other three octets, when ALL having values between 0x20 (32, ASCII Space) and 0x7E (126, ASCII "~"), fall into this category.

Other four-octet Element IDs may be allocated by the "First Come First Served" policy.

The following four-octet Element ID is RESERVED: 0x1FFFFFFF.

Values in the four-octet ranges of 0x00000000 to 0x10000000 and 0x20000000 to 0xFFFFFFFF are not valid for use as an Element ID.

Five-octet Element IDs (values from 0x080FFFFFFF to 0x0FFFFFFFFE) are RESERVED according to the "Experimental Use" policy [RFC8126]: they may be used by anyone at any time, but there is no coordination.

EBML IDs defined for the EBML Header -- as defined in Section 17.1 of [RFC8794] -- **MUST NOT** be used as Matroska Element IDs.

Matroska Element IDs Values found in this document are assigned as initial values as follows:

Element ID	Element Name	Reference
0xFD	ReferenceVirtual	Described in Section 27.5
0xFB	ReferenceBlock	Described in Section 5.1.3.4.5
0xFA	ReferencePriority	Described in Section 5.1.3.4.4
0xF7	CueTrack	Described in Section 5.1.5.1.2.1
0xF1	CueClusterPosition	Described in Section 5.1.5.1.2.2
0xF0	CueRelativePosition	Described in Section 5.1.5.1.2.3
0xEE	BlockAddID	Described in Section 5.1.3.4.2.3
0xED	TrackJoinUID	

Element ID	Element Name	Reference
		Described in Section 5.1.4.1.33.6
0xEB	CueRefCodecState	Described in Section 27.37
0xEA	CueCodecState	Described in Section 5.1.5.1.2.6
0xE9	TrackJoinBlocks	Described in Section 5.1.4.1.33.5
0xE8	TimeSlice	Described in Section 27.7
0xE7	Timestamp	Described in Section 5.1.3.1
0xE6	TrackPlaneType	Described in Section 5.1.4.1.33.4
0xE5	TrackPlaneUID	Described in Section 5.1.4.1.33.3
0xE4	TrackPlane	Described in Section 5.1.4.1.33.2
0xE3	TrackCombinePlanes	Described in Section 5.1.4.1.33.1
0xE2	TrackOperation	Described in Section 5.1.4.1.33
0xE1	Audio	Described in Section 5.1.4.1.32
0xE0	Video	Described in Section 5.1.4.1.31
0xDB	CueReference	Described in Section 5.1.5.1.2.7
0xD7	TrackNumber	Described in Section 5.1.4.1.1
0xCF	SliceDuration	Described in Section 27.12
0xCE	Delay	Described in Section 27.11
0xCD	FrameNumber	Described in Section 27.9
0xCC	LaceNumber	Described in Section 27.8
0xCB	BlockAdditionID	Described in Section 27.10
0xCA	ReferenceTimestamp	Described in Section 27.15
0xC9	ReferenceOffset	Described in Section 27.14
0xC8	ReferenceFrame	Described in Section 27.13
0xC7	TrickMasterTrackUID	Described in Section 27.29
0xC6	TrickTrackFlag	Described in Section 27.28
0xC4	TrickMasterTrackSegmentUID	Described in Section 27.30
0xC1	TrickTrackSegmentUID	Described in Section 27.27
0xC0	TrickTrackUID	Described in Section 27.26
0xBB	CuePoint	Described in Section 5.1.5.1
0xBA	PixelHeight	Described in Section 5.1.4.1.31.7
0xB9	FlagEnabled	Described in Section 5.1.4.1.4
0xB7	CueTrackPositions	

Element ID	Element Name	Reference
		Described in Section 5.1.5.1.2
0xB6	ChapterAtom	Described in Section 5.1.7.1.4
0xB5	SamplingFrequency	Described in Section 5.1.4.1.32.1
0xB3	CueTime	Described in Section 5.1.5.1.1
0xB2	CueDuration	Described in Section 5.1.5.1.2.4
0xB0	PixelWidth	Described in Section 5.1.4.1.31.6
0xAF	EncryptedBlock	Described in Section 27.16
0xAE	TrackEntry	Described in Section 5.1.4.1
0xAB	PrevSize	Described in Section 5.1.3.2
0xAA	CodecDecodeAll	Described in Section 27.21
0xA7	Position	Described in Section 27.3
0xA6	BlockMore	Described in Section 5.1.3.4.2.1
0xA5	BlockAdditional	Described in Section 5.1.3.4.2.2
0xA4	CodecState	Described in Section 5.1.3.4.6
0xA3	SimpleBlock	Described in Section 5.1.3.3
0xA2	BlockVirtual	Described in Section 27.4
0xA1	Block	Described in Section 5.1.3.4.1
0xA0	BlockGroup	Described in Section 5.1.3.4
0x9F	Channels	Described in Section 5.1.4.1.32.3
0x9D	FieldOrder	Described in Section 5.1.4.1.31.2
0x9C	FlagLacing	Described in Section 5.1.4.1.12
0x9B	BlockDuration	Described in Section 5.1.3.4.3
0x9A	FlagInterlaced	Described in Section 5.1.4.1.31.1
0x98	ChapterFlagHidden	Described in Section 5.1.7.1.4.5
0x97	CueRefCluster	Described in Section 27.35
0x96	CueRefTime	Described in Section 5.1.5.1.2.8
0x92	ChapterTimeEnd	Described in Section 5.1.7.1.4.4
0x91	ChapterTimeStart	

