
Stream: Internet Engineering Task Force (IETF)
RFC: [9559](#)
Updates: [8794](#)
Category: Standards Track
Published: April 2024
ISSN: 2070-1721
Authors: S. Lhomme M. Bunkus D. Rice

RFC 9559

Matroska Media Container Format Specifications

Abstract

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

This document updates RFC 8794 to permit the use of a previously reserved Extensible Binary Meta Language (EBML) Element ID.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <https://www.rfc-editor.org/info/rfc9559>.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	8
2. Status of This Document	9
3. Notation and Conventions	9
4. Matroska Overview	10
4.1. Principles	10
4.2. Updates to RFC 8794	10
4.3. Added EBML Constraints	11
4.4. Design Rules	11
4.5. Data Layout	12
5. Matroska Schema	20
5.1. Segment Element	20
5.1.1. SeekHead Element	20
5.1.1.1. Seek Element	21
5.1.2. Info Element	21
5.1.2.1. SegmentUUID Element	21
5.1.2.2. SegmentFilename Element	22
5.1.2.3. PrevUUID Element	22
5.1.2.4. PrevFilename Element	22
5.1.2.5. NextUUID Element	22
5.1.2.6. NextFilename Element	22
5.1.2.7. SegmentFamily Element	23
5.1.2.8. ChapterTranslate Element	23
5.1.2.9. TimestampScale Element	24
5.1.2.10. Duration Element	24
5.1.2.11. DateUTC Element	24
5.1.2.12. Title Element	24
5.1.2.13. MuxingApp Element	25
5.1.2.14. WritingApp Element	25

5.1.3. Cluster Element	25
5.1.3.1. Timestamp Element	25
5.1.3.2. Position Element	25
5.1.3.3. PrevSize Element	26
5.1.3.4. SimpleBlock Element	26
5.1.3.5. BlockGroup Element	26
5.1.4. Tracks Element	29
5.1.4.1. TrackEntry Element	29
5.1.5. Cues Element	62
5.1.5.1. CuePoint Element	62
5.1.6. Attachments Element	64
5.1.6.1. AttachedFile Element	64
5.1.7. Chapters Element	65
5.1.7.1. EditionEntry Element	65
5.1.8. Tags Element	70
5.1.8.1. Tag Element	70
6. Matroska Element Ordering	75
6.1. Top-Level Elements	75
6.2. CRC-32	75
6.3. SeekHead	76
6.4. Cues (Index)	76
6.5. Info	76
6.6. Chapters Element	76
6.7. Attachments	76
6.8. Tags	77
7. Matroska Versioning	77
8. Stream Copy	78
9. DefaultDecodedFieldDuration	78
10. Cluster Blocks	78
10.1. Block Structure	79

10.2. SimpleBlock Structure	80
10.3. Block Lacing	82
10.3.1. No Lacing	82
10.3.2. Xiph Lacing	82
10.3.3. EBML Lacing	83
10.3.4. Fixed-size Lacing	84
10.3.5. Laced Frames Timestamp	85
10.4. Random Access Points	85
11. Timestamps	89
11.1. Timestamp Ticks	89
11.1.1. Matroska Ticks	89
11.1.2. Segment Ticks	89
11.1.3. Track Ticks	90
11.2. Block Timestamps	90
11.3. TimestampScale Rounding	91
12. Language Codes	91
13. Country Codes	91
14. Encryption	92
15. Image Presentation	93
15.1. Cropping	93
15.2. Rotation	93
16. Segment Position	93
16.1. Segment Position Exception	94
16.2. Example of Segment Position	94
17. Linked Segments	94
17.1. Hard Linking	95
17.2. Medium Linking	97
17.2.1. Linked-Duration	97
17.2.2. Linked-Edition	98

18. Track Flags	98
18.1. Default Flag	98
18.2. Forced Flag	98
18.3. Hearing-Impaired Flag	98
18.4. Visual-Impaired Flag	98
18.5. Descriptions Flag	99
18.6. Original Flag	99
18.7. Commentary Flag	99
18.8. Track Operation	99
18.9. Overlay Track	99
18.10. Multi-planar and 3D Videos	99
19. Default Track Selection	100
19.1. Audio Selection	100
19.2. Subtitle Selection	102
20. Chapters	103
20.1. EditionEntry	103
20.1.1. EditionFlagDefault	103
20.1.2. Default Edition	103
20.1.3. EditionFlagOrdered	104
20.1.3.1. Ordered-Edition and Matroska Segment Linking	105
20.2. ChapterAtom	105
20.2.1. ChapterTimeStart	105
20.2.2. ChapterTimeEnd	105
20.2.3. Nested Chapters	106
20.2.4. Nested Chapters in Ordered Chapters	106
20.2.5. ChapterFlagHidden	106
20.3. Menu Features	107
20.4. Physical Types	107
20.5. Chapter Examples	108
20.5.1. Example 1: Basic Chaptering	108

20.5.2. Example 2: Nested Chapters	110
20.5.2.1. The Micronauts "Bleep To Bleep"	110
21. Attachments	112
21.1. Cover Art	112
21.2. Font Files	113
22. Cues	114
22.1. Recommendations	115
23. Matroska Streaming	115
23.1. File Access	115
23.2. Livestreaming	116
24. Tags	116
24.1. Tags Precedence	116
24.2. Tag Levels	117
25. Implementation Recommendations	117
25.1. Cluster	117
25.2. SeekHead	117
25.3. Optimum Layouts	117
25.3.1. Optimum Layout for a Muxer	118
25.3.2. Optimum Layout after Editing Tags	118
25.3.3. Optimum Layout with Cues at the Front	118
25.3.4. Optimum Layout for Livestreaming	119
26. Security Considerations	119
27. IANA Considerations	120
27.1. Matroska Element IDs Registry	120
27.2. Chapter Codec IDs Registry	131
27.3. Media Types	131
27.3.1. For Files Containing Video Tracks	132
27.3.2. For Files Containing Audio Tracks with No Video Tracks	132
27.3.3. For Files Containing a Stereoscopic Video Track	133

28. References	134
28.1. Normative References	134
28.2. Informative References	135
Appendix A. Historic Deprecated Elements	137
A.1. SilentTracks Element	137
A.2. SilentTrackNumber Element	138
A.3. BlockVirtual Element	138
A.4. ReferenceVirtual Element	138
A.5. Slices Element	138
A.6. TimeSlice Element	138
A.7. LaceNumber Element	138
A.8. FrameNumber Element	139
A.9. BlockAdditionID Element	139
A.10. Delay Element	139
A.11. SliceDuration Element	139
A.12. ReferenceFrame Element	139
A.13. ReferenceOffset Element	139
A.14. ReferenceTimestamp Element	140
A.15. EncryptedBlock Element	140
A.16. MinCache Element	140
A.17. MaxCache Element	140
A.18. TrackOffset Element	140
A.19. CodecSettings Element	140
A.20. CodecInfoURL Element	141
A.21. CodecDownloadURL Element	141
A.22. CodecDecodeAll Element	141
A.23. TrackOverlay Element	141
A.24. AspectRatioType Element	141
A.25. GammaValue Element	141
A.26. FrameRate Element	141

A.27. ChannelPositions Element	142
A.28. TrickTrackUID Element	142
A.29. TrickTrackSegmentUID Element	142
A.30. TrickTrackFlag Element	142
A.31. TrickMasterTrackUID Element	142
A.32. TrickMasterTrackSegmentUID Element	142
A.33. ContentSignature Element	143
A.34. ContentSigKeyID Element	143
A.35. ContentSigAlgo Element	143
A.36. ContentSigHashAlgo Element	143
A.37. CueRefCluster Element	143
A.38. CueRefNumber Element	144
A.39. CueRefCodecState Element	144
A.40. FileReferral Element	144
A.41. FileUsedStartTime Element	144
A.42. FileUsedEndTime Element	144
A.43. TagDefaultBogus Element	144
Authors' Addresses	145

1. Introduction

Matroska is an audiovisual data container format. It was derived from a project called [MCF] but diverges from it significantly because it is based on EBML (Extensible Binary Meta Language) [RFC8794], a binary derivative of XML. EBML provides significant advantages in terms of future format extensibility, without breaking file support in parsers reading the previous versions.

To avoid any misunderstandings, it is essential to clarify exactly what an audio/video container is:

- It is NOT a video or audio compression format (codec).
- It is an envelope in 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 such as:

- 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 [[RFC9110](#)], FTP [[RFC0959](#)], SMB [[SMB-CIFS](#)], etc.)
- Menus (like menus that DVDs have [[DVD-Video](#)])

2. Status of This Document

This document covers Matroska versions 1, 2, 3, and 4. Matroska version 4 is the current version. Matroska versions 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 the following terms in order to define the format and application of Matroska:

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 the primary purpose of playing audiovisual files, including Matroska documents.

Matroska Writer: A data writer that creates Matroska documents.

4. Matroska Overview

4.1. Principles

Matroska is a Document Type of EBML. 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 that cover EBML Element Types (Section 7), EBML Schema (Section 11.1), and EBML Structure (Section 3).

4.2. Updates to RFC 8794

Because of an oversight, [RFC8794] reserved EBML ID 0x80, which is used by deployed Matroska implementations. For this reason, this specification updates [RFC8794] to make 0x80 a legal EBML ID. Additionally, this specification makes the following updates:

- Section 17.1 of [RFC8794] (per Erratum ID #7189 [Err7189])

OLD:

One-octet Element IDs **MUST** be between 0x81 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 IDs are RESERVED: 0xFF and 0x80.

NEW:

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.

- Section 5 of [RFC8794] (per Erratum ID #7191 [Err7191])

OLD:

Element ID	Octet Length	Range of Valid Element IDs	Number of Valid Element IDs
	1	0x81 - 0xFE	126

NEW:

Element ID	Octet Length	Range of Valid Element IDs	Number of Valid Element IDs
	1	0x80 - 0xFE	127

4.3. Added EBML Constraints

As an EBML Document Type, Matroska adds the following constraints to the EBML specification [RFC8794]:

- 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.

4.4. Design Rules

The Root Element and all Top-Level Elements **MUST** 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 were present in the file but had a length of 0. They always assumed the value was 0 for integers/dates or 0x0p+0, the textual expression of floats using the format in [ISO9899], no matter the default value of the element that 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 apply:

- 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 strings cannot be defined in the XML Schema.

4.5. 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 that 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 diagrams in 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)
- 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 Element.

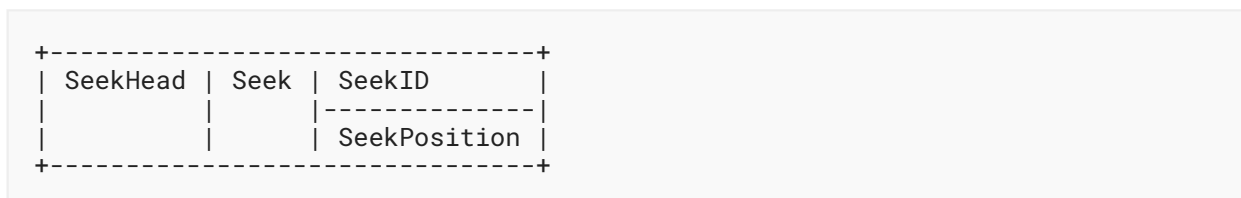


Figure 2: Representation of a SeekHead Element

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

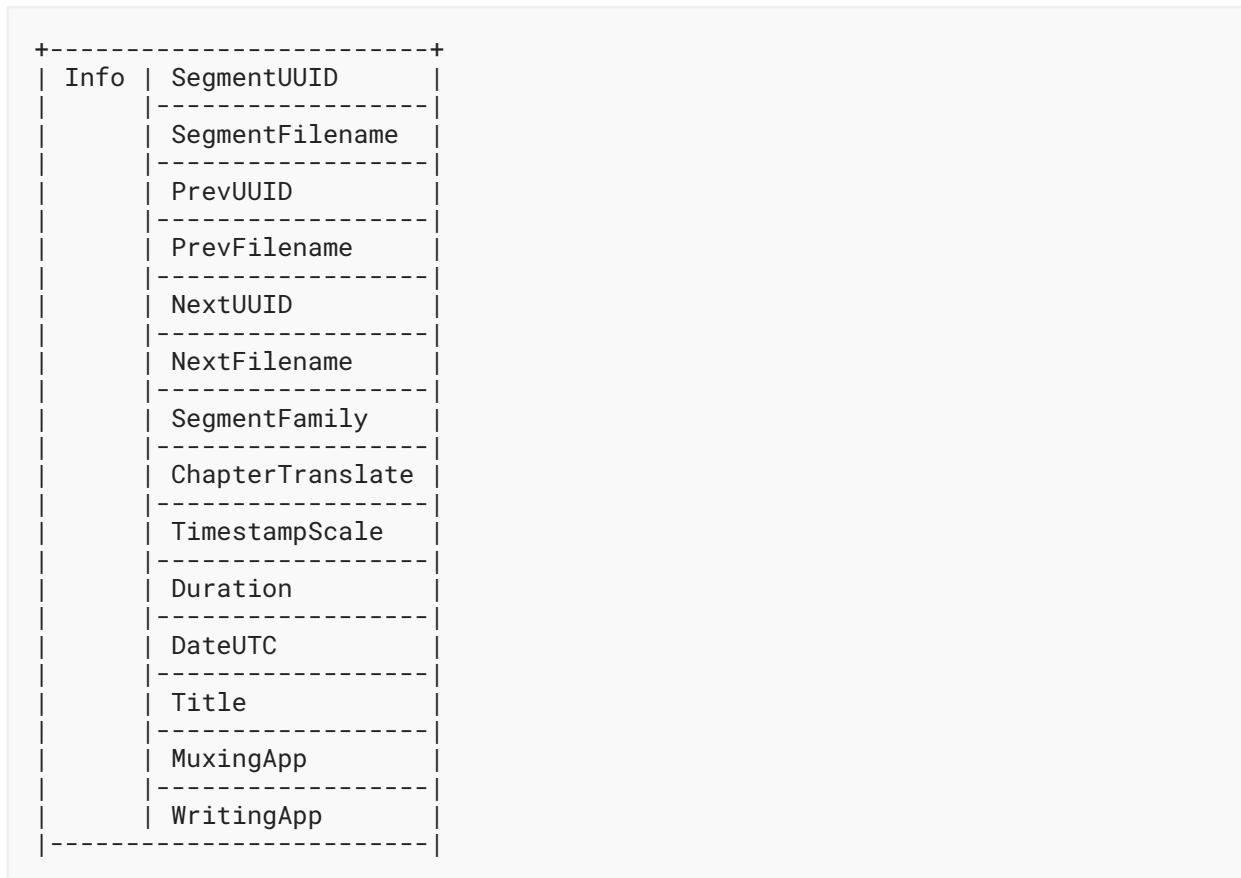


Figure 3: Representation of an `Info` Element and Its Child Elements

The `Tracks` Element defines the technical details for each track and can store the name, number, UID, 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.

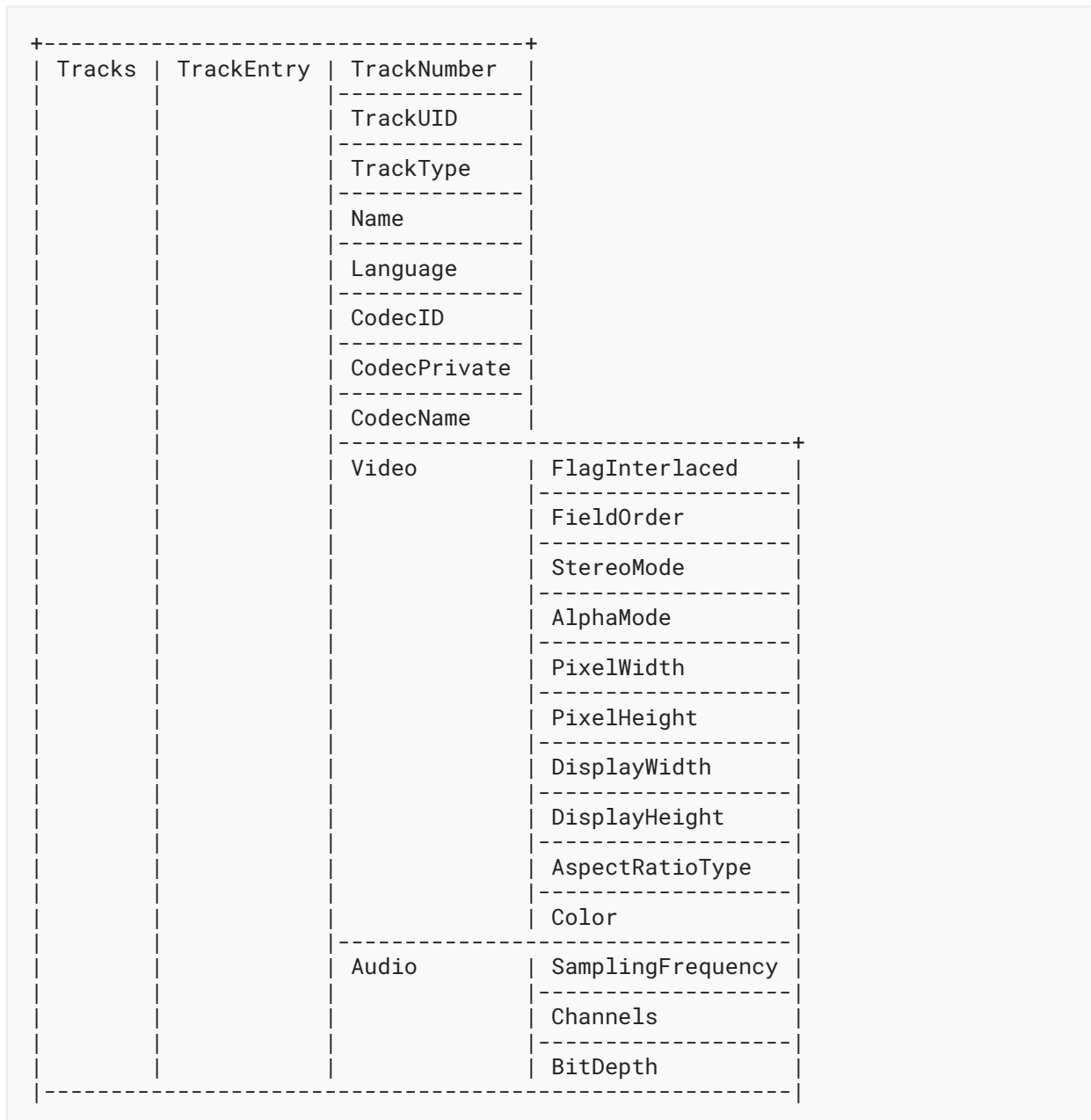


Figure 4: Representation of the Tracks Element and a Selection of Its Descendant Elements

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

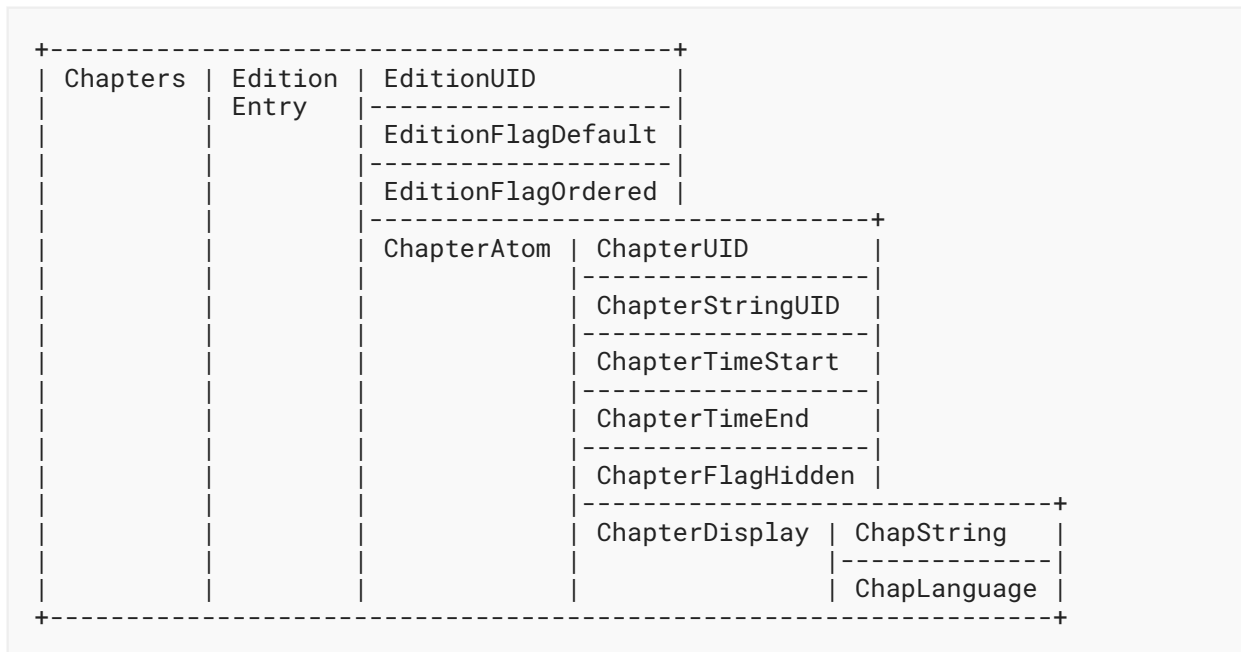


Figure 5: Representation of the Chapters Element and a Selection of Its Descendant Elements

Cluster Elements contain the content for each track, e.g., video frames. A Matroska file **SHOULD** contain at least one Cluster Element. In the rare case it doesn't, there should be a form of Segment linking with other Segments, possibly using Chapters; see [Section 17](#).

