Abstract

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

This document updates [RFC8794] to permit the use of a previously reserved EBML Element ID.

Status of This Memo

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

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

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

This Internet-Draft will expire on 24 April 2024.

Copyright Notice

Copyright (c) 2023 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
2. Status of this document
3. Notation and Conventions
4. Matroska Overview
   4.1. Principles
   4.2. Updates to RFC 8794
   4.3. Added EBML Constraints
   4.4. Design Rules
   4.5. Data Layout
5. Matroska Schema
   5.1. Segment Element
      5.1.1. SeekHead Element
         5.1.1.1. Seek Element
      5.1.2. Info Element
         5.1.2.1. SegmentUUID Element
         5.1.2.2. SegmentFilename Element
         5.1.2.3. PrevUUID Element
         5.1.2.4. PrevFilename Element
         5.1.2.5. NextUUID Element
         5.1.2.6. NextFilename Element
         5.1.2.7. SegmentFamily Element
         5.1.2.8. ChapterTranslate Element
         5.1.2.9. TimestampScale Element
         5.1.2.10. Duration Element
         5.1.2.11. DateUTC Element
         5.1.2.12. Title Element
         5.1.2.13. MuxingApp Element
         5.1.2.14. WritingApp Element
      5.1.3. Cluster Element
         5.1.3.1. Timestamp Element
         5.1.3.2. Position Element
         5.1.3.3. PrevSize Element
         5.1.3.4. SimpleBlock Element
         5.1.3.5. BlockGroup Element
   5.1.4. Tracks Element
      5.1.4.1. TrackEntry Element
      5.1.5. Cues Element
         5.1.5.1. CuePoint Element
   5.1.6. Attachments Element
      5.1.6.1. AttachedFile Element
   5.1.7. Chapters Element
      5.1.7.1. EditionEntry Element
   5.1.8. Tags Element
      5.1.8.1. Tag Element
6. Matroska Element Ordering
   6.1. Top-Level Elements
   6.2. CRC-32
6.3. SeekHead
6.4. Cues (index)
6.5. Info
6.6. Chapters Element
6.7. Attachments
6.8. Tags

7. Matroska versioning
8. Stream Copy
9. DefaultDecodedFieldDuration

10. Cluster Blocks
   10.1. Block Structure
   10.2. SimpleBlock Structure
   10.3. Block Lacing
       10.3.1. No lacing
       10.3.2. Xiph lacing
       10.3.3. EBML lacing
       10.3.4. Fixed-size lacing
       10.3.5. Laced Frames Timestamp
   10.4. Random Access Points

11. Timestamps
   11.1. Timestamp Ticks
       11.1.1. Matroska Ticks
       11.1.2. Segment Ticks
       11.1.3. Track Ticks
   11.2. Block Timestamps
   11.3. TimestampScale Rounding