Element ID	Element Name	Reference
		Described in Section 5.1.7.1.4.3
0x8E	Slices	Described in Section 27.6
0x88	FlagDefault	Described in Section 5.1.4.1.5
0x86	CodecID	Described in Section 5.1.4.1.23
0x85	ChapString	Described in Section 5.1.7.1.4.10
0x83	TrackType	Described in Section 5.1.4.1.3
0x80	ChapterDisplay	Described in Section 5.1.7.1.4.9
0x7D7B	ChannelPositions	Described in Section 27.25
0x7BA9	Title	Described in Section 5.1.2.12
0x78B5	OutputSamplingFrequency	Described in Section 5.1.4.1.32.2
0x7675	ProjectionPoseRoll	Described in Section 5.1.4.1.31.46
0x7674	ProjectionPosePitch	Described in Section 5.1.4.1.31.45
0x7673	ProjectionPoseYaw	Described in Section 5.1.4.1.31.44
0x7672	ProjectionPrivate	Described in Section 5.1.4.1.31.43
0x7671	ProjectionType	Described in Section 5.1.4.1.31.42
0x7670	Projection	Described in Section 5.1.4.1.31.41
0x75A2	DiscardPadding	Described in Section 5.1.3.4.7
0x75A1	BlockAdditions	Described in Section 5.1.3.4.2
0x7446	AttachmentLink	Described in Section 5.1.4.1.26
0x73C5	TrackUID	Described in Section 5.1.4.1.2
0x73C4	ChapterUID	Described in Section 5.1.7.1.4.1
0x73A4	SegmentUUID	Described in Section 5.1.2.1
0x7384	SegmentFilename	Described in Section 5.1.2.2
0x7373	Tag	Described in Section 5.1.8.1
0x6FAB	TrackOverlay	Described in Section 5.1.4.1.27
0x6EBC	ChapterSegmentEditionUID	

Element ID	Element Name	Reference
		Described in Section 5.1.7.1.4.7
0x6E67	ChapterSegmentUUID	Described in Section 5.1.7.1.4.6
0x6DF8	MaxCache	Described in Section 5.1.4.1.14
0x6DE7	MinCache	Described in Section 5.1.4.1.13
0x6D80	ContentEncodings	Described in Section 5.1.4.1.34
0x69FC	ChapterTranslateEditionUID	Described in Section 5.1.2.8.3
0x69BF	ChapterTranslateCodec	Described in Section 5.1.2.8.2
0x69A5	ChapterTranslateID	Described in Section 5.1.2.8.1
0x6955	ChapProcessCodecID	Described in Section 5.1.7.1.4.15
0x6944	ChapProcess	Described in Section 5.1.7.1.4.14
0x6933	ChapProcessData	Described in Section 5.1.7.1.4.19
0x6924	ChapterTranslate	Described in Section 5.1.2.8
0x6922	ChapProcessTime	Described in Section 5.1.7.1.4.18
0x6911	ChapProcessCommand	Described in Section 5.1.7.1.4.17
0x68CA	TargetTypeValue	Described in Section 5.1.8.1.1.1
0x67C8	SimpleTag	Described in Section 5.1.8.1.2
0x66FC	TrackTranslateEditionUID	Described in Section 5.1.4.1.30.3
0x66BF	TrackTranslateCodec	Described in Section 5.1.4.1.30.2
0x66A5	TrackTranslateTrackID	Described in Section 5.1.4.1.30.1
0x6624	TrackTranslate	Described in Section 5.1.4.1.30
0x63CA	TargetType	Described in Section 5.1.8.1.1.2
0x63C9	TagEditionUID	Described in Section 5.1.8.1.1.4
0x63C6	TagAttachmentUID	Described in Section 5.1.8.1.1.6
0x63C5	TagTrackUID	