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, unless a different value is needed to accommodate for more Blocks; see [Section 11.2](#).

Cluster Elements contain one or more block element, such as BlockGroup or SimpleBlock elements. In some situations, a Cluster Element **MAY** contain no 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.

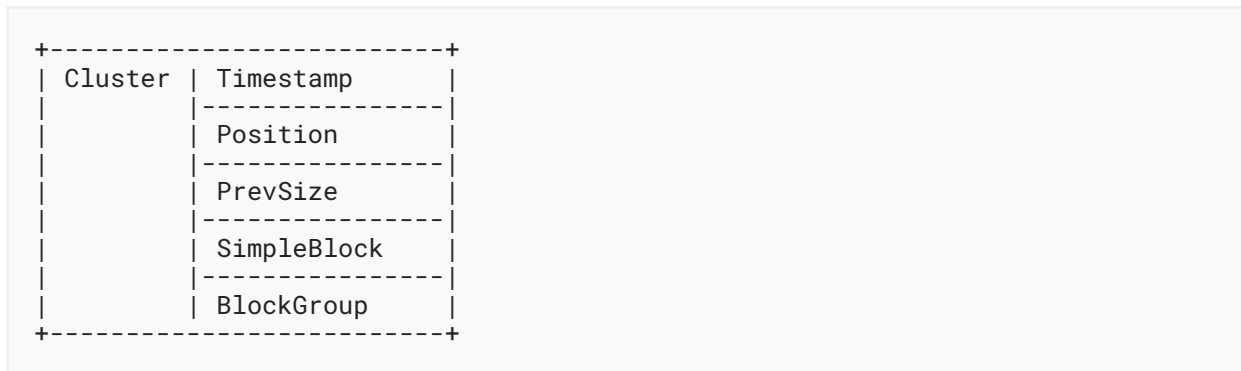


Figure 6: Representation of a Cluster Element and Its Immediate Child Elements

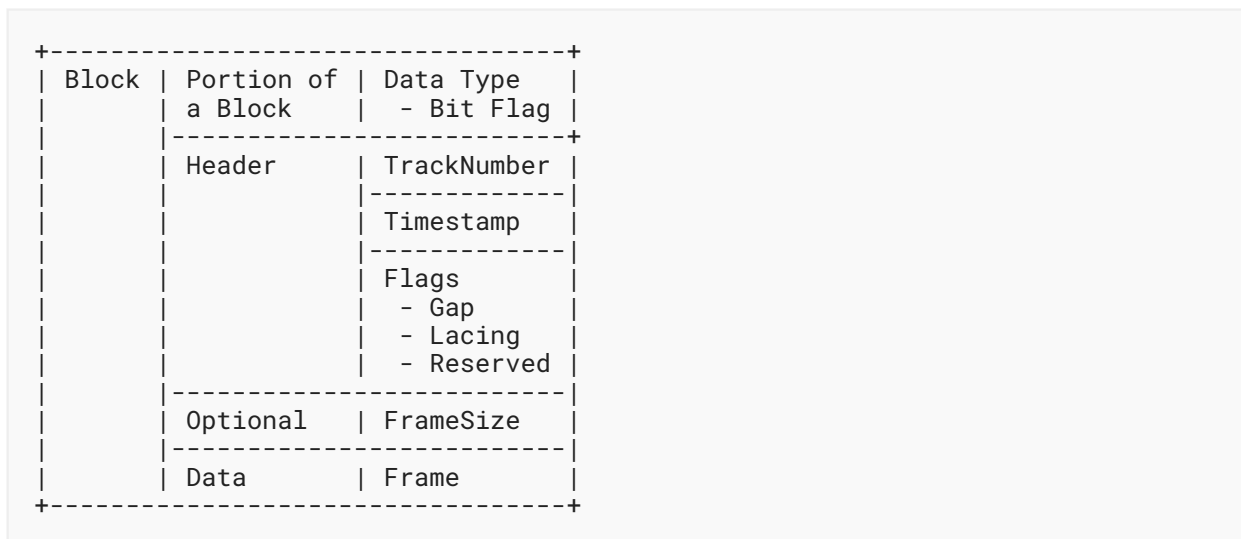


Figure 7: Representation of the Block Element Structure

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 Element 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, the 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 after, 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 is 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 to look 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 the location is stored in the CueTrackPositions Element.

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

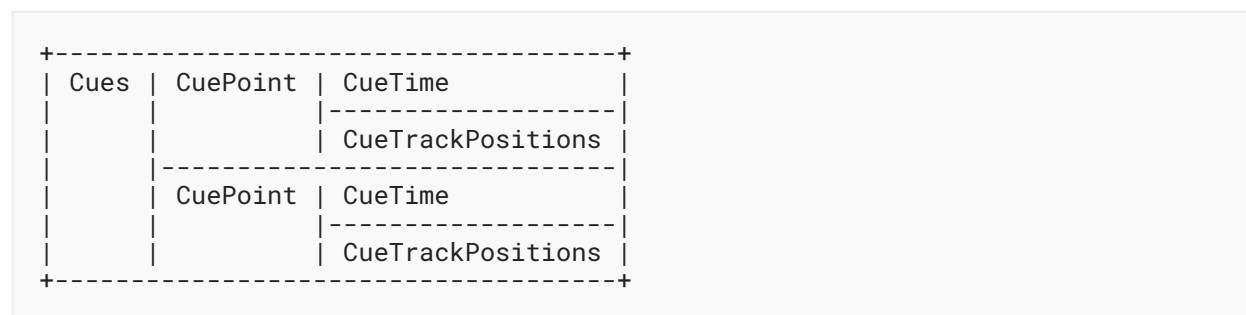


Figure 8: Representation of a Cues Element and Two Levels of Its Descendant Elements

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

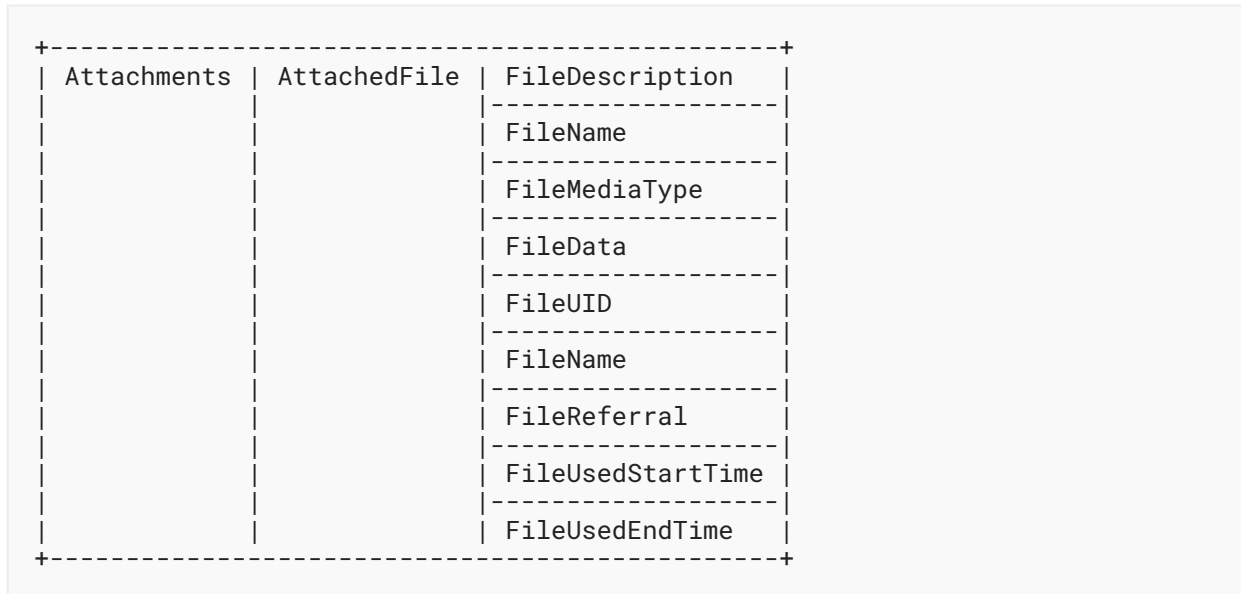


Figure 9: Representation of an Attachments Element

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: scriptwriters, singers, 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 that defines the Elements and structure of Matroska using the EBML Schema elements and attributes defined in [Section 11.1](#) of [\[RFC8794\]](#). The EBML Schema defines every valid Matroska element in a manner defined by the EBML specification [\[RFC8794\]](#).

Attributes using their default value (like `minOccurs`, `minver`, etc.) or attributes with undefined values (like `length`, `maxver`, etc.) are omitted.

The definitions for each Matroska Element are provided below.

5.1. Segment Element

id / type: 0x18538067 / master

unknownsizeallowed: True

path: \Segment

minOccurs / maxOccurs: 1 / 1

definition: The Root Element that contains all other Top-Level Elements; see [Section 4.5](#).

5.1.1. SeekHead Element

id / type: 0x114D9B74 / master
path: \Segment\SeekHead
maxOccurs: 2
definition: Contains seeking information of Top-Level Elements; see [Section 4.5](#).

5.1.1.1. Seek Element

id / type: 0x4DBB / master
path: \Segment\SeekHead\Seek
minOccurs: 1
definition: Contains a single seek entry to an EBML Element.

5.1.1.1.1. SeekID Element

id / type: 0x53AB / binary
length: 4
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

id / type: 0x53AC / uinteger
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

id / type: 0x1549A966 / master
path: \Segment\Info
minOccurs / maxOccurs: 1 / 1
recurring: True
definition: Contains general information about the Segment.

5.1.2.1. SegmentUUID Element

id / type: 0x73A4 / binary
length: 16
path: \Segment\Info\SegmentUUID
maxOccurs: 1
definition: A randomly generated UID that identifies the Segment amongst many others (128 bits). It is equivalent to a Universally Unique Identifier (UUID) v4 [[RFC4122](#)] with all bits randomly (or pseudorandomly) chosen. An actual UUID v4 value, where some bits are not random, **MAY** also be used.

usage notes: If the Segment is a part of a Linked Segment, then this Element is **REQUIRED**. The value of the UID **MUST** contain at least one bit set to 1.

5.1.2.2. SegmentFilename Element

id / type: 0x7384 / utf-8
path: \Segment\Info\SegmentFilename
maxOccurs: 1
definition: A filename corresponding to this Segment.

5.1.2.3. PrevUUID Element

id / type: 0x3CB923 / binary
length: 16
path: \Segment\Info\PrevUUID
maxOccurs: 1
definition: An ID that identifies the previous Segment of a Linked Segment.
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

id / type: 0x3C83AB / utf-8
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

id / type: 0x3EB923 / binary
length: 16
path: \Segment\Info\NextUUID
maxOccurs: 1
definition: An ID that identifies the next Segment of a Linked Segment.
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

id / type: 0x3E83BB / utf-8

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

id / type: 0x4444 / binary

length: 16

path: \Segment\Info\SegmentFamily

definition: A UID that all Segments of a Linked Segment **MUST** share (128 bits). It is equivalent to a UUID v4 [RFC4122] with all bits randomly (or pseudorandomly) chosen. An actual UUID v4 value, where some bits are not random, **MAY** also be used.

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

5.1.2.8. ChapterTranslate Element

id / type: 0x6924 / master

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 segments 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 the Chapter Codec without changing the content of the codec data, just the Segment mapping.

5.1.2.8.1. ChapterTranslateID Element

id / type: 0x69A5 / binary

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

id / type: 0x69BF / uinteger

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: See [Table 1](#).

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

id / type: 0x69FC / uinteger

path: \Segment\Info\ChapterTranslate\ChapterTranslateEditionUID

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

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

5.1.2.9. TimestampScale Element

id / type / default: 0x2AD7B1 / uinteger / 1000000

range: not 0

path: \Segment\Info\TimestampScale

minOccurs / maxOccurs: 1 / 1

definition: Base unit for Segment Ticks and Track Ticks, in nanoseconds. A TimestampScale value of 1000000 means scaled timestamps in the Segment are expressed in milliseconds; see [Section 11](#) on how to interpret timestamps.

5.1.2.10. Duration Element

id / type: 0x4489 / float

range: > 0x0p+0

path: \Segment\Info\Duration

maxOccurs: 1

definition: Duration of the Segment, expressed in Segment Ticks, which are based on TimestampScale; see [Section 11.1](#).

5.1.2.11. DateUTC Element

id / type: 0x4461 / date

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

id / type: 0x7BA9 / utf-8
path: \Segment\Info\Title
maxOccurs: 1
definition: General name of the Segment.

5.1.2.13. MuxingApp Element

id / type: 0x4D80 / utf-8
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

id / type: 0x5741 / utf-8
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

id / type: 0x1F43B675 / master
unknownsizeallowed: True
path: \Segment\Cluster
definition: The Top-Level Element containing the (monolithic) Block structure.

5.1.3.1. Timestamp Element

id / type: 0xE7 / uinteger
path: \Segment\Cluster\Timestamp
minOccurs / maxOccurs: 1 / 1
definition: Absolute timestamp of the cluster, expressed in Segment Ticks, which are 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. Position Element

id / type: 0xA7 / uinteger
path: \Segment\Cluster\Position
maxOccurs: 1
maxver: 4

definition: The Segment Position of the Cluster in the Segment (0 in live streams). It might help to resynchronize the offset on damaged streams.

5.1.3.3. PrevSize Element

id / type: 0xAB / uinteger

path: \Segment\Cluster\PrevSize

maxOccurs: 1

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

5.1.3.4. SimpleBlock Element

id / type: 0xA3 / binary

path: \Segment\Cluster\SimpleBlock

minver: 2

definition: Similar to Block (see [Section 10.1](#)) but without all the extra information. Mostly used to reduce overhead when no extra feature is needed; see [Section 10.2](#) on SimpleBlock Structure.

5.1.3.5. BlockGroup Element

id / type: 0xA0 / master

path: \Segment\Cluster\BlockGroup

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

5.1.3.5.1. Block Element

id / type: 0xA1 / binary

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 10.1](#) on Block Structure.

5.1.3.5.2. BlockAdditions Element

id / type: 0x75A1 / master

path: \Segment\Cluster\BlockGroup\BlockAdditions

maxOccurs: 1

definition: Contains additional binary data to complete the main one; see [Section 4.1.5](#) of [\[MatroskaCodec\]](#) for more information. An EBML parser that has no knowledge of the Block structure could still see and use/skip these data.

5.1.3.5.2.1. BlockMore Element

id / type: 0xA6 / master
 path: \Segment\Cluster\BlockGroup\BlockAdditions\BlockMore
 minOccurs: 1
 definition: Contains the BlockAdditional and some parameters.

5.1.3.5.2.2. BlockAdditional Element

id / type: 0xA5 / binary
 path: \Segment\Cluster\BlockGroup\BlockAdditions\BlockMore\BlockAdditional
 minOccurs / maxOccurs: 1 / 1
 definition: Interpreted by the codec as it wishes (using the BlockAddID).

5.1.3.5.2.3. BlockAddID Element

id / type / default: 0xEE / uinteger / 1
 range: not 0
 path: \Segment\Cluster\BlockGroup\BlockAdditions\BlockMore\BlockAddID
 minOccurs / maxOccurs: 1 / 1
 definition: An ID that identifies how to interpret the BlockAdditional data; see [Section 4.1.5](#) 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.

5.1.3.5.3. BlockDuration Element

id / type: 0x9B / uinteger
 path: \Segment\Cluster\BlockGroup\BlockDuration
 minOccurs / maxOccurs: see implementation notes / 1
 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: See [Table 2](#).

attribute	note
minOccurs	BlockDuration MUST be set (minOccurs=1) if the associated TrackEntry stores a DefaultDuration value.

attribute	note
default	When not written and with no DefaultDuration, the value is assumed to be the difference between the timestamp of this Block and the timestamp of the next Block in "display" order (not coding order).

Table 2: BlockDuration Implementation Notes

5.1.3.5.4. ReferencePriority Element

id / type / default: 0xFA / uinteger / 0

path: \Segment\Cluster\BlockGroup\ReferencePriority

minOccurs / maxOccurs: 1 / 1

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

5.1.3.5.5. ReferenceBlock Element

id / type: 0xFB / integer

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 that this Block depends on. Historically, Matroska Writers didn't write the actual Block(s) that this Block depends on, but they did write *some* Block(s) in the past.

The value "0" **MAY** also be used to signify that this Block cannot be decoded on its own, but without knowledge of which Block is necessary. In this case, other ReferenceBlock Elements **MUST NOT** be found in the same BlockGroup.