12. Language Codes
13. Country Codes
14. Encryption
15. Image Presentation
   15.1. Cropping
   15.2. Rotation
16. Segment Position
   16.1. Segment Position Exception
   16.2. Example of Segment Position
17. Linked Segments
   17.1. Hard Linking
   17.2. Medium Linking
       17.2.1. Linked-Duration
       17.2.2. Linked-Edition
18. Track Flags
   18.1. Default flag
   18.2. Forced flag
   18.3. Hearing-impaired flag
   18.4. Visual-impaired flag
   18.5. Descriptions flag
   18.6. Original flag
   18.7. Commentary flag
   18.8. Track Operation
18.9. Overlay Track
18.10. Multi-planar and 3D videos
19. Default track selection
   19.1. Audio Selection
   19.2. Subtitle selection
20. Chapters
   20.1. EditionEntry
       20.1.1. EditionFlagDefault
       20.1.2. Default Edition
       20.1.3. EditionFlagOrdered
           20.1.3.1. Ordered-Edition and Matroska Segment-Linking
   20.2. ChapterAtom
       20.2.1. ChapterTimeStart
       20.2.2. ChapterTimeEnd
       20.2.3. Nested Chapters
       20.2.4. Nested Chapters in Ordered Chapters
       20.2.5. ChapterFlagHidden
   20.3. Menu features
   20.4. Physical Types
   20.5. Chapter Examples
       20.5.1. Example 1 : basic chaptering
       20.5.2. Example 2 : nested chapters
           20.5.2.1. The Micronauts "Bleep To Bleep"
21. Attachments
   21.1. Cover Art
   21.2. Font files
22. Cues
   22.1. Recommendations
23. Matroska Streaming
   23.1. File Access
   23.2. Livestreaming
24. Tags
   24.1. Tags Precedence
   24.2. Tag Levels
25. Implementation Recommendations
   25.1. Cluster
   25.2. SeekHead
   25.3. Optimum Layouts
       25.3.1. Optimum layout for a muxer
       25.3.2. Optimum layout after editing tags
       25.3.3. Optimum layout with Cues at the front
       25.3.4. Optimum layout for livestreaming
26. Security Considerations
27. IANA Considerations
   27.1. Matroska Element IDs Registry
   27.2. Chapter Codec IDs Registry
   27.3. Media Types
       27.3.1. For files containing video tracks
       27.3.2. For files containing audio tracks with no video tracks
27.3.3. For files containing a stereoscopic video track


28.1. SilentTracks Element
28.2. SilentTrackNumber Element
28.3. BlockVirtual Element
28.4. ReferenceVirtual Element
28.5. Slices Element
28.6. TimeSlice Element
28.7. LaceNumber Element
28.8. FrameNumber Element
28.9. BlockAdditionID Element
28.10. Delay Element
28.11. SliceDuration Element
28.12. ReferenceFrame Element
28.13. ReferenceOffset Element
28.15. EncryptedBlock Element
28.16. MinCache Element
28.17. MaxCache Element
28.18. TrackOffset Element
28.19. CodecSettings Element
28.20. CodecInfoURL Element
28.21. CodecDownloadURL Element
28.22. CodecDecodeAll Element
28.23. TrackOverlay Element
28.24. AspectRatioType Element
28.25. GammaValue Element
28.26. FrameRate Element
28.27. ChannelPositions Element
28.28. TrickTrackUID Element
28.29. TrickTrackSegmentUID Element
28.30. TrickTrackFlag Element
28.31. TrickMasterTrackUID Element
28.32. TrickMasterTrackSegmentUID Element
28.33. ContentSignature Element
28.34. ContentSigKeyID Element
28.35. ContentSigAlgo Element
28.36. ContentSigHashAlgo Element
28.37. CueRefCluster Element
28.38. CueRefNumber Element
28.39. CueRefCodecState Element
28.40. FileReferral Element
28.41. FileUsedStartTime Element
28.42. FileUsedEndTime Element
28.43. TagDefaultBogus Element

29. Normative References

30. Informative References

Authors' Addresses
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.

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

*It is NOT a video or audio compression format (codec)
*It is an envelope 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 DVDs have [DVD-Video])

2. Status of this document

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

3. Notation and Conventions

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

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

Matroska: A multimedia container format based on EBML (Extensible Binary Meta Language).
Matroska Reader: A data parser that interprets the semantics of a Matroska document and creates a way for programs to use Matroska.
Matroska Player: A Matroska Reader with a primary purpose of playing audiovisual files, including Matroska documents.
Matroska Writer: A data writer that creates Matroska documents.

4. Matroska Overview

4.1. Principles

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

4.2. 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. Specifically, the following are changed in [RFC8794]:

*From Errata 7189

In Section 17.1,

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.

*From Errata 7191

In Section 5,
4.3. Added EBML Constraints

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

* The docType of the EBML Header **MUST** be "matroska".
* The EBMLMaxIDLength of the EBML Header **MUST** be 4.
* The EBMLMaxSizeLength of the EBML Header **MUST** be between 1 and 8 inclusive.