Element ID	Element Name	Reference
		Described in Section 5.1.8.1.1.3
0x63C4	TagChapterUID	Described in Section 5.1.8.1.1.5
0x63C3	ChapterPhysicalEquiv	Described in Section 5.1.7.1.4.8
0x63C0	Targets	Described in Section 5.1.8.1.1
0x63A2	CodecPrivate	Described in Section 5.1.4.1.24
0x6264	BitDepth	Described in Section 5.1.4.1.32.4
0x6240	ContentEncoding	Described in Section 5.1.4.1.34.1
0x61A7	AttachedFile	Described in Section 5.1.6.1
0x58D7	SilentTrackNumber	Described in Section 27.2
0x5854	SilentTracks	Described in Section 27.1
0x5741	WritingApp	Described in Section 5.1.2.14
0x56BB	SeekPreRoll	Described in Section 5.1.4.1.29
0x56AA	CodecDelay	Described in Section 5.1.4.1.28
0x5654	ChapterStringUID	Described in Section 5.1.7.1.4.2
0x55EE	MaxBlockAdditionID	Described in Section 5.1.4.1.18
0x55DA	LuminanceMin	Described in Section 5.1.4.1.31.40
0x55D9	LuminanceMax	Described in Section 5.1.4.1.31.39
0x55D8	WhitePointChromaticityY	Described in Section 5.1.4.1.31.38
0x55D7	WhitePointChromaticityX	Described in Section 5.1.4.1.31.37
0x55D6	PrimaryBChromaticityY	Described in Section 5.1.4.1.31.36
0x55D5	PrimaryBChromaticityX	Described in Section 5.1.4.1.31.35
0x55D4	PrimaryGChromaticityY	Described in Section 5.1.4.1.31.34
0x55D3	PrimaryGChromaticityX	Described in Section 5.1.4.1.31.33
0x55D2	PrimaryRChromaticityY	Described in Section 5.1.4.1.31.32
0x55D1	PrimaryRChromaticityX	

Element ID	Element Name	Reference
		Described in Section 5.1.4.1.31.31
0x55D0	MasteringMetadata	Described in Section 5.1.4.1.31.30
0x55BD	MaxFALL	Described in Section 5.1.4.1.31.29
0x55BC	MaxCLL	Described in Section 5.1.4.1.31.28
0x55BB	Primaries	Described in Section 5.1.4.1.31.27
0x55BA	TransferCharacteristics	Described in Section 5.1.4.1.31.26
0x55B9	Range	Described in Section 5.1.4.1.31.25
0x55B8	ChromaSitingVert	Described in Section 5.1.4.1.31.24
0x55B7	ChromaSitingHorz	Described in Section 5.1.4.1.31.23
0x55B6	CbSubsamplingVert	Described in Section 5.1.4.1.31.22
0x55B5	CbSubsamplingHorz	Described in Section 5.1.4.1.31.21
0x55B4	ChromaSubsamplingVert	Described in Section 5.1.4.1.31.20
0x55B3	ChromaSubsamplingHorz	Described in Section 5.1.4.1.31.19
0x55B2	BitsPerChannel	Described in Section 5.1.4.1.31.18
0x55B1	MatrixCoefficients	Described in Section 5.1.4.1.31.17
0x55B0	Colour	Described in Section 5.1.4.1.31.16
0x55AF	FlagCommentary	Described in Section 5.1.4.1.11
0x55AE	FlagOriginal	Described in Section 5.1.4.1.10
0x55AD	FlagTextDescriptions	Described in Section 5.1.4.1.9
0x55AC	FlagVisualImpaired	Described in Section 5.1.4.1.8
0x55AB	FlagHearingImpaired	Described in Section 5.1.4.1.7
0x55AA	FlagForced	Described in Section 5.1.4.1.6
0x54DD	PixelCropRight	Described in Section 5.1.4.1.31.11