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

5.1.3.5.6. CodecState Element

id / type: 0xA4 / binary

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.

5.1.3.5.7. DiscardPadding Element

id / type: 0x75A2 / integer

path: \Segment\Cluster\BlockGroup\DiscardPadding

maxOccurs: 1

minver: 4

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

5.1.4. Tracks Element

id / type: 0x1654AE6B / master

path: \Segment\Tracks

maxOccurs: 1

recurring: True

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

5.1.4.1. TrackEntry Element

id / type: 0xAE / master

path: \Segment\Tracks\TrackEntry

minOccurs: 1

definition: Describes a track with all Elements.

5.1.4.1.1. TrackNumber Element

id / type: 0xD7 / uinteger

range: not 0

path: \Segment\Tracks\TrackEntry\TrackNumber

minOccurs / maxOccurs: 1 / 1

definition: The track number as used in the Block Header.

5.1.4.1.2. TrackUID Element

id / type: 0x73C5 / uinteger

range: not 0

path: \Segment\Tracks\TrackEntry\TrackUID

minOccurs / maxOccurs: 1 / 1

definition: A UID that identifies the Track.

stream copy: True ([Section 8](#))

5.1.4.1.3. TrackType Element

id / type: 0x83 / uinteger

path: \Segment\Tracks\TrackEntry\TrackType

minOccurs / maxOccurs: 1 / 1

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

defined values: See [Table 3](#).

stream copy: True ([Section 8](#))

value	label	contents of each frame
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

id / type / default: 0xB9 / uinteger / 1

range: 0-1

path: \Segment\Tracks\TrackEntry\FlagEnabled

minOccurs / maxOccurs: 1 / 1

minver: 2

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

5.1.4.1.5. FlagDefault Element

id / type / default: 0x88 / uinteger / 1

range: 0-1

path: \Segment\Tracks\TrackEntry\FlagDefault

minOccurs / maxOccurs: 1 / 1

definition: Set if the track (audio, video, or subs) is eligible for automatic selection by the player; see [Section 19](#) for more details.

5.1.4.1.6. FlagForced Element

id / type / default: 0x55AA / uinteger / 0

range: 0-1
path: \Segment\Tracks\TrackEntry\FlagForced
minOccurs / maxOccurs: 1 / 1
definition: Applies only to subtitles. Set if the track is eligible for automatic selection by the player if it matches the user's language preference, even if the user's preferences would not normally enable subtitles with the selected audio track; this can be used for tracks containing only translations of audio in foreign languages or on-screen text. See [Section 19](#) for more details.

5.1.4.1.7. FlagHearingImpaired Element

id / type: 0x55AB / uinteger
range: 0-1
path: \Segment\Tracks\TrackEntry\FlagHearingImpaired
maxOccurs: 1
minver: 4
definition: Set to 1 if and only if the track is suitable for users with hearing impairments.

5.1.4.1.8. FlagVisualImpaired Element

id / type: 0x55AC / uinteger
range: 0-1
path: \Segment\Tracks\TrackEntry\FlagVisualImpaired
maxOccurs: 1
minver: 4
definition: Set to 1 if and only if the track is suitable for users with visual impairments.

5.1.4.1.9. FlagTextDescriptions Element

id / type: 0x55AD / uinteger
range: 0-1
path: \Segment\Tracks\TrackEntry\FlagTextDescriptions
maxOccurs: 1
minver: 4
definition: Set to 1 if and only if the track contains textual descriptions of video content.

5.1.4.1.10. FlagOriginal Element

id / type: 0x55AE / uinteger
range: 0-1
path: \Segment\Tracks\TrackEntry\FlagOriginal
maxOccurs: 1
minver: 4
definition: Set to 1 if and only if the track is in the content's original language.

5.1.4.1.11. FlagCommentary Element

id / type: 0x55AF / uinteger
range: 0-1
path: \Segment\Tracks\TrackEntry\FlagCommentary
maxOccurs: 1
minver: 4
definition: Set to 1 if and only if the track contains commentary.

5.1.4.1.12. FlagLacing Element

id / type / default: 0x9C / uinteger / 1
range: 0-1
path: \Segment\Tracks\TrackEntry\FlagLacing
minOccurs / maxOccurs: 1 / 1
definition: Set to 1 if the track **MAY** contain blocks that use lacing. When set to 0, all blocks **MUST** have their lacing flags set to "no lacing"; see [Section 10.3](#) on Block Lacing.

5.1.4.1.13. DefaultDuration Element

id / type: 0x23E383 / uinteger
range: not 0
path: \Segment\Tracks\TrackEntry\DefaultDuration
maxOccurs: 1
definition: Number of nanoseconds per frame, expressed in Matroska Ticks -- i.e., in nanoseconds; see [Section 11.1](#) ("frame" in the Matroska sense -- one Element put into a (Simple)Block).
stream copy: True ([Section 8](#))

5.1.4.1.14. DefaultDecodedFieldDuration Element

id / type: 0x234E7A / uinteger
range: not 0
path: \Segment\Tracks\TrackEntry\DefaultDecodedFieldDuration
maxOccurs: 1
minver: 4
definition: The period between two successive fields at the output of the decoding process, expressed in Matroska Ticks -- i.e., in nanoseconds; see [Section 11.1](#). See [Section 9](#) for more information
stream copy: True ([Section 8](#))

5.1.4.1.15. TrackTimestampScale Element

id / type / default: 0x23314F / float / 0x1p+0
range: > 0x0p+0

path: \Segment\Tracks\TrackEntry\TrackTimestampScale
minOccurs / maxOccurs: 1 / 1
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).
stream copy: True ([Section 8](#))

5.1.4.1.16. MaxBlockAdditionID Element

id / type / default: 0x55EE / uinteger / 0
path: \Segment\Tracks\TrackEntry\MaxBlockAdditionID
minOccurs / maxOccurs: 1 / 1
definition: The maximum value of BlockAddID ([Section 5.1.3.5.2.3](#)). A value of 0 means there is no BlockAdditions ([Section 5.1.3.5.2](#)) for this track.

5.1.4.1.17. BlockAdditionMapping Element

id / type: 0x41E4 / master
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 5.1.3.5.2.3](#)), or to the track as a whole with BlockAddIDExtraData.

5.1.4.1.17.1. BlockAddIDValue Element

id / type: 0x41F0 / uinteger
range: >=2
path: \Segment\Tracks\TrackEntry\BlockAdditionMapping\BlockAddIDValue
maxOccurs: 1
minver: 4
definition: If the track format extension needs content beside frames, the value refers to the BlockAddID ([Section 5.1.3.5.2.3](#)) value being described.
usage notes: To keep MaxBlockAdditionID as low as possible, small values **SHOULD** be used.

5.1.4.1.17.2. BlockAddIDName Element

id / type: 0x41A4 / string
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.

5.1.4.1.17.3. BlockAddIDType Element

id / type / default: 0x41E7 / uinteger / 0

path: \Segment\Tracks\TrackEntry\BlockAdditionalMapping\BlockAddIDType

minOccurs / maxOccurs: 1 / 1

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.

5.1.4.1.17.4. BlockAddIDExtraData Element

id / type: 0x41ED / binary

path: \Segment\Tracks\TrackEntry\BlockAdditionalMapping\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.

5.1.4.1.18. Name Element

id / type: 0x536E / utf-8

path: \Segment\Tracks\TrackEntry\Name

maxOccurs: 1

definition: A human-readable track name.

5.1.4.1.19. Language Element

id / type / default: 0x22B59C / string / eng

path: \Segment\Tracks\TrackEntry\Language

minOccurs / maxOccurs: 1 / 1

definition: 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.

5.1.4.1.20. LanguageBCP47 Element

id / type: 0x22B59D / string

path: \Segment\Tracks\TrackEntry\LanguageBCP47

maxOccurs: 1

minver: 4

definition: The language of the track, in the form defined in [BCP47]; see [Section 12](#) on language codes. If this Element is used, then any Language Elements used in the same TrackEntry **MUST** be ignored.

5.1.4.1.21. CodecID Element

id / type: 0x86 / string

path: \Segment\Tracks\TrackEntry\CodecID

minOccurs / maxOccurs: 1 / 1

definition: An ID corresponding to the codec; see [[MatroskaCodec](#)] for more info.

stream copy: True ([Section 8](#))

5.1.4.1.22. CodecPrivate Element

id / type: 0x63A2 / binary

path: \Segment\Tracks\TrackEntry\CodecPrivate

maxOccurs: 1

definition: Private data only known to the codec.

stream copy: True ([Section 8](#))

5.1.4.1.23. CodecName Element

id / type: 0x258688 / utf-8

path: \Segment\Tracks\TrackEntry\CodecName

maxOccurs: 1

definition: A human-readable string specifying the codec.

5.1.4.1.24. AttachmentLink Element

id / type: 0x7446 / uinteger

range: not 0

path: \Segment\Tracks\TrackEntry\AttachmentLink

maxOccurs: 1

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.

5.1.4.1.25. CodecDelay Element

id / type / default: 0x56AA / uinteger / 0

path: \Segment\Tracks\TrackEntry\CodecDelay

minOccurs / maxOccurs: 1 / 1

minver: 4

definition: The built-in delay for the codec, expressed in Matroska Ticks -- i.e., in nanoseconds; see [Section 11.1](#). It represents the number 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.

stream copy: True ([Section 8](#))

5.1.4.1.26. SeekPreRoll Element

id / type / default: 0x56BB / uinteger / 0

path: \Segment\Tracks\TrackEntry\SeekPreRoll

minOccurs / maxOccurs: 1 / 1

minver: 4

definition: After a discontinuity, the duration of the data that the decoder **MUST** decode before the decoded data is valid, expressed in Matroska Ticks -- i.e., in nanoseconds; see [Section 11.1](#).

stream copy: True ([Section 8](#))

5.1.4.1.27. TrackTranslate Element

id / type: 0x6624 / master

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 a 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.

5.1.4.1.27.1. TrackTranslateTrackID Element

id / type: 0x66A5 / binary

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](#).

5.1.4.1.27.2. TrackTranslateCodec Element

id / type: 0x66BF / uinteger

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

minOccurs / maxOccurs: 1 / 1

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

defined values: See [Table 4](#).

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

5.1.4.1.27.3. TrackTranslateEditionUID Element

id / type: 0x66FC / uinteger

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

definition: Specifies a chapter edition UID to 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.

5.1.4.1.28. Video Element

id / type: 0xE0 / master

path: \Segment\Tracks\TrackEntry\Video

maxOccurs: 1

definition: Video settings.

5.1.4.1.28.1. FlagInterlaced Element

id / type / default: 0x9A / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1

minver: 2

definition: Specifies whether the video frames in this track are interlaced.

defined values: See [Table 5](#).

stream copy: True ([Section 8](#))

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

Table 5: FlagInterlaced Values

5.1.4.1.28.2. FieldOrder Element

id / type / default: 0x9D / uinteger / 2

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

minOccurs / maxOccurs: 1 / 1

minver: 4

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

defined values: See [Table 6](#).

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

stream copy: True ([Section 8](#))

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

5.1.4.1.28.3. StereoMode Element

id / type / default: 0x53B8 / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1

minver: 3

definition: Stereo-3D video mode. See [Section 18.10](#) for more details.

restrictions: See [Table 7](#).

stream copy: True ([Section 8](#))

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

value	label
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

5.1.4.1.28.4. AlphaMode Element

id / type / default: 0x53C0 / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1

minver: 3

definition: Indicates whether the BlockAdditional Element with BlockAddID of "1" contains Alpha data, as defined by 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: See [Table 8](#).

stream copy: True ([Section 8](#))

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

5.1.4.1.28.5. OldStereoMode Element

id / type: 0x53B9 / uinteger

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

maxOccurs: 1

maxver: 2

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

restrictions: See [Table 9](#).

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

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

Table 9: *OldStereoMode Values*

5.1.4.1.28.6. PixelWidth Element

id / type: 0xB0 / uinteger
range: not 0
path: \Segment\Tracks\TrackEntry\Video\PixelWidth
minOccurs / maxOccurs: 1 / 1
definition: Width of the encoded video frames in pixels.
stream copy: True ([Section 8](#))

5.1.4.1.28.7. PixelHeight Element

id / type: 0xBA / uinteger
range: not 0
path: \Segment\Tracks\TrackEntry\Video\PixelHeight
minOccurs / maxOccurs: 1 / 1
definition: Height of the encoded video frames in pixels.
stream copy: True ([Section 8](#))

5.1.4.1.28.8. PixelCropBottom Element

id / type / default: 0x54AA / uinteger / 0
path: \Segment\Tracks\TrackEntry\Video\PixelCropBottom
minOccurs / maxOccurs: 1 / 1
definition: The number of video pixels to remove at the bottom of the image.
stream copy: True ([Section 8](#))

5.1.4.1.28.9. PixelCropTop Element

id / type / default: 0x54BB / uinteger / 0
path: \Segment\Tracks\TrackEntry\Video\PixelCropTop
minOccurs / maxOccurs: 1 / 1
definition: The number of video pixels to remove at the top of the image.
stream copy: True ([Section 8](#))

5.1.4.1.28.10. PixelCropLeft Element

id / type / default: 0x54CC / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1

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

stream copy: True ([Section 8](#))

5.1.4.1.28.11. PixelCropRight Element

id / type / default: 0x54DD / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1

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

stream copy: True ([Section 8](#))

5.1.4.1.28.12. DisplayWidth Element

id / type: 0x54B0 / uinteger

range: not 0

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

maxOccurs: 1

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

notes: See [Table 10](#).

stream copy: True ([Section 8](#))

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

Table 10: DisplayWidth Implementation Notes

5.1.4.1.28.13. DisplayHeight Element

id / type: 0x54BA / uinteger

range: not 0

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

maxOccurs: 1

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

notes: See [Table 11](#).

stream copy: True ([Section 8](#))

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

Table 11: DisplayHeight Implementation Notes

5.1.4.1.28.14. DisplayUnit Element

id / type / default: 0x54B2 / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1

definition: How DisplayWidth and DisplayHeight are interpreted.

restrictions: See [Table 12](#).

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

Table 12: DisplayUnit Values

5.1.4.1.28.15. UncompressedFourCC Element

id / type: 0x2EB524 / binary

length: 4

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

minOccurs / maxOccurs: see implementation notes / 1

definition: Specifies 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](#)]. There is neither a definitive list of FourCC values nor an official registry. Some common values for YUV pixel formats can be found at [[MSYUV8](#)], [[MSYUV16](#)], and [[FourCC-YUV](#)]. Some common values for uncompressed RGB pixel formats can be found at [[MSRGB](#)] and [[FourCC-RGB](#)].

notes: See [Table 13](#).

stream copy: True ([Section 8](#))

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

5.1.4.1.28.16. Colour Element

id / type: 0x55B0 / master
 path: \Segment\Tracks\TrackEntry\Video\Colour
 maxOccurs: 1
 minver: 4
 definition: Settings describing the colour format.
 stream copy: True ([Section 8](#))

5.1.4.1.28.17. MatrixCoefficients Element

id / type / default: 0x55B1 / uinteger / 2
 path: \Segment\Tracks\TrackEntry\Video\Colour\MatrixCoefficients
 minOccurs / maxOccurs: 1 / 1
 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 [\[ITU-H.273\]](#).
 restrictions: See [Table 14](#).
 stream copy: True ([Section 8](#))

value	label
0	Identity
1	ITU-R BT.709
2	unspecified
3	reserved
4	US FCC 73.682
5	ITU-R BT.470BG
6	SMPTE 170M
7	SMPTE 240M
8	YCoCg

value	label
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

5.1.4.1.28.18. BitsPerChannel Element

id / type / default: 0x55B2 / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1

minver: 4

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

stream copy: True ([Section 8](#))

5.1.4.1.28.19. ChromaSubsamplingHorz Element

id / type: 0x55B3 / uinteger

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

maxOccurs: 1

minver: 4

definition: The number 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.

stream copy: True ([Section 8](#))

5.1.4.1.28.20. ChromaSubsamplingVert Element

id / type: 0x55B4 / uinteger

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

maxOccurs: 1

minver: 4

definition: The number 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.

stream copy: True ([Section 8](#))

5.1.4.1.28.21. CbSubsamplingHorz Element

id / type: 0x55B5 / uinteger

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

maxOccurs: 1

minver: 4

definition: The number 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.

stream copy: True ([Section 8](#))

5.1.4.1.28.22. CbSubsamplingVert Element

id / type: 0x55B6 / uinteger

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

maxOccurs: 1

minver: 4

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

stream copy: True ([Section 8](#))

5.1.4.1.28.23. ChromaSitingHorz Element

id / type / default: 0x55B7 / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1

minver: 4

definition: How chroma is subsampled horizontally.

restrictions: See [Table 15](#).

stream copy: True ([Section 8](#))

value	label
0	unspecified
1	left collocated
2	half

Table 15: ChromaSitingHorz Values

5.1.4.1.28.24. ChromaSitingVert Element

id / type / default: 0x55B8 / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1
 minver: 4
 definition: How chroma is subsampled vertically.
 restrictions: See [Table 16](#).
 stream copy: True ([Section 8](#))

value	label
0	unspecified
1	top collocated
2	half

Table 16: ChromaSitingVert Values

5.1.4.1.28.25. Range Element

id / type / default: 0x55B9 / uinteger / 0
 path: \Segment\Tracks\TrackEntry\Video\Colour\Range
 minOccurs / maxOccurs: 1 / 1
 minver: 4
 definition: Clipping of the color ranges.
 restrictions: See [Table 17](#).
 stream copy: True ([Section 8](#))

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

Table 17: Range Values

5.1.4.1.28.26. TransferCharacteristics Element

id / type / default: 0x55BA / uinteger / 2
 path: \Segment\Tracks\TrackEntry\Video\Colour\TransferCharacteristics
 minOccurs / maxOccurs: 1 / 1
 minver: 4
 definition: The transfer characteristics of the video. For clarity, the value and meanings for TransferCharacteristics are adopted from Table 3 of [\[ITU-H.273\]](#).
 restrictions: See [Table 18](#).
 stream copy: True ([Section 8](#))

value	label
0	reserved
1	ITU-R BT.709
2	unspecified
3	reserved2
4	Gamma 2.2 curve - BT.470M
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

5.1.4.1.28.27. Primaries Element

id / type / default: 0x55BB / uinteger / 2

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

minOccurs / maxOccurs: 1 / 1

minver: 4

definition: The colour primaries of the video. For clarity, the value and meanings for Primaries are adopted from Table 2 of [\[ITU-H.273\]](#).

restrictions: See [Table 19](#).
 stream copy: True ([Section 8](#))

value	label
0	reserved
1	ITU-R BT.709
2	unspecified
3	reserved2
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
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

5.1.4.1.28.28. MaxCLL Element

id / type: 0x55BC / uinteger

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

maxOccurs: 1

minver: 4

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

stream copy: True ([Section 8](#))

5.1.4.1.28.29. MaxFALL Element

id / type: 0x55BD / uinteger

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

maxOccurs: 1
minver: 4
definition: Maximum brightness of a single full frame (Maximum Frame-Average Light Level) in candelas per square meter (cd/m²).
stream copy: True ([Section 8](#))