4.4. Design Rules

The Root Element and all Top-Levels 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 present in the file but with a length of zero. They always assumed the value was 0 for integers/dates or 0x0p+0, the textual expression of floats using the ISO9899 format, no matter the default value of the element which should have been used instead. Therefore, Matroska writers **MUST NOT** use EBML Empty Elements, if the element has a default value that is not 0 for integers/dates and 0x0p+0 for floats.

When adding new elements to Matroska, these rules 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 string 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 which may occur within the Segment.

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

*EBML Header
*Segment

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

*EBML Header
*Segment
*EBML Header
*Segment

The following diagram represents a simple Matroska file, comprised of an EBML Document with an EBML Header, a Segment Element (the Root Element), and all eight Matroska Top-Level Elements. In the following diagrams of this section, horizontal spacing expresses a parent-child relationship between Matroska Elements (e.g., the Info Element is contained within the Segment Element) whereas vertical alignment represents the storage order within the file.
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 Elements.
The Info Element contains vital information for identifying the whole Segment. This includes the title for the Segment, a randomly generated unique identifier, and the unique identifier(s) of any linked Segment Elements.

| +-------------------------+ |
| | Info | SegmentUUID       | |
| | | ------------------| |
| | | SegmentFilename   | |
| | | ------------------| |
| | | PrevUUID          | |
| | | ------------------| |
| | | PrevFilename      | |
| | | ------------------| |
| | | NextUUID          | |
| | | ------------------| |
| | | NextFilename      | |
| | | ------------------| |
| | | SegmentFamily     | |
| | | ------------------| |
| | | ChapterTranslate  | |
| | | ------------------| |
| | | TimestampScale    | |
| | | ------------------| |
| | | Duration          | |
| | | ------------------| |
| | | DateUTC           | |
| | | ------------------| |
| | | Title             | |
| | | ------------------| |
| | | MuxingApp         | |
| | | ------------------| |
| | | WritingApp        | |
| | ------------------| |

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

The Tracks Element defines the technical details for each track and can store the name, number, unique identifier, language, and type (audio, video, subtitles, etc.) of each track. For example, the Tracks Element MAY store information about the resolution of a video track or sample rate of an audio track.

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

<table>
<thead>
<tr>
<th>Tracks</th>
<th>TrackEntry</th>
<th>TrackNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TrackUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TrackType</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Language</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CodecID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CodecPrivate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CodecName</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Video</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FlagInterlaced</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FieldOrder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StereoMode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AlphaMode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PixelWidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PixelHeight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DisplayWidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DisplayHeight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AspectRatioType</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Audio</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SamplingFrequency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Channels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BitDepth</td>
</tr>
</tbody>
</table>
The Chapters Element lists all of the chapters. Chapters are a way to set predefined points to jump to in video or audio.

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.

---

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Position</td>
</tr>
<tr>
<td></td>
<td>PrevSize</td>
</tr>
<tr>
<td></td>
<td>SimpleBlock</td>
</tr>
<tr>
<td></td>
<td>BlockGroup</td>
</tr>
</tbody>
</table>

---

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

---

<table>
<thead>
<tr>
<th>Block</th>
<th>Portion of</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a Block</td>
<td>- Bit Flag</td>
</tr>
<tr>
<td></td>
<td>Header</td>
<td>TrackNumber</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Gap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Lacing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Reserved</td>
</tr>
<tr>
<td></td>
<td>Optional</td>
<td>FrameSize</td>
</tr>
<tr>
<td></td>
<td>Data</td>
<td>Frame</td>
</tr>
</tbody>
</table>

---

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, timestamp in the Block itself is relative to the Timestamp Element in the Cluster. For example, if the Timestamp Element in the Cluster is set to 10 seconds and a Block in that Cluster is supposed to be played 12 seconds into the clip, the timestamp in the Block would be set to 2 seconds.