Element ID	Element Name	Reference
0x54CC	PixelCropLeft	Described in Section 5.1.4.1.31.10
0x54BB	PixelCropTop	Described in Section 5.1.4.1.31.9
0x54BA	DisplayHeight	Described in Section 5.1.4.1.31.13
0x54B3	AspectRatioType	Described in Section 27.22
0x54B2	DisplayUnit	Described in Section 5.1.4.1.31.14
0x54B0	DisplayWidth	Described in Section 5.1.4.1.31.12
0x54AA	PixelCropBottom	Described in Section 5.1.4.1.31.8
0x53C0	AlphaMode	Described in Section 5.1.4.1.31.4
0x53B9	OldStereoMode	Described in Section 5.1.4.1.31.5
0x53B8	StereoMode	Described in Section 5.1.4.1.31.3
0x53AC	SeekPosition	Described in Section 5.1.1.1.2
0x53AB	SeekID	Described in Section 5.1.1.1.1
0x537F	TrackOffset	Described in Section 27.17
0x5378	CueBlockNumber	Described in Section 5.1.5.1.2.5
0x536E	Name	Described in Section 5.1.4.1.20
0x535F	CueRefNumber	Described in Section 27.36
0x5035	ContentEncryption	Described in Section 5.1.4.1.34.8
0x5034	ContentCompression	Described in Section 5.1.4.1.34.5
0x5033	ContentEncodingType	Described in Section 5.1.4.1.34.4
0x5032	ContentEncodingScope	Described in Section 5.1.4.1.34.3
0x5031	ContentEncodingOrder	Described in Section 5.1.4.1.34.2
0x4DBB	Seek	Described in Section 5.1.1.1
0x4D80	MuxingApp	Described in Section 5.1.2.13
0x47E8	AESSettingsCipherMode	Described in Section 5.1.4.1.34.12
0x47E7	ContentEncAESSettings	Described in Section 5.1.4.1.34.11

Element ID	Element Name	Reference
0x47E6	ContentSigHashAlgo	Described in Section 27.34
0x47E5	ContentSigAlgo	Described in Section 27.33
0x47E4	ContentSigKeyID	Described in Section 27.32
0x47E3	ContentSignature	Described in Section 27.31
0x47E2	ContentEncKeyID	Described in Section 5.1.4.1.34.10
0x47E1	ContentEncAlgo	Described in Section 5.1.4.1.34.9
0x46AE	FileUID	Described in Section 5.1.6.1.5
0x467E	FileDescription	Described in Section 5.1.6.1.1
0x4675	FileReferral	Described in Section 27.38
0x466E	FileName	Described in Section 5.1.6.1.2
0x4662	FileUsedEndTime	Described in Section 27.40
0x4661	FileUsedStartTime	Described in Section 27.39
0x4660	FileMimeType	Described in Section 5.1.6.1.3
0x465C	FileData	Described in Section 5.1.6.1.4
0x45DD	EditionFlagOrdered	Described in Section 5.1.7.1.3
0x45DB	EditionFlagDefault	Described in Section 5.1.7.1.2
0x45BC	EditionUID	Described in Section 5.1.7.1.1
0x45B9	EditionEntry	Described in Section 5.1.7.1
0x45A3	TagName	Described in Section 5.1.8.1.2.1
0x450D	ChapProcessPrivate	Described in Section 5.1.7.1.4.16
0x44B4	TagDefaultBogus	Described in Section 27.41
0x4489	Duration	Described in Section 5.1.2.10
0x4487	TagString	Described in Section 5.1.8.1.2.5
0x4485	TagBinary	Described in Section 5.1.8.1.2.6
0x4484	TagDefault	Described in Section 5.1.8.1.2.4
0x447B	TagLanguageBCP47	Described in Section 5.1.8.1.2.3
0x447A	TagLanguage	Described in Section 5.1.8.1.2.2
0x4461	DateUTC	

Element ID	Element Name	Reference
		Described in Section 5.1.2.11
0x4444	SegmentFamily	Described in Section 5.1.2.7
0x437E	ChapCountry	Described in Section 5.1.7.1.4.13
0x437D	ChapLanguageBCP47	Described in Section 5.1.7.1.4.12
0x437C	ChapLanguage	Described in Section 5.1.7.1.4.11
0x4255	ContentCompSettings	Described in Section 5.1.4.1.34.7
0x4254	ContentCompAlgo	Described in Section 5.1.4.1.34.6
0x41F0	BlockAddIDValue	Described in Section 5.1.4.1.19.1
0x41ED	BlockAddIDExtraData	Described in Section 5.1.4.1.19.4
0x41E7	BlockAddIDType	Described in Section 5.1.4.1.19.3
0x41E4	BlockAdditionMapping	Described in Section 5.1.4.1.19
0x41A4	BlockAddIDName	Described in Section 5.1.4.1.19.2
0x3EB923	NextUUID	Described in Section 5.1.2.5
0x3E83BB	NextFilename	Described in Section 5.1.2.6
0x3CB923	PrevUUID	Described in Section 5.1.2.3
0x3C83AB	PrevFilename	Described in Section 5.1.2.4
0x3B4040	CodecInfoURL	Described in Section 27.19
0x3A9697	CodecSettings	Described in Section 27.18
0x2FB523	GammaValue	Described in Section 27.23
0x2EB524	UncompressedFourCC	Described in Section 5.1.4.1.31.15
0x2AD7B1	TimestampScale	Described in Section 5.1.2.9
0x26B240	CodecDownloadURL	Described in Section 27.20
0x258688	CodecName	Described in Section 5.1.4.1.25
0x23E383	DefaultDuration	Described in Section 5.1.4.1.15
0x2383E3	FrameRate	Described in Section 27.24
0x234E7A	DefaultDecodedFieldDuration	Described in Section 5.1.4.1.16
0x23314F	TrackTimestampScale	Described in Section 5.1.4.1.17
0x22B59D	LanguageBCP47	Described in Section 5.1.4.1.22
0x22B59C	Language	