5.1.4.1.28.30. MasteringMetadata Element

id / type: 0x55D0 / master
path: \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata
maxOccurs: 1
minver: 4
definition: SMPTE 2086 mastering data.
stream copy: True ([Section 8](#))

5.1.4.1.28.31. PrimaryRChromaticityX Element

id / type: 0x55D1 / float
range: 0x0p+0-0x1p+0
path:
 \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\PrimaryRChromaticity
 X
maxOccurs: 1
minver: 4
definition: Red X chromaticity coordinate, as defined by [[CIE-1931](#)].
stream copy: True ([Section 8](#))

5.1.4.1.28.32. PrimaryRChromaticityY Element

id / type: 0x55D2 / float
range: 0x0p+0-0x1p+0
path:
 \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\PrimaryRChromaticity
 Y
maxOccurs: 1
minver: 4
definition: Red Y chromaticity coordinate, as defined by [[CIE-1931](#)].
stream copy: True ([Section 8](#))

5.1.4.1.28.33. PrimaryGChromaticityX Element

id / type: 0x55D3 / float
range: 0x0p+0-0x1p+0

path:

```
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\PrimaryGChromaticity  
X
```

maxOccurs: 1

minver: 4

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

stream copy: True (Section 8)

5.1.4.1.28.34. PrimaryGChromaticityY Element

id / type: 0x55D4 / float

range: 0x0p+0-0x1p+0

path:

```
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\PrimaryGChromaticity  
Y
```

maxOccurs: 1

minver: 4

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

stream copy: True (Section 8)

5.1.4.1.28.35. PrimaryBChromaticityX Element

id / type: 0x55D5 / float

range: 0x0p+0-0x1p+0

path:

```
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\PrimaryBChromaticity  
X
```

maxOccurs: 1

minver: 4

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

stream copy: True (Section 8)

5.1.4.1.28.36. PrimaryBChromaticityY Element

id / type: 0x55D6 / float

range: 0x0p+0-0x1p+0

path:

```
\Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\PrimaryBChromaticity  
Y
```

maxOccurs: 1

minver: 4

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

stream copy: True (Section 8)

5.1.4.1.28.37. WhitePointChromaticityX Element

id / type: 0x55D7 / float
range: 0x0p+0-0x1p+0
path: \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\WhitePointChromaticityX
maxOccurs: 1
minver: 4
definition: White X chromaticity coordinate, as defined by [CIE-1931].
stream copy: True (Section 8)

5.1.4.1.28.38. WhitePointChromaticityY Element

id / type: 0x55D8 / float
range: 0x0p+0-0x1p+0
path: \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\WhitePointChromaticityY
maxOccurs: 1
minver: 4
definition: White Y chromaticity coordinate, as defined by [CIE-1931].
stream copy: True (Section 8)

5.1.4.1.28.39. LuminanceMax Element

id / type: 0x55D9 / float
range: $\geq 0x0p+0$
path: \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\LuminanceMax
maxOccurs: 1
minver: 4
definition: Maximum luminance. Represented in candelas per square meter (cd/m^2).
stream copy: True (Section 8)

5.1.4.1.28.40. LuminanceMin Element

id / type: 0x55DA / float
range: $\geq 0x0p+0$
path: \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\LuminanceMin
maxOccurs: 1
minver: 4
definition: Minimum luminance. Represented in candelas per square meter (cd/m^2).
stream copy: True (Section 8)

5.1.4.1.28.41. Projection Element

id / type: 0x7670 / master

path: \Segment\Tracks\TrackEntry\Video\Projection
 maxOccurs: 1
 minver: 4
 definition: Describes the video projection details. Used to render spherical or VR videos or to flip videos horizontally or vertically.
 stream copy: True ([Section 8](#))

5.1.4.1.28.42. ProjectionType Element

id / type / default: 0x7671 / uinteger / 0
 path: \Segment\Tracks\TrackEntry\Video\Projection\ProjectionType
 minOccurs / maxOccurs: 1 / 1
 minver: 4
 definition: Describes the projection used for this video track.
 restrictions: See [Table 20](#).
 stream copy: True ([Section 8](#))

value	label
0	rectangular
1	equirectangular
2	cubemap
3	mesh

Table 20: ProjectionType Values

5.1.4.1.28.43. ProjectionPrivate Element

id / type: 0x7672 / binary
 path: \Segment\Tracks\TrackEntry\Video\Projection\ProjectionPrivate
 maxOccurs: 1
 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.
stream copy: True ([Section 8](#))

5.1.4.1.28.44. ProjectionPoseYaw Element

id / type / default: 0x7673 / float / 0x0p+0
range: >= -0xB4p+0, <= 0xB4p+0
path: \Segment\Tracks\TrackEntry\Video\Projection\ProjectionPoseYaw
minOccurs / maxOccurs: 1 / 1
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 ProjectionPoseRoll and ProjectionPosePitch set to 0 degrees flips the image horizontally.

stream copy: True ([Section 8](#))

5.1.4.1.28.45. ProjectionPosePitch Element

id / type / default: 0x7674 / float / 0x0p+0
range: >= -0x5Ap+0, <= 0x5Ap+0
path: \Segment\Tracks\TrackEntry\Video\Projection\ProjectionPosePitch
minOccurs / maxOccurs: 1 / 1
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.

stream copy: True ([Section 8](#))

5.1.4.1.28.46. ProjectionPoseRoll Element

id / type / default: 0x7675 / float / 0x0p+0
range: >= -0xB4p+0, <= 0xB4p+0
path: \Segment\Tracks\TrackEntry\Video\Projection\ProjectionPoseRoll
minOccurs / maxOccurs: 1 / 1
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 and `ProjectionPoseYaw` to 180 or -180 degrees with `ProjectionPosePitch` set to 0 degrees flips the image vertically.

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

stream copy: True ([Section 8](#))

5.1.4.1.29. Audio Element

id / type: 0xE1 / master
 path: \Segment\Tracks\TrackEntry\Audio
 maxOccurs: 1
 definition: Audio settings.

5.1.4.1.29.1. SamplingFrequency Element

id / type / default: 0xB5 / float / 0x1.f4p+12
 range: > 0x0p+0
 path: \Segment\Tracks\TrackEntry\Audio\SamplingFrequency
 minOccurs / maxOccurs: 1 / 1
 definition: Sampling frequency in Hz.
 stream copy: True ([Section 8](#))

5.1.4.1.29.2. OutputSamplingFrequency Element

id / type: 0x78B5 / float
 range: > 0x0p+0
 path: \Segment\Tracks\TrackEntry\Audio\OutputSamplingFrequency
 maxOccurs: 1
 definition: Real output sampling frequency in Hz (used for SBR techniques).
 notes: See [Table 21](#).

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

Table 21: *OutputSamplingFrequency Implementation Notes*

5.1.4.1.29.3. Channels Element

id / type / default: 0x9F / uinteger / 1
range: not 0
path: \Segment\Tracks\TrackEntry\Audio\Channels
minOccurs / maxOccurs: 1 / 1
definition: Numbers of channels in the track.
stream copy: True ([Section 8](#))

5.1.4.1.29.4. BitDepth Element

id / type: 0x6264 / uinteger
range: not 0
path: \Segment\Tracks\TrackEntry\Audio\BitDepth
maxOccurs: 1
definition: Bits per sample, mostly used for PCM.
stream copy: True ([Section 8](#))

5.1.4.1.30. TrackOperation Element

id / type: 0xE2 / master
path: \Segment\Tracks\TrackEntry\TrackOperation
maxOccurs: 1
minver: 3
definition: Operation that needs to be applied on tracks to create this virtual track. For more details, see [Section 18.8](#).
stream copy: True ([Section 8](#))

5.1.4.1.30.1. TrackCombinePlanes Element

id / type: 0xE3 / master
path: \Segment\Tracks\TrackEntry\TrackOperation\TrackCombinePlanes
maxOccurs: 1
minver: 3
definition: Contains the list of all video plane tracks that need to be combined to create this 3D track.
stream copy: True ([Section 8](#))

5.1.4.1.30.2. TrackPlane Element

id / type: 0xE4 / master
path: \Segment\Tracks\TrackEntry\TrackOperation\TrackCombinePlanes\TrackPlane
minOccurs: 1
minver: 3
definition: Contains a video plane track that needs to be combined to create this 3D track.

stream copy: True ([Section 8](#))

5.1.4.1.30.3. TrackPlaneUID Element

id / type: 0xE5 / uinteger

range: not 0

path:

```
\Segment\Tracks\TrackEntry\TrackOperation\TrackCombinePlanes\TrackPlane\TrackP
laneUID
```

minOccurs / maxOccurs: 1 / 1

minver: 3

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

stream copy: True ([Section 8](#))

5.1.4.1.30.4. TrackPlaneType Element

id / type: 0xE6 / uinteger

path:

```
\Segment\Tracks\TrackEntry\TrackOperation\TrackCombinePlanes\TrackPlane\TrackP
laneType
```

minOccurs / maxOccurs: 1 / 1

minver: 3

definition: The kind of plane this track corresponds to.

restrictions: See [Table 22](#).

stream copy: True ([Section 8](#))

value	label
0	left eye
1	right eye
2	background

Table 22: TrackPlaneType Values

5.1.4.1.30.5. TrackJoinBlocks Element

id / type: 0xE9 / master

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

maxOccurs: 1

minver: 3

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

stream copy: True ([Section 8](#))

5.1.4.1.30.6. TrackJoinUID Element

id / type: 0xED / uinteger

range: not 0

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

minOccurs: 1

minver: 3

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

stream copy: True ([Section 8](#))

5.1.4.1.31. ContentEncodings Element

id / type: 0x6D80 / master

path: \Segment\Tracks\TrackEntry\ContentEncodings

maxOccurs: 1

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

stream copy: True ([Section 8](#))

5.1.4.1.31.1. ContentEncoding Element

id / type: 0x6240 / master

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

minOccurs: 1

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

stream copy: True ([Section 8](#))

5.1.4.1.31.2. ContentEncodingOrder Element

id / type / default: 0x5031 / uinteger / 0

path:

\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncodingOrder

minOccurs / maxOccurs: 1 / 1

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 for each ContentEncoding found in the ContentEncodings of this TrackEntry.

stream copy: True ([Section 8](#))

5.1.4.1.31.3. ContentEncodingScope Element

id / type / default: 0x5032 / uinteger / 1

path:

\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncodingScope

minOccurs / maxOccurs: 1 / 1

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

defined values: See [Table 23](#).

stream copy: True ([Section 8](#))

value	label	definition
1	Block	All frame contents, excluding lacing data.
2	Private	The track's CodecPrivate data.
4	Next	The next ContentEncoding (next ContentEncodingOrder; the data inside ContentCompression and/or ContentEncryption). This value SHOULD NOT be used, as it's not supported by players.

Table 23: ContentEncodingScope Values

5.1.4.1.31.4. ContentEncodingType Element

id / type / default: 0x5033 / uinteger / 0

path:

\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncodingType

minOccurs / maxOccurs: 1 / 1

definition: A value describing the kind of transformation that is applied.

restrictions: See [Table 24](#).

stream copy: True ([Section 8](#))

value	label
0	Compression
1	Encryption

Table 24: ContentEncodingType Values

5.1.4.1.31.5. ContentCompression Element

id / type: 0x5034 / master

path:

\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentCompression

maxOccurs: 1

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 to not prevent seeking.

stream copy: True (Section 8)

5.1.4.1.31.6. ContentCompAlgo Element

id / type / default: 0x4254 / uinteger / 0

path:

```
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentCompression
\ContentCompAlgo
```

minOccurs / maxOccurs: 1 / 1

definition: The compression algorithm used.

defined values: See Table 25.

usage notes: Compression method "1" (bzlib) and "2" (lzo1x) lack 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. A Matroska Writer **SHOULD NOT** use these compression methods by default. A Matroska Reader **MAY** support methods "1" and "2" as possible and **SHOULD** support other methods.

stream copy: True (Section 8)

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 5.1.4.1.31.7) have been stripped from each frame.

Table 25: ContentCompAlgo Values

5.1.4.1.31.7. ContentCompSettings Element

id / type: 0x4255 / binary

path:

```
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentCompression
\ContentCompSettings
```

maxOccurs: 1

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

stream copy: True (Section 8)

5.1.4.1.31.8. ContentEncryption Element

id / type: 0x5035 / master

path:

\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption

maxOccurs: 1

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

stream copy: True (Section 8)

5.1.4.1.31.9. ContentEncAlgo Element

id / type / default: 0x47E1 / uinteger / 0

path:

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

minOccurs / maxOccurs: 1 / 1

definition: The encryption algorithm used.

defined values: See Table 26.

stream copy: True (Section 8)

value	label	definition
0	Not encrypted	The data are not encrypted.
1	DES	Data Encryption Standard (DES) [FIPS46-3]. This value SHOULD be avoided.
2	3DES	Triple Data Encryption Algorithm [SP800-67]. This value SHOULD be avoided.
3	Twofish	Twofish Encryption Algorithm [Twofish].
4	Blowfish	Blowfish Encryption Algorithm [Blowfish]. This value SHOULD be avoided.
5	AES	Advanced Encryption Standard (AES) [FIPS197].

Table 26: ContentEncAlgo Values

5.1.4.1.31.10. ContentEncKeyID Element

id / type: 0x47E2 / binary

path:

```
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\
ContentEncKeyID
```

maxOccurs: 1

definition: For public key algorithms, the ID of the public key that the data was encrypted with.

stream copy: True ([Section 8](#))

5.1.4.1.31.11. ContentEncAESSettings Element

id / type: 0x47E7 / master

path:

```
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\
ContentEncAESSettings
```

maxOccurs: 1

minver: 4

definition: Settings describing the encryption algorithm used.

notes: See [Table 27](#).

stream copy: True ([Section 8](#))

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

Table 27: ContentEncAESSettings Implementation Notes

5.1.4.1.31.12. AESSettingsCipherMode Element

id / type: 0x47E8 / uinteger

path:

```
\Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\
ContentEncAESSettings\AESSettingsCipherMode
```

minOccurs / maxOccurs: 1 / 1

minver: 4

definition: The AES cipher mode used in the encryption.

defined values: See [Table 28](#).

notes: See [Table 29](#).

stream copy: True ([Section 8](#))

value	label	definition
1	AES-CTR	Counter [SP800-38A]
2	AES-CBC	Cipher Block Chaining [SP800-38A]

Table 28: AESSettingsCipherMode Values

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

id / type: 0x1C53BB6B / master

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: See [Table 30](#).

attribute	note
minOccurs	This Element SHOULD be set when the Segment is not transmitted as a live stream; see Section 23.2 .

Table 30: Cues Implementation Notes

5.1.5.1. CuePoint Element

id / type: 0xBB / master

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

id / type: 0xB3 / uinteger

path: \Segment\Cues\CuePoint\CueTime

minOccurs / maxOccurs: 1 / 1

definition: Absolute timestamp of the seek point, expressed in Matroska Ticks -- i.e., in nanoseconds; see [Section 11.1](#).

5.1.5.1.2. CueTrackPositions Element

id / type: 0xB7 / master

path: \Segment\Cues\CuePoint\CueTrackPositions

minOccurs: 1

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

5.1.5.1.2.1. CueTrack Element

id / type: 0xF7 / uinteger
range: not 0
path: \Segment\Cues\CuePoint\CueTrackPositions\CueTrack
minOccurs / maxOccurs: 1 / 1
definition: The track for which a position is given.

5.1.5.1.2.2. CueClusterPosition Element

id / type: 0xF1 / uinteger
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

id / type: 0xF0 / uinteger
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

id / type: 0xB2 / uinteger
path: \Segment\Cues\CuePoint\CueTrackPositions\CueDuration
maxOccurs: 1
minver: 4
definition: The duration of the block, expressed in Segment Ticks, which are 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

id / type: 0x5378 / uinteger
range: not 0
path: \Segment\Cues\CuePoint\CueTrackPositions\CueBlockNumber
maxOccurs: 1
definition: Number of the Block in the specified Cluster.

5.1.5.1.2.6. CueCodecState Element

id / type / default: 0xEA / uinteger / 0
path: \Segment\Cues\CuePoint\CueTrackPositions\CueCodecState
minOccurs / maxOccurs: 1 / 1

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

id / type: 0xDB / master

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

minver: 2

definition: The Clusters containing the referenced Blocks.

5.1.5.1.2.8. CueRefTime Element

id / type: 0x96 / uinteger

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

minOccurs / maxOccurs: 1 / 1

minver: 2

definition: Timestamp of the referenced Block, expressed in Matroska Ticks -- i.e., in nanoseconds; see [Section 11.1](#).

5.1.6. Attachments Element

id / type: 0x1941A469 / master

path: \Segment\Attachments

maxOccurs: 1

definition: Contains attached files.

5.1.6.1. AttachedFile Element

id / type: 0x61A7 / master

path: \Segment\Attachments\AttachedFile

minOccurs: 1

definition: An attached file.

5.1.6.1.1. FileDescription Element

id / type: 0x467E / utf-8

path: \Segment\Attachments\AttachedFile\FileDescription

maxOccurs: 1

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

5.1.6.1.2. FileName Element

id / type: 0x466E / utf-8

path: \Segment\Attachments\AttachedFile\FileName

minOccurs / maxOccurs: 1 / 1
definition: Filename of the attached file.

5.1.6.1.3. FileMediaType Element

id / type: 0x4660 / string
path: \Segment\Attachments\AttachedFile\FileMediaType
minOccurs / maxOccurs: 1 / 1
definition: Media type of the file following the format described in [RFC6838].
stream copy: True (Section 8)

5.1.6.1.4. FileData Element

id / type: 0x465C / binary
path: \Segment\Attachments\AttachedFile\FileData
minOccurs / maxOccurs: 1 / 1
definition: The data of the file.
stream copy: True (Section 8)

5.1.6.1.5. FileUID Element

id / type: 0x46AE / uinteger
range: not 0
path: \Segment\Attachments\AttachedFile\FileUID
minOccurs / maxOccurs: 1 / 1
definition: UID representing the file, as random as possible.
stream copy: True (Section 8)

5.1.7. Chapters Element

id / type: 0x1043A770 / master
path: \Segment\Chapters
maxOccurs: 1
recurring: True
definition: A system to define basic menus and partition data. For more detailed information, see Section 20.

5.1.7.1. EditionEntry Element

id / type: 0x45B9 / master
path: \Segment\Chapters\EditionEntry
minOccurs: 1
definition: Contains all information about a Segment edition.

5.1.7.1.1. EditionUID Element

id / type: 0x45BC / uinteger
range: not 0
path: \Segment\Chapters\EditionEntry\EditionUID
maxOccurs: 1
definition: A UID to identify the edition. It's useful for tagging an edition.
stream copy: True ([Section 8](#))

5.1.7.1.2. EditionFlagDefault Element

id / type / default: 0x45DB / uinteger / 0
range: 0-1
path: \Segment\Chapters\EditionEntry\EditionFlagDefault
minOccurs / maxOccurs: 1 / 1
definition: Set to 1 if the edition **SHOULD** be used as the default one.