Element ID	Element Name	Reference
		Described in Section 5.1.4.1.21
0x1F43B675	Cluster	Described in Section 5.1.3
0x1C53BB6B	Cues	Described in Section 5.1.5
0x1941A469	Attachments	Described in Section 5.1.6
0x18538067	Segment	Described in Section 5.1
0x1654AE6B	Tracks	Described in Section 5.1.4
0x1549A966	Info	Described in Section 5.1.2
0x1254C367	Tags	Described in Section 5.1.8
0x114D9B74	SeekHead	Described in Section 5.1.1
0x1043A770	Chapters	Described in Section 5.1.7

Table 59: IDs and Names for Matroska Element IDs assigned by this document

26.2. Chapter Codec IDs Registry

This document creates a new IANA registry called the "Matroska Chapter Codec IDs" registry. The values correspond to the ChapProcessCodecID value described in Section 5.1.7.1.4.15.

ChapProcessCodecID values of "0" and "1" are RESERVED to the IETF for future use.

26.3. MIME Types

Matroska files and streams are found in three main forms: audio-video files, audio-only and occasionally with stereoscopic video tracks.

The MIME types to use for each type is:

* "video/matroska" for streams containing video tracks

* "audio/matroska" for streams containing audio tracks with no video tracks

* "video/matroska-3d" for streams containing at least a stereoscopic video track

Historically Matroska files and streams have used the following MIME types with a "x-" prefix:

* "video/x-matroska" for streams containing video tracks

* "audio/x-matroska" for streams containing audio tracks with no video tracks

*"video/x-matroska-3d" for streams containing at least a stereoscopic video track

For better compatibility a system **SHOULD** be able to handle both formats. Newer systems **SHOULD NOT** use the historic format and use the format that follows the [RFC6838] format instead.

27. Annex A: Historic Deprecated Elements

As Matroska evolved since 2002 many parts that were considered for use in the format were never used and often incorrectly designed. Many of the elements that were then defined are not found in any known files but were part of public specs. DivX also had a few custom elements that were designed for custom features.

We list these elements that have a known ID that **SHOULD NOT** be reused to avoid colliding with existing files.

27.1. SilentTracks Element

type / id: master / 0x5854

path: \Segment\Cluster\SilentTracks

documentation: The list of tracks that are not used in that part of the stream. It is useful when using overlay tracks on seeking or to decide what track to use.

27.2. SilentTrackNumber Element

type / id: uinteger / 0x58D7

path: \Segment\Cluster\SilentTracks\SilentTrackNumber

documentation: One of the track number that are not used from now on in the stream. It could change later if not specified as silent in a further Cluster.

27.3. Position Element

type / id: uinteger / 0xA7

path: \Segment\Cluster\Position

documentation: The Segment Position of the Cluster in the Segment (0 in live streams). It might help to resynchronise offset on damaged streams.

27.4. BlockVirtual Element

type / id:
binary / 0xA2

path: \Segment\Cluster\BlockGroup\BlockVirtual

documentation: A Block with no data. It **MUST** be stored in the stream at the place the real Block would be in display order.

27.5. ReferenceVirtual Element

type / id: integer / 0xFD

path: \Segment\Cluster\BlockGroup\ReferenceVirtual

documentation: The Segment Position of the data that would otherwise be in position of the virtual block.

27.6. Slices Element

type / id: master / 0x8E

path: \Segment\Cluster\BlockGroup\Slices

documentation: Contains slices description.

27.7. TimeSlice Element

type / id: master / 0xE8

path: \Segment\Cluster\BlockGroup\Slices\TimeSlice

documentation: Contains extra time information about the data contained in the Block. Being able to interpret this Element is not **REQUIRED** for playback.

27.8. LaceNumber Element

type / id: uinteger / 0xCC

path: \Segment\Cluster\BlockGroup\Slices\TimeSlice\LaceNumber

documentation: The reverse number of the frame in the lace (0 is the last frame, 1 is the next to last, etc). Being able to interpret this Element is not **REQUIRED** for playback.