5.1.7.1.3. EditionFlagOrdered Element

id / type / default: 0x45DD / uinteger / 0
range: 0-1
path: \Segment\Chapters\EditionEntry\EditionFlagOrdered
minOccurs / maxOccurs: 1 / 1
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

id / type: 0xB6 / master
path: \Segment\Chapters\EditionEntry\+ChapterAtom
minOccurs: 1
recursive: True
definition: Contains the atom information to use as the chapter atom (applies to all tracks).

5.1.7.1.4.1. ChapterUID Element

id / type: 0x73C4 / uinteger
range: not 0
path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterUID
minOccurs / maxOccurs: 1 / 1
definition: A UID to identify the Chapter.
stream copy: True ([Section 8](#))

5.1.7.1.4.2. ChapterStringUID Element

id / type: 0x5654 / utf-8
path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterStringUID

maxOccurs: 1
 minver: 3
 definition: A unique string ID to identify the Chapter. For example, it is used as the storage for cue identifier values [[WebVTT](#)].

5.1.7.1.4.3. ChapterTimeStart Element

id / type: 0x91 / uinteger
 path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterTimeStart
 minOccurs / maxOccurs: 1 / 1
 definition: Timestamp of the start of Chapter, expressed in Matroska Ticks -- i.e., in nanoseconds; see [Section 11.1](#).

5.1.7.1.4.4. ChapterTimeEnd Element

id / type: 0x92 / uinteger
 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 -- i.e., in nanoseconds; see [Section 11.1](#). The value **MUST** be greater than or equal to the ChapterTimeStart of the same ChapterAtom.
 usage notes: With the ChapterTimeEnd timestamp value being excluded, it **MUST** take into account the duration of the last frame it includes, especially for the ChapterAtom using the last frames of the Segment.
 notes: See [Table 31](#).

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

Table 31: ChapterTimeEnd Implementation Notes

5.1.7.1.4.5. ChapterFlagHidden Element

id / type / default: 0x98 / uinteger / 0
 range: 0-1
 path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterFlagHidden
 minOccurs / maxOccurs: 1 / 1
 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

id / type: 0x6E67 / binary
 length: 16

path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterSegmentUUID
 minOccurs / maxOccurs: see implementation notes / 1
 definition: The SegmentUUID of another Segment to play during this chapter.
 usage notes: The value **MUST NOT** be the SegmentUUID value of the Segment it belongs to.
 notes: See [Table 32](#).

attribute	note
minOccurs	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

id / type: 0x6EBC / uinteger
 range: not 0
 path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterSegmentEditionUID
 maxOccurs: 1
 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

id / type: 0x63C3 / uinteger
 path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterPhysicalEquiv
 maxOccurs: 1
 definition: Specifies the physical equivalent of this ChapterAtom, e.g., "DVD" (60) or "SIDE" (50); see [Section 20.4](#) for a complete list of values.

5.1.7.1.4.9. ChapterDisplay Element

id / type: 0x80 / master
 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

id / type: 0x85 / utf-8
 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

id / type / default: 0x437C / string / eng

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

minOccurs: 1

definition: A language corresponding to the string, in the Matroska languages form; see [Section 12](#) on language codes. This Element **MUST** be ignored if a ChapLanguageBCP47 Element is used within the same ChapterDisplay Element.

5.1.7.1.4.12. ChapLanguageBCP47 Element

id / type: 0x437D / string

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

minver: 4

definition: A language corresponding to the ChapString, in the form defined in [BCP47]; see [Section 12](#) on language codes. 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

id / type: 0x437E / string

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

definition: A country corresponding to the string, in the Matroska countries form; see [Section 13](#) on country 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

id / type: 0x6944 / master

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

definition: Contains all the commands associated to the Atom.

5.1.7.1.4.15. ChapProcessCodecID Element

id / type / default: 0x6955 / uinteger / 0

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

minOccurs / maxOccurs: 1 / 1

definition: Contains the type of the codec used for processing. A value of 0 means built-in Matroska processing (to be defined), and 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

id / type: 0x450D / binary

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

maxOccurs: 1

definition: 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

id / type: 0x6911 / master

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

definition: Contains all the commands associated with the Atom.

5.1.7.1.4.18. ChapProcessTime Element

id / type: 0x6922 / uinteger

path: \Segment\Chapters\EditionEntry\

+ChapterAtom\ChapProcess\ChapProcessCommand\ChapProcessTime

minOccurs / maxOccurs: 1 / 1

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

restrictions: See [Table 33](#).

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

id / type: 0x6933 / binary

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

id / type: 0x1254C367 / master

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

id / type: 0x7373 / master
 path: \Segment\Tags\Tag
 minOccurs: 1
 definition: A single metadata descriptor.

5.1.8.1.1. Targets Element

id / type: 0x63C0 / master
 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 omitted, then the Tag describes everything in the Segment.

5.1.8.1.1.1. TargetTypeValue Element

id / type / default: 0x68CA / uinteger / 50
 path: \Segment\Tags\Tag\Targets\TargetTypeValue
 minOccurs / maxOccurs: 1 / 1
 definition: A number to indicate the logical level of the target.
 defined values: See [Table 34](#).

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 (e.g., 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 / MOVEMENT / SCENE	Corresponds to parts of a track for audio, such as a movement or scene in a movie.
10	SHOT	The lowest hierarchy found in music or movies.

Table 34: TargetTypeValue Values

5.1.8.1.1.2. TargetType Element

id / type: 0x63CA / string

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

maxOccurs: 1

definition: An informational string that can be used to display the logical level of the target, such as "ALBUM", "TRACK", "MOVIE", "CHAPTER", etc.

restrictions: See [Table 35](#).

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
MOVEMENT	TargetTypeValue 20
SCENE	TargetTypeValue 20

value	label
SHOT	TargetTypeValue 10

Table 35: TargetType Values

5.1.8.1.1.3. TagTrackUID Element

id / type / default: 0x63C5 / uinteger / 0

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

definition: A UID to identify the Track(s) that 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

id / type / default: 0x63C9 / uinteger / 0

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

definition: A UID to identify the EditionEntry(s) that 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

id / type / default: 0x63C4 / uinteger / 0

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

definition: A UID to identify the Chapter(s) that 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

id / type / default: 0x63C6 / uinteger / 0

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

definition: A UID to identify the Attachment(s) that 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

id / type: 0x67C8 / master

path: \Segment\Tags\Tag\+SimpleTag

minOccurs: 1

recursive: True

definition: Contains general information about the target.

5.1.8.1.2.1. TagName Element

id / type: 0x45A3 / utf-8
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

id / type / default: 0x447A / string / und
path: \Segment\Tags\Tag\+SimpleTag\TagLanguage
minOccurs / maxOccurs: 1 / 1
definition: Specifies the language of the specified tag 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

id / type: 0x447B / string
path: \Segment\Tags\Tag\+SimpleTag\TagLanguageBCP47
maxOccurs: 1
minver: 4
definition: The language used in the TagString, in the form defined in [\[BCP47\]](#); see [Section 12](#) on language codes. 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

id / type / default: 0x4484 / uinteger / 1
range: 0-1
path: \Segment\Tags\Tag\+SimpleTag\TagDefault
minOccurs / maxOccurs: 1 / 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

id / type: 0x4487 / utf-8
path: \Segment\Tags\Tag\+SimpleTag\TagString
maxOccurs: 1
definition: The value of the Tag.

5.1.8.1.2.6. TagBinary Element

id / type: 0x4485 / binary

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

With the exceptions of the EBML Header and the CRC-32 Element, the EBML specification [RFC8794] does not require any particular storage order for Elements. However, this specification defines mandates and recommendations for ordering certain Elements 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 either **MUST** be stored before the first Cluster Element or **SHALL** both be referenced by a SeekHead Element occurring before the first Cluster Element.

All Top-Level Elements **MUST** use a 4-octet EBML Element ID.

When using Medium Linking, chapters are used to reference other Segments to play in a given order (see Section 17.2). A Segment containing these Linked Chapters does not require 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) itemizes 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 [RFC8794], 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 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 Element.

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 Elements 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 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 the 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 can 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; thus, 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 and `Segment` sizes will only marginally change.

7. Matroska Versioning

Matroska is based on the principle that a reading application does not have to support 100% of the specifications in order to be able to play the file. Therefore, a Matroska file 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 the `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 5.1.3.4) **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 a file with `DocReadTypeVersion` equal to or lower than *V*, even if `DocTypeVersion` is greater than *V*.

A reading application supporting at least Matroska version *V* and 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 need 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.

An Element is marked with a property: `stream copy: True` when the values of that Element need to be kept identical between the source and destination files. If that property is not set, elements may or may not keep the same value between the source and destination files.

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 ([Section 5.1.4.1.28.1](#)), then `DefaultDecodedFieldDuration` equals the period between two successive fields at the output of the decoding process. For video sequences signaled as progressive, `DefaultDecodedFieldDuration` is half of the period between two successive frames at the output of the decoding process.

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: $1000000000 \text{ ns} / (48 / 1.001) = 20854167 \text{ ns}$
- PAL broadcast/DVD: $1000000000 \text{ ns} / (50 / 1.000) = 20000000 \text{ ns}$
- N/ATSC broadcast: $1000000000 \text{ ns} / (60 / 1.001) = 16683333 \text{ ns}$
- Hard-telecined DVD: $1000000000 \text{ ns} / (60 / 1.001) = 16683333 \text{ ns}$ (60 encoded interlaced fields per second)
- Soft-telecined DVD: $1000000000 \text{ ns} / (60 / 1.001) = 16683333 \text{ ns}$ (48 encoded interlaced fields per second, with "repeat_first_field = 1")

10. Cluster Blocks

Frames using references **SHOULD** be stored in "coding order" (i.e., the references first and then the frames referencing them). A consequence is that timestamps might not be consecutive. However, a frame with a past timestamp **MUST** reference a frame already known; otherwise, it is considered bad/void.

Matroska has two similar ways to store frames in a block:

- in a Block that is contained inside a BlockGroup
- in a SimpleBlock that is directly in the Cluster

The SimpleBlock is usually preferred unless some extra elements of the BlockGroup need to be used. A Matroska Reader **MUST** support both types of blocks.

Each block contains the same parts in the following order:

- a variable-length header
- the lacing information (optional)
- the consecutive frame(s)

The block header starts with the number of the Track it corresponds to. The value **MUST** correspond to the TrackNumber (Section 5.1.4.1.1) of a TrackEntry of the Segment.

The TrackNumber is coded using the Variable-Size Integer (VINT) mechanism described in Section 4 of [RFC8794]. To save space, the shortest VINT form **SHOULD** be used. The value can be coded on up to 8 octets. This is the only element with a variable size in the block header.

The timestamp is expressed in Track Ticks; see Section 11.1. The value is stored as a signed value on 16 bits.

10.1. Block Structure

This section describes the binary data contained in the Block Element (Section 5.1.3.5.1). Bit 0 is the most significant bit.

As the TrackNumber size can vary between 1 and 8 octets, there are 8 different sizes for the Block header. The definitions for TrackNumber sizes of 1 and 2 are provided; the other variants can be deduced by extending the size of the TrackNumber by multiples of 8 bits.

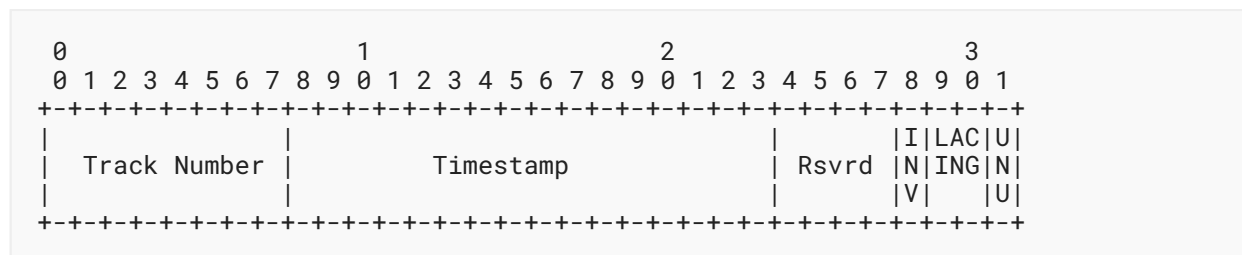


Figure 11: Block Header with 1-Octet TrackNumber

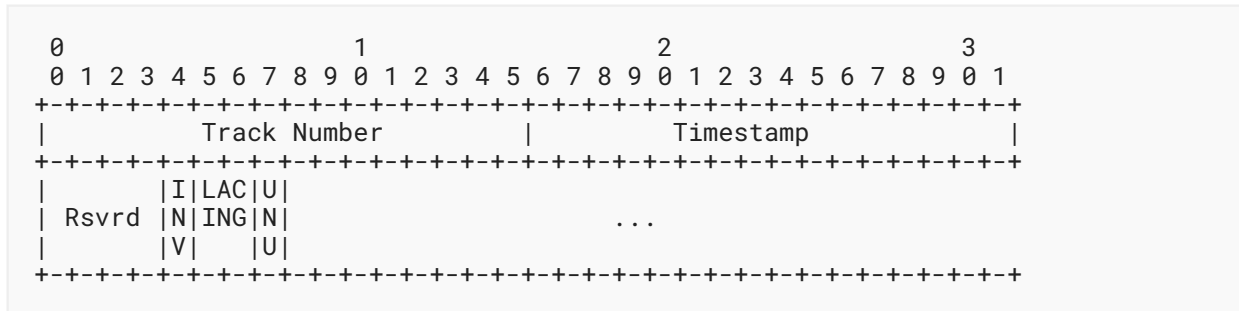


Figure 12: Block Header with 2-Octet TrackNumber

where:

Track Number: 8, 16, 24, 32, 40, 48, or 64 bits. An EBML VINT-coded track number.

Timestamp: 16 bits. Signed timestamp in Track Ticks.

Rsvrd: 4 bits. Reserved bits **MUST** be set to 0.

INV: 1 bit. Invisible. The codec **SHOULD** decode this frame but not display it.

LACING: 2 bits. Uses lacing mode.

- 00b: no lacing ([Section 10.3.1](#))
- 01b: Xiph lacing ([Section 10.3.2](#))
- 11b: EBML lacing ([Section 10.3.3](#))
- 10b: fixed-size lacing ([Section 10.3.4](#))

UNU: 1 bit. Unused bit.

The following data in the Block corresponds to the lacing data and frames usage as described in each respective lacing mode.

10.2. SimpleBlock Structure

This section describes the binary data contained in the SimpleBlock Element ([Section 5.1.3.4](#)). Bit 0 is the most significant bit.

The SimpleBlock structure is inspired by the Block structure; see [Section 10.1](#). The main differences are the added Keyframe flag and Discardable flag. Otherwise, everything is the same.

As the TrackNumber size can vary between 1 and 8 octets, there are 8 different sizes for the SimpleBlock header. The definitions for TrackNumber sizes of 1 and 2 are provided; the other variants can be deduced by extending the size of the TrackNumber by multiples of 8 bits.

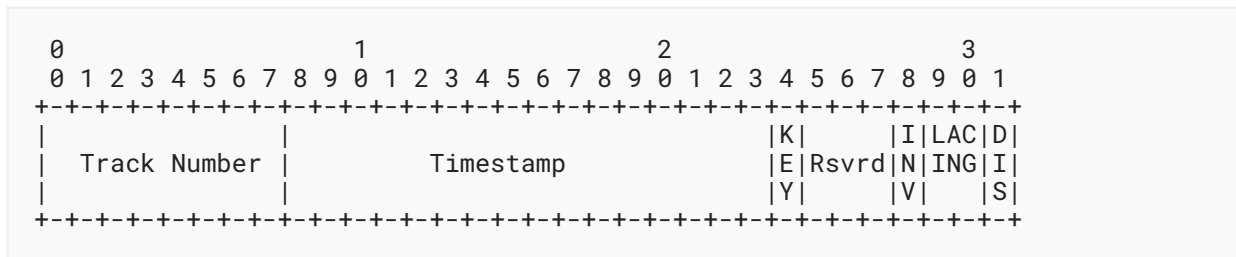


Figure 13: SimpleBlock Header with 1-Octet TrackNumber

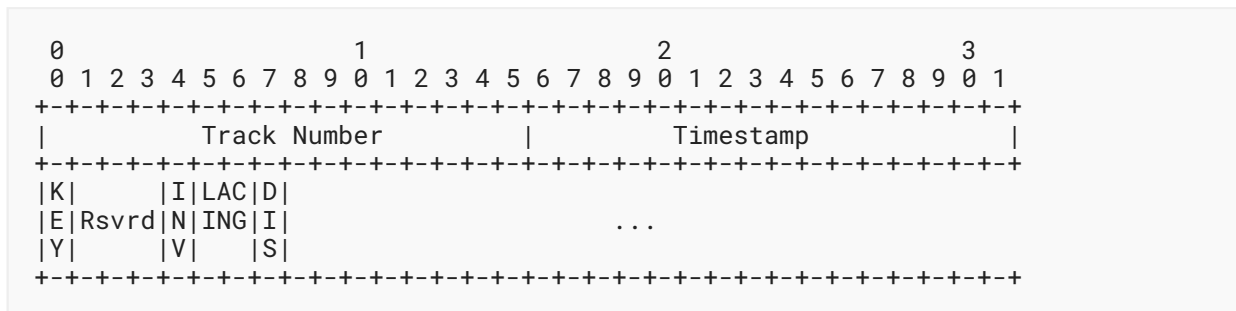


Figure 14: SimpleBlock Header with 2-Octet TrackNumber

where:

Track Number: 8, 16, 24, 32, 40, 48, or 64 bits. An EBML VINT-coded track number.

Timestamp: 16 bits. Signed timestamp in Track Ticks.

KEY: 1 bit. Keyframe. Set when the Block contains only keyframes.

Rsvrd: 3 bits. Reserved bits **MUST** be set to 0.

INV: 1 bit. Invisible. The codec **SHOULD** decode this frame but not display it.

LACING: 2 bits. Uses lacing mode.

- 00b: no lacing ([Section 10.3.1](#))
- 01b: Xiph lacing ([Section 10.3.2](#))
- 11b: EBML lacing ([Section 10.3.3](#))
- 10b: fixed-size lacing ([Section 10.3.4](#))

DIS: 1 bit. Discardable. The frames of the Block can be discarded during playing if needed.

The following data in the SimpleBlock corresponds to the lacing data and frames usage as described in each respective lacing mode.

10.3. 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 three types of lacing:

- Xiph, which is inspired by what is found in the Ogg container [RFC3533]
- EBML, which is the same with sizes coded differently
- Fixed-size, where the size is not coded

When lacing is not used, i.e., to store a single frame, lacing bits 5 and 6 of the Block or SimpleBlock **MUST** be set to 0.

For example, a user wants to store three 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. Because these frames are small, they can be stored in a lace to save space.

It is possible to not 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.

10.3.1. No Lacing

When no lacing is used, the number of frames in the lace is omitted, and only one frame can be stored in the Block. Bits 5 and 6 of the Block Header flags are set to 0b00.

The Block for an 800-octet frame is as follows:

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

Table 36: No Lacing

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

10.3.2. Xiph Lacing

The Xiph lacing uses the same coding of size as found in the Ogg container [RFC3533]. Bits 5 and 6 of the Block Header flags are set to 0b01.

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-octet frames are stored with Xiph lacing in a Block as follows:

Block Octets	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 37: Xiph Lacing Example

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

10.3.3. EBML Lacing

The EBML lacing encodes the frame size with an EBML-like encoding [RFC8794]. Bits 5 and 6 of the Block Header flags are set to 0b11.

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 VINT value. The remaining frame sizes are encoded as signed values using the difference between the frame size and the previous frame size. These signed values are encoded as VINT, with a mapping from signed to unsigned numbers. Decoding the unsigned number stored in the VINT to a signed number is done by subtracting $2^{((7*n)-1)-1}$, where n is the octet size of the VINT.

Bit Representation of Signed VINT	Possible Value Range
1xxx xxxx	2^7 values from $-(2^6-1)$ to 2^6
01xx xxxx xxxx xxxx	2^{14} values from $-(2^{13}-1)$ to 2^{13}
001x xxxx xxxx xxxx xxxx xxxx	2^{21} values from $-(2^{20}-1)$ to 2^{20}
0001 xxxx xxxx xxxx xxxx xxxx xxxx xxxx	2^{28} values from $-(2^{27}-1)$ to 2^{27}
0000 1xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	2^{35} values from $-(2^{34}-1)$ to 2^{34}

Table 38: EBML Lacing Signed VINT Bits Usage

In our example, the 800-, 500- and 1000-octet 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 39: EBML Lacing Example

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

10.3.4. Fixed-size Lacing

Fixed-size lacing doesn't store the frame size; rather, it only stores 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. Bits 5 and 6 of the Block Header flags are set to $0b10$.

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 three frames that are 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 40: Fixed-Size Lacing Example

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

10.3.5. 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. However, 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.

10.4. Random Access Points

Random Access Points (RAPs) are positions where the parser can seek to and start playback without decoding 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 `BlockGroups` or `SimpleBlocks` are RAPs. They are independent of each other and can be played randomly.

On the other hand, video tracks often use references to previous and future frames for better coding efficiency. Frames with such references **MUST** either contain one or more ReferenceBlock Elements in their BlockGroup or **MUST** be marked as non-keyframe in a SimpleBlock; see [Section 10.2](#).

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 RAPs (i.e., frames that 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 `DiscardPadding` element. For those 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, the `BlockGroup` **MUST** contain at least a `ReferenceBlock`. The `ReferenceBlocks` **MUST** be used in one of the following ways:

- each reference frame listed as a `ReferenceBlock`,
- some referenced frames listed as a `ReferenceBlock`, even if the timestamp value is accurate, or
- one `ReferenceBlock` with the timestamp 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 a 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. Thus, 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 information on references 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: Matroska Ticks, Segment Ticks, and Track Ticks.

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 5.1.4.1.13](#)
- `TrackEntry\DefaultDecodedFieldDuration`; defined in [Section 5.1.4.1.14](#)
- `TrackEntry\SeekPreRoll`; defined in [Section 5.1.4.1.26](#)
- `TrackEntry\CodecDelay`; defined in [Section 5.1.4.1.25](#)
- `BlockGroup\DiscardPadding`; defined in [Section 5.1.3.5.7](#)
- `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 for storage of smaller integer values in the elements.

When using the default value of "1,000,000" for `TimestampScale`, 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 for storage of smaller integer values in the elements. The resulting floating-point values of the timestamps are still expressed in nanoseconds.

When using the default values of "1,000,000" for `TimestampScale` and "1.0" for `TrackTimestampScale`, 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 5.1.3.5.3](#)
- `Cluster\BlockGroup\ReferenceBlock`; defined in [Section 5.1.3.5.5](#)

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 into account. Thus, using a value other than "1.0" might 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, the following formula is used:

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

The `Block Element` and `SimpleBlock Element` store their timestamps as 16-bit 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 subtracted 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:

$$\left(\left(\text{Cluster\Timestamp} + \left(\text{block timestamp} * \text{TrackTimestampScale} \right) \right) * \text{TimestampScale} \right) - \text{CodecDelay}$$

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

12. Language Codes

Matroska versions 1 through 3 use language codes that can be either the three-letter 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.

13. Country Codes

Country codes are the [BCP47] two-letter region subtags, without the UK exception.

14. Encryption

This Matroska specification provides no interoperable solution for securing the data container with any assurances of confidentiality, integrity, authenticity, or to provide authorization. The `ContentEncryption` Element (Section 5.1.4.1.31.8) and associated sub-fields (Section 5.1.4.1.31.9 to Section 5.1.4.1.31.12) are defined only for the benefit of implementers to construct their own proprietary solution or as the basis for further standardization activities. How to use these fields to secure a Matroska data container is out of scope, as are any related issues such as key management and distribution.

A Matroska Reader who encounters containers that use the fields defined in this section **MUST** rely on out-of-scope guidance to decode the associated content.

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 algorithms described in Table 26 support different modes of operations and key sizes. The specification of these parameters is required for a complete solution but is out of scope of this document and left to the proprietary implementations using them or subsequent profiles of this document.

The `ContentEncodingScope` Element gives an idea of which part of the track is encrypted, but each `ContentEncAlgo` Element and its sub-elements (like `AESSettingsCipherMode`) define exactly how the encrypted track should be interpreted.

An example of an extension that builds upon these security-related fields in this specification is [\[WebM-Enc\]](#). It uses AES-CTR, `ContentEncAlgo` = 5 (Section 5.1.4.1.31.9), and `AESSettingsCipherMode` = 1 (Section 5.1.4.1.31.12).

A Matroska Writer **MUST NOT** use insecure cryptographic algorithms to create new archives or streams, but a Matroska Reader **MAY** support these algorithms to read previously made archives or streams.

15. Image Presentation

15.1. Cropping

The `PixelCrop` Elements (`PixelCropTop`, `PixelCropBottom`, `PixelCropRight`, and `PixelCropLeft`) indicate when, and by how much, encoded video 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 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.

15.2. Rotation

The `ProjectionPoseRoll` Element ([Section 5.1.4.1.28.46](#)) can be used to indicate that the image from the associated video track **SHOULD** be rotated for presentation. For instance, the following example of the `Projection` Element ([Section 5.1.4.1.28.41](#)) 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 15: Rotation Example

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 that 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 the 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 quickly accessible based on their SegmentUUID in order to have a 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 SegmentUUIDs 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** consecutively follow 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 within the chain.

In a chain of Segments of a Linked Segment, the NextUUID always takes precedence over the PrevUUID. Thus, if SegmentA has a NextUUID to SegmentB and SegmentB has a PrevUUID to SegmentC, the link to use is NextUUID between SegmentA and SegmentB, and 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 shown in this table:

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

file name	SegmentUUID	PrevUUID	NextUUID
end.mkv	6c92285fa6d3e827 b198d120ea3ac674	a77b3598941cb803 eac0fcdafe44fac9	Invalid

Table 41: Usual Hard Linking UUIDs

An 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 42: 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 43: Hard Linking without NextUUID

An example where 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

file name	SegmentUUID	PrevUUID	NextUUID
end.mkv	6c92285fa6d3e827 b198d120ea3ac674	n/a	Invalid

Table 44: 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 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 an optional 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 two 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 the 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 ChapterSegmentUID.

When using Linked-Edition chapter linking, ChapterTimeEnd is **OPTIONAL**.

18. Track Flags

18.1. Default Flag

The Default 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 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" that is 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 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 audio track that is currently selected. This can be used to indicate that a track contains translations of on-screen text or dialogue spoken in a different language than the track's primary language.

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 user that is not hearing-impaired.

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 user that is not visually impaired.

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 user that is not visually impaired.

18.6. Original Flag

The `Original` flag tells the `Matroska Player` that this track is in the original language and that it **SHOULD** prefer the original language 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 for the combination of multiple tracks to make a virtual one. It uses two separate systems to combine tracks. One to create a 3D "composition" (left/right/background planes) and one to simply 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 union 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 it's 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 number of pixels (for example, 3840x1080 for full HD side by side), and the `DisplayWidth/DisplayHeight` in pixels is the number 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) that consists of packing two frames together in a `Block` that uses 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` does that. For more details, see [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 5.1.4.1.28.5](#)). `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), and 3 (both eyes); these are 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	

No.	Type	Lang	Layout	Original	Default	Other Flags	Name
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	
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 45: Audio Tracks for Default Selection

The table above shows a file with seven audio tracks -- five in English and two 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 tracks with the Original flag. 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 a track with the Original flag 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 and 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, the 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 more than one 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 46: Subtitle Tracks for Default Selection

The table above shows two audio tracks and five 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 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 on-screen 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 a marker in the timeline only. An Edition can be more complex with Ordered Chapters where a chapter end timestamp 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 its own built-in 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 is 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	FlagDefault	Default Edition
Edition 3	true	

Table 47: Default Edition, All Default

Edition	FlagDefault	Default Edition
Edition 1	false	X
Edition 2	false	
Edition 3	false	

Table 48: Default Edition, No Default

Edition	FlagDefault	Default Edition
Edition 1	false	
Edition 2	true	X
Edition 3	false	

Table 49: Default Edition, With Default

20.1.3. EditionFlagOrdered

The `EditionFlagOrdered` flag is a significant feature, as it enables an `Edition of Ordered Chapters` that 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 a 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`.

- `ChapterAtom/ChapterSegmentUUID`
- `ChapterAtom/ChapterSegmentEditionUID`

- ChapterAtom/ChapterTrack
- ChapterAtom/ChapProcess
- Info/ChapterTranslate
- TrackEntry/TrackTranslate

Furthermore, there are other EBML Elements that could be used if the `EditionFlagOrdered` evaluates to "1".

20.1.3.1. Ordered-Edition and Matroska Segment Linking

Hard Linking: Ordered Chapters supersede 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 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

`ChapterTimeStart` is the timestamp of the start of `Chapter` with nanosecond accuracy and is not scaled by `TimestampScale`. For `Simple Chapters`, this is the position of the chapter markers in the timeline.

20.2.2. ChapterTimeEnd

`ChapterTimeEnd` is the timestamp of the end of `Chapter` with nanosecond accuracy and is 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 timestamp 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	Start timestamp	End timestamp	Duration
Chapter 4	9000000000	8000000000	Invalid (-1000000000)

Table 50: 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 of 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 of Parent Chapters. A Nested Chapter with a ChapterFlagHidden flag 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 51: 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 two values are supported: 0 (Matroska Script) and 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 perform actions on the `Matroska Player`, such as 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_MEDIA_TYPE` tag [MatroskaTags] 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 52: *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) in which each track corresponds to a chapter.

- 00000 ms - 05000 ms: Intro
- 05000 ms - 25000 ms: Before the crime
- 25000 ms - 27500 ms: The crime
- 27500 ms - 38000 ms: The killer arrested
- 38000 ms - 43000 ms: 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>Apres 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>Generique</ChapString>
        <ChapLanguage>fra</ChapLanguage>
      </ChapterDisplay>
  </EditionEntry>
</Chapters>
```