27.9. FrameNumber Element

type / id: uinteger / 0xCD

path: \Segment\Cluster\BlockGroup\Slices\TimeSlice\FrameNumber

documentation:

The number of the frame to generate from this lace with this delay (allow you to generate many frames from the same Block/Frame).

27.10. BlockAdditionID Element

type / id: uinteger / 0xCB

path: \Segment\Cluster\BlockGroup\Slices\TimeSlice\BlockAdditionID

documentation: The ID of the BlockAdditional Element (0 is the main Block).

27.11. Delay Element

type / id: uinteger / 0xCE

path: \Segment\Cluster\BlockGroup\Slices\TimeSlice\Delay

documentation: The delay to apply to the Element, expressed in Track Ticks; see Section 11.1.

27.12. SliceDuration Element

type / id: uinteger / 0xCF

path: \Segment\Cluster\BlockGroup\Slices\TimeSlice\SliceDuration

documentation: The duration to apply to the Element, expressed in Track Ticks; see Section 11.1.

27.13. ReferenceFrame Element

type / id: master / 0xC8

path: \Segment\Cluster\BlockGroup\ReferenceFrame

documentation: Contains information about the last reference frame. See [DivXTrickTrack].

27.14. ReferenceOffset Element

type / id: uinteger / 0xC9

path: \Segment\Cluster\BlockGroup\ReferenceFrame\ReferenceOffset

documentation: The relative offset, in bytes, from the previous BlockGroup element for this Smooth FF/RW video track to the containing BlockGroup element. See [DivXTrickTrack].

27.15. ReferenceTimestamp Element

type / id: uinteger / 0xCA

path: \Segment\Cluster\BlockGroup\ReferenceFrame\ReferenceTimestamp

documentation: The timestamp of the BlockGroup pointed to by ReferenceOffset, expressed in Track Ticks; see Section 11.1. See [DivXTrickTrack].

27.16. EncryptedBlock Element

type / id: binary / 0xAF

path: \Segment\Cluster\EncryptedBlock

documentation: Similar to SimpleBlock, see Section 10.3, but the data inside the Block are Transformed (encrypt and/or signed).

27.17. TrackOffset Element

type / id: integer / 0x537F

path: \Segment\Tracks\TrackEntry\TrackOffset

documentation: A value to add to the Block's Timestamp, expressed in Matroska Ticks -- ie in nanoseconds; see Section 11.1. This can be used to adjust the playback offset of a track.

27.18. CodecSettings Element

type / id: utf-8 / 0x3A9697

path: \Segment\Tracks\TrackEntry\CodecSettings

documentation: A string describing the encoding setting used.

27.19. CodecInfoURL Element

type / id: string / 0x3B4040

path: \Segment\Tracks\TrackEntry\CodecInfoURL

documentation: A URL to find information about the codec used.

27.20. CodecDownloadURL Element

type / id: string / 0x26B240

path: \Segment\Tracks\TrackEntry\CodecDownloadURL

documentation:
A URL to download about the codec used.

27.21. CodecDecodeAll Element

type / id: uinteger / 0xAA

path: \Segment\Tracks\TrackEntry\CodecDecodeAll

documentation: Set to 1 if the codec can decode potentially damaged data.

27.22. AspectRatioType Element

type / id: uinteger / 0x54B3

path: \Segment\Tracks\TrackEntry\Video\AspectRatioType

documentation: Specify the possible modifications to the aspect ratio.

27.23. GammaValue Element

type / id: float / 0x2FB523

path: \Segment\Tracks\TrackEntry\Video\GammaValue

documentation: Gamma Value.

27.24. FrameRate Element

type / id: float / 0x2383E3

path: \Segment\Tracks\TrackEntry\Video\FrameRate

documentation: Number of frames per second. This value is Informational only. It is intended for constant frame rate streams, and **SHOULD NOT** be used for a variable frame rate TrackEntry.

27.25. ChannelPositions Element

type / id: binary / 0x7D7B

path: \Segment\Tracks\TrackEntry\Audio\ChannelPositions

documentation: Table of horizontal angles for each successive channel.

27.26. TrickTrackUID Element

type / id:
uinteger / 0xC0

path: \Segment\Tracks\TrackEntry\TrickTrackUID

documentation: The TrackUID of the Smooth FF/RW video in the paired EBML structure corresponding to this video track. See [DivXTrickTrack].

27.27. TrickTrackSegmentUID Element

type / id: binary / 0xC1

path: \Segment\Tracks\TrackEntry\TrickTrackSegmentUID

documentation: The SegmentUID of the Segment containing the track identified by TrickTrackUID. See [DivXTrickTrack].

27.28. TrickTrackFlag Element

type / id: uinteger / 0xC6

path: \Segment\Tracks\TrackEntry\TrickTrackFlag

documentation: Set to 1 if this video track is a Smooth FF/RW track. If set to 1, MasterTrackUID and MasterTrackSegUID should must be present and BlockGroups for this track must contain ReferenceFrame structures. Otherwise, TrickTrackUID and TrickTrackSegUID must be present if this track has a corresponding Smooth FF/RW track. See [DivXTrickTrack].

27.29. TrickMasterTrackUID Element

type / id: uinteger / 0xC7

path: \Segment\Tracks\TrackEntry\TrickMasterTrackUID

documentation: The TrackUID of the video track in the paired EBML structure that corresponds to this Smooth FF/RW track. See [DivXTrickTrack].

27.30. TrickMasterTrackSegmentUID Element

type / id: binary / 0xC4

path: \Segment\Tracks\TrackEntry\TrickMasterTrackSegmentUID

documentation: The SegmentUID of the Segment containing the track identified by MasterTrackUID. See [DivXTrickTrack].

27.31. ContentSignature Element

type / id:
binary / 0x47E3

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\ContentSignature

documentation: A cryptographic signature of the contents.

27.32. ContentSigKeyID Element

type / id: binary / 0x47E4

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\ContentSigKeyID

documentation: This is the ID of the private key the data was signed with.

27.33. ContentSigAlgo Element

type / id: uinteger / 0x47E5

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\ContentSigAlgo

documentation: The algorithm used for the signature.

27.34. ContentSigHashAlgo Element

type / id: uinteger / 0x47E6

path:
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\ContentSigHashAlgo

documentation: The hash algorithm used for the signature.

27.35. CueRefCluster Element

type / id: uinteger / 0x97

path:
\Segment\Cues\CuePoint\CueTrackPositions\CueReference\CueRefCluster

documentation: The Segment Position of the Cluster containing the referenced Block.

27.36. CueRefNumber Element

type / id: uinteger / 0x535F

path:
\Segment\Cues\CuePoint\CueTrackPositions\CueReference\CueRefNumber

documentation: Number of the referenced Block of Track X in the specified Cluster.

27.37. CueRefCodecState Element

type / id: uinteger / 0xEB

path:
\Segment\Cues\CuePoint\CueTrackPositions\CueReference\CueRefCodecState

documentation: The Segment Position of the Codec State corresponding to this referenced Element. 0 means that the data is taken from the initial Track Entry.

27.38. FileReferral Element

type / id: binary / 0x4675

path: \Segment\Attachments\AttachedFile\FileReferral

documentation: A binary value that a track/codec can refer to when the attachment is needed.

27.39. FileUsedStartTime Element

type / id: uinteger / 0x4661

path: \Segment\Attachments\AttachedFile\FileUsedStartTime

documentation: The timestamp at which this optimized font attachment comes into context, expressed in Segment Ticks which is based on TimestampScale. See [DivXWorldFonts].

27.40. FileUsedEndTime Element

type / id: uinteger / 0x4662

path: \Segment\Attachments\AttachedFile\FileUsedEndTime

documentation: The timestamp at which this optimized font attachment goes out of context, expressed in Segment Ticks which is based on TimestampScale. See [DivXWorldFonts].

27.41. TagDefaultBogus Element

type / id: uinteger / 0x44B4

path: \Segment\Tags\Tag\+SimpleTag\TagDefaultBogus

documentation: A variant of the TagDefault element with a bogus Element ID; see Section 5.1.8.1.2.4.

28. Normative References

[BCP47] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", , DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[Blowfish] Schneier, B., "The Blowfish Encryption Algorithm", , 1993, <https://www.schneier.com/academic/blowfish/>.

[BZIP2] Seward, J., "bzip2", , 18 July 1996, <https://sourceware.org/bzip2/>.

[CIE-1931] Commission Internationale de l'Eclairage, "CIE 1931 Standard Colorimetric System", , 1931, <https://cie.co.at/>.

[FIPS.197] US National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", , DOI 10.6028/NIST.FIPS.197, 26 November 2001, <https://csrc.nist.gov/publications/detail/fips/197/final>.

[FIPS.46-3] US National Institute of Standards and Technology, "Data Encryption Standard (DES)", , FIPS PUB 46, 25 October

1999, <https://csrc.nist.gov/publications/detail/fips/46/3/archive/1999-10-25>.

[IANADomains] "IANA Root Zone Database", , <https://www.iana.org/domains/root/db>.

[IANALangRegistry] "IANA Language Subtag Registry", , 28 February 2013, <https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry>.

[ISO3166-1] International Organization for Standardization, "Codes for the representation of names of countries and their subdivisions -- Part 1: Country code", , ISO 3166-1:2020, August 2020, <https://www.iso.org/standard/72482.html>.