```
</ChapterDisplay>
</ChapterAtom>
</EditionEntry>
</Chapters>
```

Figure 16: 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 contains 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>
    <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>Bleper_0+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>
  </EditionEntry>
</Chapters>
```

```

    </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 17: 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, pictures, 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 [RFC2083] 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 (e.g., 960x600 for landscape, 600x800 for portrait, or 600x600 for square). The dimension of the `small` cover **SHOULD** be 120 pixels on the smallest side (e.g., 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.

File Name	Image Orientation	Pixel Length of Smallest Side
<code>cover.jpg</code>	Portrait or square	600
<code>small_cover.png</code>	Portrait or square	120
<code>cover_land.png</code>	Landscape	600
<code>small_cover_land.jpg</code>	Landscape	120

Table 53: 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 that will be referred to as `Font Name` from now on. This `Font Name` can be different from 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) and Advanced SubStation Alpha (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** restrict its font rendering to use only these fonts.

A Matroska Player **SHOULD** handle the official font media types from [RFC8081] when the system can handle the type:

- `font/sfnt`: Generic SFNT Font Type
- `font/ttf`: TrueType Font (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 media types for fonts were used in existing files. Therefore, it is **RECOMMENDED** for a Matroska Player to support the following legacy media types for font attachments:

- `application/x-truetype-font`: TrueType fonts, equivalent to `font/ttf` and sometimes `font/otf`
- `application/x-font-ttf`: TrueType 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` media 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 media type from [RFC8081] in the `AttachedFile\FileMediaType` of the font attachment. They **MAY** use the media 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, each of which **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 that 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 that 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 once every 500 milliseconds at most.
- 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 a `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 it also includes accessing a file from an HTTP (web) server or Common Internet File System (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. When followed, the guidelines in [Section 25](#) 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 two families of servers for livestreaming: RTP / Real-Time Streaming Protocol (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 file-like protocols like HTTP, QUIC, etc., 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 nor a Cues list at the beginning of the stream, it **SHOULD** be considered non-seekable. Even though it is possible to seek 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. Tags

24.1. Tags Precedence

Tags allow tagging all kinds of Matroska parts with very detailed metadata in multiple languages.

Some Matroska elements also contain their own string value, like the Track Name ([Section 5.1.4.1.18](#)) or the Chapter String ([Section 5.1.7.1.4.10](#)).

The following Matroska elements can also be defined with tags:

- The Track Name Element ([Section 5.1.4.1.18](#)) corresponds to a tag with the TagTrackUID ([Section 5.1.8.1.1.3](#)) set to the given track, a TagName of TITLE ([Section 5.1.8.1.2.1](#)), and a TagLanguage ([Section 5.1.8.1.2.2](#)) or TagLanguageBCP47 ([Section 5.1.8.1.2.3](#)) of "und".
- The Chapter String Element ([Section 5.1.7.1.4.10](#)) corresponds to a tag with the TagChapterUID ([Section 5.1.8.1.1.5](#)) set to the same chapter UID, a TagName of TITLE ([Section 5.1.8.1.2.1](#)), and a TagLanguage ([Section 5.1.8.1.2.2](#)) or TagLanguageBCP47 ([Section 5.1.8.1.2.3](#)) matching the ChapLanguage ([Section 5.1.7.1.4.11](#)) or ChapLanguageBCP47 ([Section 5.1.7.1.4.12](#)), respectively.

- The FileDescription Element ([Section 5.1.6.1.1](#)) of an attachment corresponds to a tag with the TagAttachmentUID ([Section 5.1.8.1.1.6](#)) set to the given attachment, a TagName of TITLE ([Section 5.1.8.1.2.1](#)), and a TagLanguage ([Section 5.1.8.1.2.2](#)) or TagLanguageBCP47 ([Section 5.1.8.1.2.3](#)) of "und".

When both values exist in the file, the value found in Tags takes precedence over the value found in the original location of the element. For example, if you have a TrackEntry\Name element and Tag TITLE for that track in a Matroska Segment, the Tag string **SHOULD** be used instead of the TrackEntry\Name string to identify the track.

As the Tag element is optional, a lot of Matroska Readers do not handle it and will not use the tags value when it's found. Thus, for maximum compatibility, it's usually better to put the strings in the TrackEntry, ChapterAtom, and Attachment and keep the tags matching these values if tags are also used.

24.2. Tag Levels

Tag elements allow tagging information on multiple levels, with each level having a TargetTypeValue [Section 5.1.8.1.1.1](#). An element for a given TargetTypeValue also applies to the lower levels denoted by smaller TargetTypeValue values. If an upper value doesn't apply to a level but the actual value to use is not known, an empty TagString ([Section 5.1.8.1.2.5](#)) or an empty TagBinary ([Section 5.1.8.1.2.6](#)) **MUST** be used as the tag value for this level.

See [\[MatroskaTags\]](#) for more details on common tag names, types, and descriptions.

25. Implementation Recommendations

25.1. Cluster

It is **RECOMMENDED** that each individual Cluster Element contain no more than five seconds or five megabytes of content.

25.2. 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 on the Matroska file already having Tags, Chapters, and Attachments Elements.

25.3. Optimum Layouts

While there can be Top-Level Elements in any order, some orderings of Elements are better than others. The following subsections detail optimum layouts for different use cases.

25.3.1. 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

25.3.2. 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

25.3.3. Optimum Layout with Cues at the Front

Cues are usually a big chunk of data referencing a lot of locations in the file. Players that want to seek in the file need to seek to the end of the file to access 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, this layout is rarely used:

- SeekHead
- Info
- Tracks
- Chapters
- Attachments

- Tags
- Cues
- Clusters

25.3.4. Optimum Layout for Livestreaming

In livestreaming ([Section 23.2](#)), only a few elements make sense. For example, SeekHead and Cues are useless. All elements other than the Clusters **MUST** be placed before the Clusters.

- Info
- Tracks
- Attachments (rare)
- Tags
- Clusters

26. Security Considerations

Matroska inherits security considerations from EBML.

Attacks on a Matroska Reader could include:

- Storage of an 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 or not use the attachment.
- A Matroska Attachment with an inaccurate media type.
- Damage to the Encryption and Compression fields ([Section 14](#)) that would result in bogus binary data interpreted by the decoder.
- Chapter Codecs running unwanted commands on the host system.

The same error handling done for EBML applies to Matroska files. Particular error handling is not covered in this specification, as this depends on the goal of the Matroska Readers. It is up to the decision of the Matroska Readers on how to handle the errors if they are recoverable in their code or not. For example, if the checksum of the \Segment\Tracks is invalid, some could decide to try to read the data anyway, some will just reject the file, and most will not even check it.

Matroska Reader implementations need to be robust against malicious payloads. Those related to denial of service are outlined in [Section 2.1](#) of [[RFC4732](#)].

Although rarer, the same may apply to a Matroska Writer. Malicious stream data must not cause the Matroska Writer to misbehave, as this might allow an attacker access to transcoding gateways.

As an audio and visual container format, a Matroska file or stream will potentially encapsulate numerous byte streams created with a variety of codecs. Implementers will need to consider the security considerations of these encapsulated formats.

27. IANA Considerations

27.1. Matroska Element IDs Registry

IANA has created a new registry called the "Matroska Element IDs" registry.

The following are needed to register a new Element ID in this registry: an Element ID, a Change Controller (IETF or email of registrant), and an optional Reference to a document describing the Element ID.

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 Element IDs are possible only if declared in the EBML Header.

Element IDs are described in [Section 5](#) of [\[RFC8794\]](#), with the changes in [\[Err7189\]](#) and [\[Err7191\]](#).

One-octet Matroska Element IDs are to be allocated according to the "RFC Required" policy [\[RFC8126\]](#).

Two-octet Matroska Element IDs are to be allocated according to the "Specification Required" policy [\[RFC8126\]](#).

Three-octet and four-octet Matroska Element IDs are to be allocated according to the "First Come First Served" policy [\[RFC8126\]](#).

The allowed values in the "Matroska Element IDs" registry are similar to the ones found in the "EBML Element IDs" registry defined in [Section 17.1](#) of [\[RFC8794\]](#).

EBML Element IDs defined for the EBML Header -- as defined in [Section 17.1](#) of [\[RFC8794\]](#) -- **MUST NOT** be used as Matroska Element IDs.

Given the scarcity of one-octet Element IDs, they should only be created to save space for elements found many times in a file (for example, within a BlockGroup or Chapters). The four-octet Element IDs are mostly for synchronization of large elements. They should only be used for such high-level elements. Elements that are not expected to be used often should use three-octet Element IDs.

Elements found in [Appendix A](#) have an assigned Matroska Element ID for historical reasons. These elements are not in use and **SHOULD NOT** be reused unless there are no other IDs available with the desired size. Such IDs are marked as "Reclaimed" in the "Matroska Element IDs" registry, as they could be used for other things in the future.

[Table 54](#) shows the initial contents of the "Matroska Element IDs" registry.

Element ID	Element Name	Reference
0x80	ChapterDisplay	RFC 9559, Section 5.1.7.1.4.9

Element ID	Element Name	Reference
0x83	TrackType	RFC 9559, Section 5.1.4.1.3
0x85	ChapString	RFC 9559, Section 5.1.7.1.4.10
0x86	CodecID	RFC 9559, Section 5.1.4.1.21
0x88	FlagDefault	RFC 9559, Section 5.1.4.1.5
0x8E	Slices	Reclaimed (RFC 9559, Appendix A.5)
0x91	ChapterTimeStart	RFC 9559, Section 5.1.7.1.4.3
0x92	ChapterTimeEnd	RFC 9559, Section 5.1.7.1.4.4
0x96	CueRefTime	RFC 9559, Section 5.1.5.1.2.8
0x97	CueRefCluster	Reclaimed (RFC 9559, Appendix A.37)
0x98	ChapterFlagHidden	RFC 9559, Section 5.1.7.1.4.5
0x9A	FlagInterlaced	RFC 9559, Section 5.1.4.1.28.1
0x9B	BlockDuration	RFC 9559, Section 5.1.3.5.3
0x9C	FlagLacing	RFC 9559, Section 5.1.4.1.12
0x9D	FieldOrder	RFC 9559, Section 5.1.4.1.28.2
0x9F	Channels	RFC 9559, Section 5.1.4.1.29.3
0xA0	BlockGroup	RFC 9559, Section 5.1.3.5
0xA1	Block	RFC 9559, Section 5.1.3.5.1
0xA2	BlockVirtual	Reclaimed (RFC 9559, Appendix A.3)
0xA3	SimpleBlock	RFC 9559, Section 5.1.3.4
0xA4	CodecState	RFC 9559, Section 5.1.3.5.6
0xA5	BlockAdditional	RFC 9559, Section 5.1.3.5.2.2
0xA6	BlockMore	RFC 9559, Section 5.1.3.5.2.1
0xA7	Position	RFC 9559, Section 5.1.3.2
0xAA	CodecDecodeAll	Reclaimed (RFC 9559, Appendix A.22)

Element ID	Element Name	Reference
0xAB	PrevSize	RFC 9559, Section 5.1.3.3
0xAE	TrackEntry	RFC 9559, Section 5.1.4.1
0xAF	EncryptedBlock	Reclaimed (RFC 9559, Appendix A.15)
0xB0	PixelWidth	RFC 9559, Section 5.1.4.1.28.6
0xB2	CueDuration	RFC 9559, Section 5.1.5.1.2.4
0xB3	CueTime	RFC 9559, Section 5.1.5.1.1
0xB5	SamplingFrequency	RFC 9559, Section 5.1.4.1.29.1
0xB6	ChapterAtom	RFC 9559, Section 5.1.7.1.4
0xB7	CueTrackPositions	RFC 9559, Section 5.1.5.1.2
0xB9	FlagEnabled	RFC 9559, Section 5.1.4.1.4
0xBA	PixelHeight	RFC 9559, Section 5.1.4.1.28.7
0xBB	CuePoint	RFC 9559, Section 5.1.5.1
0xC0	TrickTrackUID	Reclaimed (RFC 9559, Appendix A.28)
0xC1	TrickTrackSegmentUID	Reclaimed (RFC 9559, Appendix A.29)
0xC4	TrickMasterTrackSegmentUID	Reclaimed (RFC 9559, Appendix A.32)
0xC6	TrickTrackFlag	Reclaimed (RFC 9559, Appendix A.30)
0xC7	TrickMasterTrackUID	Reclaimed (RFC 9559, Appendix A.31)
0xC8	ReferenceFrame	Reclaimed (RFC 9559, Appendix A.12)
0xC9	ReferenceOffset	Reclaimed (RFC 9559, Appendix A.13)
0xCA	ReferenceTimestamp	Reclaimed (RFC 9559, Appendix A.14)
0xCB	BlockAdditionID	Reclaimed (RFC 9559, Appendix A.9)
0xCC	LaceNumber	Reclaimed (RFC 9559, Appendix A.7)
0xCD	FrameNumber	Reclaimed (RFC 9559, Appendix A.8)
0xCE	Delay	Reclaimed (RFC 9559, Appendix A.10)

Element ID	Element Name	Reference
0xCF	SliceDuration	Reclaimed (RFC 9559, Appendix A.11)
0xD7	TrackNumber	RFC 9559, Section 5.1.4.1.1
0xDB	CueReference	RFC 9559, Section 5.1.5.1.2.7
0xE0	Video	RFC 9559, Section 5.1.4.1.28
0xE1	Audio	RFC 9559, Section 5.1.4.1.29
0xE2	TrackOperation	RFC 9559, Section 5.1.4.1.30
0xE3	TrackCombinePlanes	RFC 9559, Section 5.1.4.1.30.1
0xE4	TrackPlane	RFC 9559, Section 5.1.4.1.30.2
0xE5	TrackPlaneUID	RFC 9559, Section 5.1.4.1.30.3
0xE6	TrackPlaneType	RFC 9559, Section 5.1.4.1.30.4
0xE7	Timestamp	RFC 9559, Section 5.1.3.1
0xE8	TimeSlice	Reclaimed (RFC 9559, Appendix A.6)
0xE9	TrackJoinBlocks	RFC 9559, Section 5.1.4.1.30.5
0xEA	CueCodecState	RFC 9559, Section 5.1.5.1.2.6
0xEB	CueRefCodecState	Reclaimed (RFC 9559, Appendix A.39)
0xED	TrackJoinUID	RFC 9559, Section 5.1.4.1.30.6
0xEE	BlockAddID	RFC 9559, Section 5.1.3.5.2.3
0xF0	CueRelativePosition	RFC 9559, Section 5.1.5.1.2.3
0xF1	CueClusterPosition	RFC 9559, Section 5.1.5.1.2.2
0xF7	CueTrack	RFC 9559, Section 5.1.5.1.2.1
0xFA	ReferencePriority	RFC 9559, Section 5.1.3.5.4
0xFB	ReferenceBlock	RFC 9559, Section 5.1.3.5.5
0xFD	ReferenceVirtual	Reclaimed (RFC 9559, Appendix A.4)
0x41A4	BlockAddIDName	RFC 9559, Section 5.1.4.1.17.2

Element ID	Element Name	Reference
0x41E4	BlockAdditionMapping	RFC 9559, Section 5.1.4.1.17
0x41E7	BlockAddIDType	RFC 9559, Section 5.1.4.1.17.3
0x41ED	BlockAddIDExtraData	RFC 9559, Section 5.1.4.1.17.4
0x41F0	BlockAddIDValue	RFC 9559, Section 5.1.4.1.17.1
0x4254	ContentCompAlgo	RFC 9559, Section 5.1.4.1.31.6
0x4255	ContentCompSettings	RFC 9559, Section 5.1.4.1.31.7
0x437C	ChapLanguage	RFC 9559, Section 5.1.7.1.4.11
0x437D	ChapLanguageBCP47	RFC 9559, Section 5.1.7.1.4.12
0x437E	ChapCountry	RFC 9559, Section 5.1.7.1.4.13
0x4444	SegmentFamily	RFC 9559, Section 5.1.2.7
0x4461	DateUTC	RFC 9559, Section 5.1.2.11
0x447A	TagLanguage	RFC 9559, Section 5.1.8.1.2.2
0x447B	TagLanguageBCP47	RFC 9559, Section 5.1.8.1.2.3
0x4484	TagDefault	RFC 9559, Section 5.1.8.1.2.4
0x4485	TagBinary	RFC 9559, Section 5.1.8.1.2.6
0x4487	TagString	RFC 9559, Section 5.1.8.1.2.5
0x4489	Duration	RFC 9559, Section 5.1.2.10
0x44B4	TagDefaultBogus	Reclaimed (RFC 9559, Appendix A.43)
0x450D	ChapProcessPrivate	RFC 9559, Section 5.1.7.1.4.16
0x45A3	TagName	RFC 9559, Section 5.1.8.1.2.1
0x45B9	EditionEntry	RFC 9559, Section 5.1.7.1
0x45BC	EditionUID	RFC 9559, Section 5.1.7.1.1
0x45DB	EditionFlagDefault	RFC 9559, Section 5.1.7.1.2
0x45DD	EditionFlagOrdered	RFC 9559, Section 5.1.7.1.3

Element ID	Element Name	Reference
0x465C	FileData	RFC 9559, Section 5.1.6.1.4
0x4660	FileMediaType	RFC 9559, Section 5.1.6.1.3
0x4661	FileUsedStartTime	Reclaimed (RFC 9559, Appendix A.41)
0x4662	FileUsedEndTime	Reclaimed (RFC 9559, Appendix A.42)
0x466E	FileName	RFC 9559, Section 5.1.6.1.2
0x4675	FileReferral	Reclaimed (RFC 9559, Appendix A.40)
0x467E	FileDescription	RFC 9559, Section 5.1.6.1.1
0x46AE	FileUID	RFC 9559, Section 5.1.6.1.5
0x47E1	ContentEncAlgo	RFC 9559, Section 5.1.4.1.31.9
0x47E2	ContentEncKeyID	RFC 9559, Section 5.1.4.1.31.10
0x47E3	ContentSignature	Reclaimed (RFC 9559, Appendix A.33)
0x47E4	ContentSigKeyID	Reclaimed (RFC 9559, Appendix A.34)
0x47E5	ContentSigAlgo	Reclaimed (RFC 9559, Appendix A.35)
0x47E6	ContentSigHashAlgo	Reclaimed (RFC 9559, Appendix A.36)
0x47E7	ContentEncAESSettings	RFC 9559, Section 5.1.4.1.31.11
0x47E8	AESSettingsCipherMode	RFC 9559, Section 5.1.4.1.31.12
0x4D80	MuxingApp	RFC 9559, Section 5.1.2.13
0x4DBB	Seek	RFC 9559, Section 5.1.1.1
0x5031	ContentEncodingOrder	RFC 9559, Section 5.1.4.1.31.2
0x5032	ContentEncodingScope	RFC 9559, Section 5.1.4.1.31.3
0x5033	ContentEncodingType	RFC 9559, Section 5.1.4.1.31.4
0x5034	ContentCompression	RFC 9559, Section 5.1.4.1.31.5
0x5035	ContentEncryption	RFC 9559, Section 5.1.4.1.31.8
0x535F	CueRefNumber	Reclaimed (RFC 9559, Appendix A.38)

Element ID	Element Name	Reference
0x536E	Name	RFC 9559, Section 5.1.4.1.18
0x5378	CueBlockNumber	RFC 9559, Section 5.1.5.1.2.5
0x537F	TrackOffset	Reclaimed (RFC 9559, Appendix A.18)
0x53AB	SeekID	RFC 9559, Section 5.1.1.1.1
0x53AC	SeekPosition	RFC 9559, Section 5.1.1.1.2
0x53B8	StereoMode	RFC 9559, Section 5.1.4.1.28.3
0x53B9	OldStereoMode	RFC 9559, Section 5.1.4.1.28.5
0x53C0	AlphaMode	RFC 9559, Section 5.1.4.1.28.4
0x54AA	PixelCropBottom	RFC 9559, Section 5.1.4.1.28.8
0x54B0	DisplayWidth	RFC 9559, Section 5.1.4.1.28.12
0x54B2	DisplayUnit	RFC 9559, Section 5.1.4.1.28.14
0x54B3	AspectRatioType	Reclaimed (RFC 9559, Appendix A.24)
0x54BA	DisplayHeight	RFC 9559, Section 5.1.4.1.28.13
0x54BB	PixelCropTop	RFC 9559, Section 5.1.4.1.28.9
0x54CC	PixelCropLeft	RFC 9559, Section 5.1.4.1.28.10
0x54DD	PixelCropRight	RFC 9559, Section 5.1.4.1.28.11
0x55AA	FlagForced	RFC 9559, Section 5.1.4.1.6
0x55AB	FlagHearingImpaired	RFC 9559, Section 5.1.4.1.7
0x55AC	FlagVisualImpaired	RFC 9559, Section 5.1.4.1.8
0x55AD	FlagTextDescriptions	RFC 9559, Section 5.1.4.1.9
0x55AE	FlagOriginal	RFC 9559, Section 5.1.4.1.10
0x55AF	FlagCommentary	RFC 9559, Section 5.1.4.1.11
0x55B0	Colour	RFC 9559, Section 5.1.4.1.28.16
0x55B1	MatrixCoefficients	RFC 9559, Section 5.1.4.1.28.17

Element ID	Element Name	Reference
0x55B2	BitsPerChannel	RFC 9559, Section 5.1.4.1.28.18
0x55B3	ChromaSubsamplingHorz	RFC 9559, Section 5.1.4.1.28.19
0x55B4	ChromaSubsamplingVert	RFC 9559, Section 5.1.4.1.28.20
0x55B5	CbSubsamplingHorz	RFC 9559, Section 5.1.4.1.28.21
0x55B6	CbSubsamplingVert	RFC 9559, Section 5.1.4.1.28.22
0x55B7	ChromaSitingHorz	RFC 9559, Section 5.1.4.1.28.23
0x55B8	ChromaSitingVert	RFC 9559, Section 5.1.4.1.28.24
0x55B9	Range	RFC 9559, Section 5.1.4.1.28.25
0x55BA	TransferCharacteristics	RFC 9559, Section 5.1.4.1.28.26
0x55BB	Primaries	RFC 9559, Section 5.1.4.1.28.27
0x55BC	MaxCLL	RFC 9559, Section 5.1.4.1.28.28
0x55BD	MaxFALL	RFC 9559, Section 5.1.4.1.28.29
0x55D0	MasteringMetadata	RFC 9559, Section 5.1.4.1.28.30
0x55D1	PrimaryRChromaticityX	RFC 9559, Section 5.1.4.1.28.31
0x55D2	PrimaryRChromaticityY	RFC 9559, Section 5.1.4.1.28.32
0x55D3	PrimaryGChromaticityX	RFC 9559, Section 5.1.4.1.28.33
0x55D4	PrimaryGChromaticityY	RFC 9559, Section 5.1.4.1.28.34
0x55D5	PrimaryBChromaticityX	RFC 9559, Section 5.1.4.1.28.35
0x55D6	PrimaryBChromaticityY	RFC 9559, Section 5.1.4.1.28.36
0x55D7	WhitePointChromaticityX	RFC 9559, Section 5.1.4.1.28.37
0x55D8	WhitePointChromaticityY	RFC 9559, Section 5.1.4.1.28.38
0x55D9	LuminanceMax	RFC 9559, Section 5.1.4.1.28.39
0x55DA	LuminanceMin	RFC 9559, Section 5.1.4.1.28.40
0x55EE	MaxBlockAdditionID	RFC 9559, Section 5.1.4.1.16

Element ID	Element Name	Reference
0x5654	ChapterStringUID	RFC 9559, Section 5.1.7.1.4.2
0x56AA	CodecDelay	RFC 9559, Section 5.1.4.1.25
0x56BB	SeekPreRoll	RFC 9559, Section 5.1.4.1.26
0x5741	WritingApp	RFC 9559, Section 5.1.2.14
0x5854	SilentTracks	Reclaimed (RFC 9559, Appendix A.1)
0x58D7	SilentTrackNumber	Reclaimed (RFC 9559, Appendix A.2)
0x61A7	AttachedFile	RFC 9559, Section 5.1.6.1
0x6240	ContentEncoding	RFC 9559, Section 5.1.4.1.31.1
0x6264	BitDepth	RFC 9559, Section 5.1.4.1.29.4
0x63A2	CodecPrivate	RFC 9559, Section 5.1.4.1.22
0x63C0	Targets	RFC 9559, Section 5.1.8.1.1
0x63C3	ChapterPhysicalEquiv	RFC 9559, Section 5.1.7.1.4.8