[ISO639-2] United States Library Of Congress, "Codes for the Representation of Names of Languages", , ISO 639-2:1998, 21 December 2017, <https://www.loc.gov/standards/iso639-2/php/code_list.php>.

[LZO] Tarreau, W., Rodgman, R., and M. Oberhumer, "Lempel–Ziv–Oberhumer compression", , 30 October 2018, <https://www.kernel.org/doc/Documentation/lzo.txt>.

[MatroskaCodec] Lhomme, S., Bunkus, M., and D. Rice, "Media Container Codec Specifications", , Work in Progress, Internet-Draft, draft-ietf-cellar-codec-09, 12 April 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-cellar-codec-09>.

[MatroskaTags] Lhomme, S., Bunkus, M., and D. Rice, "Matroska Media Container Tag Specifications", , Work in Progress, Internet-Draft, draft-ietf-cellar-tags-09, 12 April 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-cellar-tags-09>.

[RFC1950] Deutsch, P. and J-L. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, DOI 10.17487/RFC1950, May 1996, <https://www.rfc-editor.org/info/rfc1950>.

[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>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC8081] Lilley, C., "The "font" Top-Level Media Type", RFC 8081, DOI 10.17487/RFC8081, February 2017, <https://www.rfc-editor.org/info/rfc8081>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC8794] Lhomme, S., Rice, D., and M. Bunkus, "Extensible Binary Meta Language", RFC 8794, DOI 10.17487/RFC8794, July 2020, <https://www.rfc-editor.org/info/rfc8794>.

[SP.800-38A] US National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", , DOI 10.6028/NIST.SP.800-38A, 1 December 2001, <https://csrc.nist.gov/publications/detail/fips/197/final>.

[SP.800-67] US National Institute of Standards and Technology, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher", , DOI 10.6028/10.6028/NIST.SP.800-67r2, 1 November 2017, <https://csrc.nist.gov/publications/detail/fips/197/final>.

[Twofish] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., and N. Ferguson, "Twofish: A 128-Bit Block Cipher", , 15 June 1998, <https://www.schneier.com/academic/twofish/>.

[WebM-Enc] Galligan, F., "WebM Encryption", , 19 September 2016, <https://www.webmproject.org/docs/webm-encryption/>.

[WebVTT] Pieters, S., Pfeiffer, S., Ed., Jägenstedt, P., and I. Hickson, "WebVTT Cue Identifier", , 4 April 2019, <https://www.w3.org/TR/webvtt1/#webvtt-cue-identifier>.

29. Informative References

[AVIFormat]

Microsoft, "AVI RIFF File Reference", , 31 May 2018, <https://docs.microsoft.com/en-us/windows/win32/directshow/avi-riff-file-reference>.

[DivXTrickTrack] "DivX Trick Track Extensions", , 14 December 2010, <http://web.archive.org/web/20101222001148/http://labs.divx.com/node/16601>.

[DivXWorldFonts] "DivX World Fonts Extensions", , 14 December 2010, <http://web.archive.org/web/20110214132246/http://labs.divx.com/node/16602>.

[DVD-Video] DVD Forum, "DVD-Books: Part 3 DVD-Video Book", , 1 November 1995, <http://www.dvdforum.org/>.

[FourCC-RGB] Silicon.dk ApS, "RGB Pixel Format FourCCs", , <https://www.fourcc.org/rgb.php>.

[FourCC-YUV] Silicon.dk ApS, "YUV Pixel Format FourCCs", , <https://www.fourcc.org/yuv.php>.

[MCF] "Media Container Format", , 17 July 2002, <http://mukoli.free.fr/mcf/mcf.html>.

[RFC3533] Pfeiffer, S., "The Ogg Encapsulation Format Version 0", RFC 3533, DOI 10.17487/RFC3533, May 2003, <https://www.rfc-editor.org/info/rfc3533>.

[RFC5378] Bradner, S., Ed. and J. Contreras, Ed., "Rights Contributors Provide to the IETF Trust", BCP 78, RFC 5378, DOI 10.17487/RFC5378, November 2008, <https://www.rfc-editor.org/info/rfc5378>.

[RFC8179] Bradner, S. and J. Contreras, "Intellectual Property Rights in IETF Technology", BCP 79, RFC 8179, DOI 10.17487/RFC8179, May 2017, <https://www.rfc-editor.org/info/rfc8179>.

Authors' Addresses

Steve Lhomme

Email: slhomme@matroska.org

Moritz Bunkus

Email: moritz@bunkus.org

Dave Rice

Email: dave@dericed.com