0x63C4	TagChapterUID	RFC 9559, Section 5.1.8.1.1.5
0x63C5	TagTrackUID	RFC 9559, Section 5.1.8.1.1.3
0x63C6	TagAttachmentUID	RFC 9559, Section 5.1.8.1.1.6
0x63C9	TagEditionUID	RFC 9559, Section 5.1.8.1.1.4
0x63CA	TargetType	RFC 9559, Section 5.1.8.1.1.2
0x6624	TrackTranslate	RFC 9559, Section 5.1.4.1.27
0x66A5	TrackTranslateTrackID	RFC 9559, Section 5.1.4.1.27.1
0x66BF	TrackTranslateCodec	RFC 9559, Section 5.1.4.1.27.2
0x66FC	TrackTranslateEditionUID	RFC 9559, Section 5.1.4.1.27.3
0x67C8	SimpleTag	RFC 9559, Section 5.1.8.1.2
0x68CA	TargetTypeValue	RFC 9559, Section 5.1.8.1.1.1
0x6911	ChapProcessCommand	RFC 9559, Section 5.1.7.1.4.17

Element ID	Element Name	Reference
0x6922	ChapProcessTime	RFC 9559, Section 5.1.7.1.4.18
0x6924	ChapterTranslate	RFC 9559, Section 5.1.2.8
0x6933	ChapProcessData	RFC 9559, Section 5.1.7.1.4.19
0x6944	ChapProcess	RFC 9559, Section 5.1.7.1.4.14
0x6955	ChapProcessCodecID	RFC 9559, Section 5.1.7.1.4.15
0x69A5	ChapterTranslateID	RFC 9559, Section 5.1.2.8.1
0x69BF	ChapterTranslateCodec	RFC 9559, Section 5.1.2.8.2
0x69FC	ChapterTranslateEditionUID	RFC 9559, Section 5.1.2.8.3
0x6D80	ContentEncodings	RFC 9559, Section 5.1.4.1.31
0x6DE7	MinCache	Reclaimed (RFC 9559, Appendix A.16)
0x6DF8	MaxCache	Reclaimed (RFC 9559, Appendix A.17)
0x6E67	ChapterSegmentUUID	RFC 9559, Section 5.1.7.1.4.6
0x6EBC	ChapterSegmentEditionUID	RFC 9559, Section 5.1.7.1.4.7
0x6FAB	TrackOverlay	Reclaimed (RFC 9559, Appendix A.23)
0x7373	Tag	RFC 9559, Section 5.1.8.1
0x7384	SegmentFilename	RFC 9559, Section 5.1.2.2
0x73A4	SegmentUUID	RFC 9559, Section 5.1.2.1
0x73C4	ChapterUID	RFC 9559, Section 5.1.7.1.4.1
0x73C5	TrackUID	RFC 9559, Section 5.1.4.1.2
0x7446	AttachmentLink	RFC 9559, Section 5.1.4.1.24
0x75A1	BlockAdditions	RFC 9559, Section 5.1.3.5.2
0x75A2	DiscardPadding	RFC 9559, Section 5.1.3.5.7
0x7670	Projection	RFC 9559, Section 5.1.4.1.28.41
0x7671	ProjectionType	RFC 9559, Section 5.1.4.1.28.42

Element ID	Element Name	Reference
0x7672	ProjectionPrivate	RFC 9559, Section 5.1.4.1.28.43
0x7673	ProjectionPoseYaw	RFC 9559, Section 5.1.4.1.28.44
0x7674	ProjectionPosePitch	RFC 9559, Section 5.1.4.1.28.45
0x7675	ProjectionPoseRoll	RFC 9559, Section 5.1.4.1.28.46
0x78B5	OutputSamplingFrequency	RFC 9559, Section 5.1.4.1.29.2
0x7BA9	Title	RFC 9559, Section 5.1.2.12
0x7D7B	ChannelPositions	Reclaimed (RFC 9559, Appendix A.27)
0x22B59C	Language	RFC 9559, Section 5.1.4.1.19
0x22B59D	LanguageBCP47	RFC 9559, Section 5.1.4.1.20
0x23314F	TrackTimestampScale	RFC 9559, Section 5.1.4.1.15
0x234E7A	DefaultDecodedFieldDuration	RFC 9559, Section 5.1.4.1.14
0x2383E3	FrameRate	Reclaimed (RFC 9559, Appendix A.26)
0x23E383	DefaultDuration	RFC 9559, Section 5.1.4.1.13
0x258688	CodecName	RFC 9559, Section 5.1.4.1.23
0x26B240	CodecDownloadURL	Reclaimed (RFC 9559, Appendix A.21)
0x2AD7B1	TimestampScale	RFC 9559, Section 5.1.2.9
0x2EB524	UncompressedFourCC	RFC 9559, Section 5.1.4.1.28.15
0x2FB523	GammaValue	Reclaimed (RFC 9559, Appendix A.25)
0x3A9697	CodecSettings	Reclaimed (RFC 9559, Appendix A.19)
0x3B4040	CodecInfoURL	Reclaimed (RFC 9559, Appendix A.20)
0x3C83AB	PrevFilename	RFC 9559, Section 5.1.2.4
0x3CB923	PrevUUID	RFC 9559, Section 5.1.2.3
0x3E83BB	NextFilename	RFC 9559, Section 5.1.2.6
0x3EB923	NextUUID	RFC 9559, Section 5.1.2.5

Element ID	Element Name	Reference
0x1043A770	Chapters	RFC 9559, Section 5.1.7
0x114D9B74	SeekHead	RFC 9559, Section 5.1.1
0x1254C367	Tags	RFC 9559, Section 5.1.8
0x1549A966	Info	RFC 9559, Section 5.1.2
0x1654AE6B	Tracks	RFC 9559, Section 5.1.4
0x18538067	Segment	RFC 9559, Section 5.1
0x1941A469	Attachments	RFC 9559, Section 5.1.6
0x1C53BB6B	Cues	RFC 9559, Section 5.1.5
0x1F43B675	Cluster	RFC 9559, Section 5.1.3

Table 54: Initial Contents of "Matroska Element IDs" Registry

27.2. Chapter Codec IDs Registry

IANA has created a new registry called the "Matroska Chapter Codec IDs" registry. The values correspond to the unsigned integer ChapProcessCodecID value described in [Section 5.1.7.1.4.15](#).

To register a new Chapter Codec ID in this registry, one needs a Chapter Codec ID, a Change Controller (IETF or email of registrant), and an optional Reference to a document describing the Chapter Codec ID.

The Chapter Codec IDs are to be allocated according to the "First Come First Served" policy [[RFC8126](#)].

Values of "0" and "1" are reserved for future use (with the IETF as the Change Controller).

27.3. Media Types

Matroska files and streams are found in three main forms: audio-video files, audio-only, and occasionally with stereoscopic video tracks.

Historically, Matroska files and streams have used the following media types with an "x-" prefix. 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 format in [[RFC6838](#)] instead.

IANA has registered three media types per the templates (see [[RFC6838](#)]) in the following subsections.

27.3.1. For Files Containing Video Tracks

Type name: video

Subtype name: matroska

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: As per RFCs 9559 and 8794

Security considerations: See [Section 26](#) of RFC 9559.

Interoperability considerations: Due to the extensibility of Matroska, it is possible to encounter files with unknown but valid EBML Elements. Readers should be ready to handle this case. The fixed byte order, octet boundaries, and UTF-8 usage allow for broad interoperability.

Published specification: RFC 9559

Applications that use this media type: FFmpeg, VLC, ...

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: video/x-matroska

Magic number(s): N/A

File extension(s): mkv

Macintosh file type code(s): N/A

Person & email address to contact for further information: IETF CELLAR WG (cellar@ietf.org)

Intended usage: COMMON

Restrictions on usage: None

Author: IETF CELLAR WG

Change controller: IETF

27.3.2. For Files Containing Audio Tracks with No Video Tracks

Type name: audio

Subtype name: matroska

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: As per RFCs 9559 and 8794

Security considerations: See [Section 26](#) of RFC 9559.

Interoperability considerations: Due to the extensibility of Matroska, it is possible to encounter files with unknown but valid EBML Elements. Readers should be ready to handle this case. The fixed byte order, octet boundaries, and UTF-8 usage allow for broad interoperability.

Published specification: RFC 9559

Applications that use this media type: FFmpeg, VLC, ...

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: audio/x-matroska

Magic number(s): N/A

File extension(s): mka

Macintosh file type code(s): N/A

Person & email address to contact for further information: IETF CELLAR WG (cellar@ietf.org)

Intended usage: COMMON

Restrictions on usage: None

Author: IETF CELLAR WG

Change controller: IETF

27.3.3. For Files Containing a Stereoscopic Video Track

Type name: video

Subtype name: matroska-3d

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: As per RFCs 9559 and 8794

Security considerations: See [Section 26](#) of RFC 9559.

Interoperability considerations: Due to the extensibility of Matroska, it is possible to encounter files with unknown but valid EBML Elements. Readers should be ready to handle this case. The fixed byte order, octet boundaries, and UTF-8 usage allow for broad interoperability.

Published specification: RFC 9559

Applications that use this media type: FFmpeg, VLC, ...

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: video/x-matroska-3d

Magic number(s): N/A

File extension(s): mk3d

Macintosh file type code(s): N/A

Person & email address to contact for further information: IETF CELLAR WG (cellar@ietf.org)

Intended usage: COMMON

Restrictions on usage: None

Author: IETF CELLAR WG

Change controller: IETF

28. References

28.1. Normative References

- [BCP47] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <<https://www.rfc-editor.org/info/rfc5646>>.
- [CIE-1931] Wikipedia, "CIE 1931 color space", <https://en.wikipedia.org/wiki/CIE_1931_color_space>.
- [ISO639-2] International Organization for Standardization, "Codes for the Representation of Names of Languages", ISO 639-2, December 2017, <https://www.loc.gov/standards/iso639-2/php/code_list.php>.
- [ISO9899] International Organization for Standardization, "Information technology -- Programming languages -- C", ISO/IEC 9899:2018, June 2018, <<https://www.iso.org/standard/74528.html>>.
- [ITU-H.273] ITU-T, "Coding-independent code points for video signal type identification", ITU-T Recommendation H.273, September 2023, <<https://www.itu.int/rec/T-REC-H.273-202309-P/en>>.
- [RFC1950] Deutsch, P. and J. 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>>.

28.2. Informative References

- [AVIFormat] Microsoft Corporation, "AVI RIFF File Reference", June 2023, <<https://docs.microsoft.com/en-us/windows/win32/directshow/avi-riff-file-reference>>.
- [Blowfish] Schneier, B., "The Blowfish Encryption Algorithm", 1993, <<https://www.schneier.com/academic/blowfish/>>.
- [BZIP2] Seward, J., "bzip2", July 2019, <<https://sourceware.org/bzip2/>>.
- [DivXTrickTrack] "DivX Trick Track Extensions", December 2010, <<https://web.archive.org/web/20101222001148/http://labs.divx.com/node/16601>>.
- [DivXWorldFonts] "World Fonts", December 2010, <<https://web.archive.org/web/20110214132246/http://labs.divx.com/node/16602>>.
- [DVD-Video] DVD Forum, "DVD-Books: Part 3 DVD-Video Book", November 1995, <<http://www.dvdforum.org/>>.
- [Err7189] RFC Errata, Erratum ID 7189, RFC 8794, <<https://www.rfc-editor.org/errata/eid7189>>.
- [Err7191] RFC Errata, Erratum ID 7191, RFC 8794, <<https://www.rfc-editor.org/errata/eid7191>>.
- [FIPS197] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)", FIPS PUB 197, DOI 10.6028/NIST.FIPS.197, November 2001, <<https://csrc.nist.gov/publications/detail/fips/197/final>>.

-
- [FIPS46-3]** National Institute of Standards and Technology (NIST), "Data Encryption Standard (DES)", FIPS PUB 46, October 1999, <<https://csrc.nist.gov/publications/detail/fips/46/3/archive/1999-10-25>>.
- [FourCC-RGB]** FOURCC, "RGB pixel formats", <<https://web.archive.org/web/20160609214806/https://www.fourcc.org/rgb.php>>.
- [FourCC-YUV]** FOURCC, "YUV pixel formats", <<https://web.archive.org/web/20160609214806/https://www.fourcc.org/yuv.php>>.
- [JPEG]** ITU, "INFORMATION TECHNOLOGY - DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES - REQUIREMENTS AND GUIDELINES", ITU Recommendation T.81, September 1992, <<https://www.w3.org/Graphics/JPEG/itu-t81.pdf>>.
- [LZO]** Tarreau, W. and R. Rodgman, "LZO stream format as understood by Linux's LZO decompressor", October 2018, <<https://www.kernel.org/doc/Documentation/lzo.txt>>.
- [MatroskaCodec]** Lhomme, S., Bunkus, M., and D. Rice, "Matroska Media Container Codec Specifications", Work in Progress, Internet-Draft, draft-ietf-cellar-codec-12, 27 January 2024, <<https://datatracker.ietf.org/doc/html/draft-ietf-cellar-codec-12>>.
- [MatroskaTags]** Lhomme, S., Bunkus, M., and D. Rice, "Matroska Media Container Tag Specifications", Work in Progress, Internet-Draft, draft-ietf-cellar-tags-12, 22 October 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-cellar-tags-12>>.
- [MCF]** "MCF specification, introduction", <<http://mukoli.free.fr/mcf/>>.
- [MSRGB]** Microsoft Corporation, "Compression Enumeration", June 2021, <https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-wmf/4e588f70-bd92-4a6f-b77f-35d0feaf7a57>.
- [MSYUV16]** Microsoft Corporation, "10-bit and 16-bit YUV Video Formats", November 2022, <<https://learn.microsoft.com/en-us/windows/win32/medfound/10-bit-and-16-bit-yuv-video-formats>>.
- [MSYUV8]** Microsoft Corporation, "Recommended 8-Bit YUV Formats for Video Rendering", January 2021, <<https://learn.microsoft.com/en-us/windows/win32/medfound/recommended-8-bit-yuv-formats-for-video-rendering>>.
- [RFC0959]** Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, DOI 10.17487/RFC0959, October 1985, <<https://www.rfc-editor.org/info/rfc959>>.
- [RFC2083]** Boutell, T., "PNG (Portable Network Graphics) Specification Version 1.0", RFC 2083, DOI 10.17487/RFC2083, March 1997, <<https://www.rfc-editor.org/info/rfc2083>>.
- [RFC3533]** Pfeiffer, S., "The Ogg Encapsulation Format Version 0", RFC 3533, DOI 10.17487/RFC3533, May 2003, <<https://www.rfc-editor.org/info/rfc3533>>.

-
- [RFC4732]** Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <<https://www.rfc-editor.org/info/rfc4732>>.
 - [RFC9110]** Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <<https://www.rfc-editor.org/info/rfc9110>>.
 - [SMB-CIFS]** Microsoft Corporation, "[MS-CIFS]: Common Internet File System (CIFS) Protocol", October 2020, <<https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-CIFS/%5bMS-CIFS%5d.pdf>>.
 - [SP800-38A]** National Institute of Standards and Technology (NIST), "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", DOI 10.6028/NIST.SP.800-38A, NIST Special Publication 800-38A, December 2001, <<https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf>>.
 - [SP800-67]** National Institute of Standards and Technology (NIST), "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher", DOI 10.6028/NIST.SP.800-67r2, NIST Special Publication 800-67, November 2017, <<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-67r2.pdf>>.
 - [Twofish]** Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., and N. Ferguson, "Twofish: A 128-Bit Block Cipher", June 1998, <<https://www.schneier.com/academic/twofish/>>.
 - [WebM-Enc]** Galligan, F., "WebM Encryption", September 2016, <<https://www.webmproject.org/docs/webm-encryption/>>.
 - [WebVTT]** Pieters, S., Pfeiffer, S., Ed., Jaegenstedt, P., and I. Hickson, "WebVTT: The Web Video Text Tracks Format", W3C Candidate Recommendation, April 2019, <<https://www.w3.org/TR/2019/CR-webvtt1-20190404/>>.

Appendix A. Historic Deprecated Elements

As Matroska has 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 defined then 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.

In this appendix, we list elements that have a known ID that **SHOULD NOT** be reused to avoid colliding with existing files. These might be reassigned by IANA in the future if there are no more IDs for a given size. A short description of what each ID was used for is included, but the text is not normative.

A.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 for seeking or deciding what track to use.

A.2. SilentTrackNumber Element

type / id: uinteger / 0x58D7

path: \Segment\Cluster\SilentTracks\SilentTrackNumber

documentation: One of the track numbers that is not used from now on in the stream. It could change later if not specified as silent in a further Cluster.

A.3. 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.

A.4. 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.

A.5. Slices Element

type / id: master / 0x8E

path: \Segment\Cluster\BlockGroup\Slices

documentation: Contains slices description.

A.6. 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.

A.7. 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.

A.8. **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 (allows for the generation of many frames from the same Block/Frame).

A.9. **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).

A.10. **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](#).

A.11. **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](#).

A.12. **ReferenceFrame Element**

type / id: master / 0xC8

path: \Segment\Cluster\BlockGroup\ReferenceFrame

documentation: Contains information about the last reference frame. See [\[DivXTrickTrack\]](#).

A.13. **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 [\[DivXTrickTrack\]](#).

A.14. 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 [\[DivXTrickTrack\]](#).

A.15. EncryptedBlock Element

type / id: binary / 0xAF

path: \Segment\Cluster\EncryptedBlock

documentation: Similar to SimpleBlock (see [Section 10.2](#)), but the data inside the Block are Transformed (encrypted and/or signed).

A.16. MinCache Element

type / id: uinteger / 0x6DE7

path: \Segment\Tracks\TrackEntry\MinCache

documentation: 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.

A.17. MaxCache Element

type / id: uinteger / 0x6DF8

path: \Segment\Tracks\TrackEntry\MaxCache

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

A.18. 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 -- i.e., in nanoseconds; see [Section 11.1](#). This can be used to adjust the playback offset of a track.

A.19. CodecSettings Element

type / id: utf-8 / 0x3A9697

path: \Segment\Tracks\TrackEntry\CodecSettings

documentation: A string describing the encoding setting used.

A.20. CodecInfoURL Element

type / id: string / 0x3B4040

path: \Segment\Tracks\TrackEntry\CodecInfoURL

documentation: A URL to find information about the codec used.

A.21. CodecDownloadURL Element

type / id: string / 0x26B240

path: \Segment\Tracks\TrackEntry\CodecDownloadURL

documentation: A URL to download about the codec used.

A.22. CodecDecodeAll Element

type / id: uinteger / 0xAA

path: \Segment\Tracks\TrackEntry\CodecDecodeAll

documentation: Set to 1 if the codec can decode potentially damaged data.

A.23. TrackOverlay Element

type / id: uinteger / 0x6FAB

path: \Segment\Tracks\TrackEntry\TrackOverlay

documentation: Specify that this track is an overlay track for the Track specified (in the integer). This means that when this track has a gap 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 the first one is not found, it should be the second, etc.

A.24. AspectRatioType Element

type / id: uinteger / 0x54B3

path: \Segment\Tracks\TrackEntry\Video\AspectRatioType

documentation: Specify the possible modifications to the aspect ratio.

A.25. GammaValue Element

type / id: float / 0x2FB523

path: \Segment\Tracks\TrackEntry\Video\GammaValue

documentation: Gamma value.

A.26. 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.

A.27. ChannelPositions Element

type / id: binary / 0x7D7B

path: \Segment\Tracks\TrackEntry\Audio\ChannelPositions

documentation: Table of horizontal angles for each successive channel.

A.28. 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 [\[DivXTrickTrack\]](#).

A.29. TrickTrackSegmentUID Element

type / id: binary / 0xC1

path: \Segment\Tracks\TrackEntry\TrickTrackSegmentUID

documentation: The SegmentUID of the Segment containing the track identified by TrickTrackUID. See [\[DivXTrickTrack\]](#).

A.30. 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 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 [\[DivXTrickTrack\]](#).

A.31. 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 [\[DivXTrickTrack\]](#).

A.32. TrickMasterTrackSegmentUID Element

type / id: binary / 0xC4
path: \Segment\Tracks\TrackEntry\TrickMasterTrackSegmentUID
documentation: The SegmentUID of the Segment containing the track identified by MasterTrackUID. See [\[DivXTrickTrack\]](#).

A.33. ContentSignature Element

type / id: binary / 0x47E3
path:
 \Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\
 ContentSignature
documentation: A cryptographic signature of the contents.

A.34. ContentSigKeyID Element

type / id: binary / 0x47E4
path:
 \Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\
 ContentSigKeyID
documentation: This is the ID of the private key that the data was signed with.

A.35. ContentSigAlgo Element

type / id: uinteger / 0x47E5
path:
 \Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\
 ContentSigAlgo
documentation: The algorithm used for the signature.

A.36. ContentSigHashAlgo Element

type / id: uinteger / 0x47E6
path:
 \Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\
 ContentSigHashAlgo
documentation: The hash algorithm used for the signature.

A.37. CueRefCluster Element

type / id: uinteger / 0x97
path: \Segment\Cues\CuePoint\CueTrackPositions\CueReference\CueRefCluster
documentation: The Segment Position of the Cluster containing the referenced Block.

A.38. 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.

A.39. 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.

A.40. 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.

A.41. 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 are based on TimestampScale. See [\[DivXWorldFonts\]](#).

A.42. 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 are based on TimestampScale. See [\[DivXWorldFonts\]](#).

A.43. 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](#).

Authors' Addresses

Steve Lhomme

Email: slhomme@matroska.org

Moritz Bunkus

Email: moritz@bunkus.org

Dave Rice

Email: dave@dericed.com