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

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

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

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

```
+-------------------------------------+
| Cues | CuePoint | CueTime          |
|      |          |                 |
|      |          | ----------------|
|      |          | CueTrackPositions|
|      |------------------------------|
|      | CuePoint | CueTime          |
|      |          |                 |
|      |          | ----------------|
|      |          | CueTrackPositions|
+-------------------------------------+
```

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

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

<table>
<thead>
<tr>
<th>Attachments</th>
<th>AttachedFile</th>
<th>FileDescription</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FileName</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FileMediaType</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FileData</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FileUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FileReferral</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FileUsedStartTime</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FileUsedEndTime</td>
</tr>
</tbody>
</table>

Figure 9: Representation of an Attachments Element.
Figure 10: Representation of a Tags Element and three levels of its Children Elements.

5. Matroska Schema

This specification includes an EBML Schema, which defines the Elements and structure of Matroska 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.

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

Here the definition of each Matroska Element is provided.

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

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 unique ID to identify the Segment amongst many others (128 bits). It is equivalent to a UUID v4 [RFC4122] with all bits randomly (or pseudo-randomly) 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 unique ID **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 to identify 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 to identify 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 unique ID that all Segments of a Linked Segment MUST share (128 bits). It is equivalent to a UUID v4 [RFC4122] with all bits randomly (or pseudo-randomly) 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 segment when stored in Matroska. This element and its child elements add a way to map the internal segments known to the Chapter Codec to the Segment IDs in Matroska. This allows remuxing a file with Chapter Codec without changing the content of the codec data, just the Segment mapping.

5.1.2.8.1. ChapterTranslateID Element

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:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Matroska</td>
<td>Chapter commands using the Matroska Script codec.</td>
</tr>
<tr>
<td>1</td>
<td>DVD-menu</td>
<td>Chapter commands using the DVD-like codec.</td>
</tr>
</tbody>
</table>

Table 1: ChapterTranslateCodec values

5.1.2.8.3. ChapterTranslateEditionUID Element

id / type: 0x69FC / integer
path: \\Segment\Info\ChapterTranslate\ChapterTranslateEditionUID
definition: Specify a chapter edition UID on which this ChapterTranslate applies.
usage notes: When no ChapterTranslateEditionUID is specified in the ChapterTranslate, the ChapterTranslate applies to all chapter editions found in the Segment using the given ChapterTranslateCodec.

5.1.2.9. TimestampScale Element

id / type / default: 0x2AD7B1 / integer / 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 is 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 is based on TimestampScale; see Section 11.1.
usage notes: This element SHOULD be the first child element of the Cluster it belongs to, or the second if that Cluster contains a CRC-32 element (Section 6.2).

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

5.1.3.5.2.1. BlockMore Element

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

5.1.3.5.2.2. BlockAdditional Element

id / type: 0xA5 / binary
path: \\Segment\\Cluster\\BlockGroup\\BlockAdditions\\BlockMore\\BlockAdditiona
l
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 to identify how to interpret the BlockAdditional
data; see Codec BlockAdditions section of [MatroskaCodec] for
more information. A value of 1 indicates that the meaning of the
BlockAdditional data is defined by the codec. Any other value
indicates the meaning of the BlockAdditional data is found in the
BlockAddIDType found in the TrackEntry.
usage notes: Each BlockAddID value MUST be unique between all
BlockMore elements found in a BlockAdditions.
usage notes: To keep MaxBlockAdditionID as low as possible, small
values SHOULD be used.

5.1.3.5.3. 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:

attribute note

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

default When not written and with no DefaultDuration, the value is assumed to be the difference between the 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 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 this Block depends on. Historically Matroska Writer didn't write the actual Block(s) this Block depends on, but some Block in the past.

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

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

5.1.3.5.6. 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
min: 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 value, at the beginning of the Block for negative value). The duration of DiscardPadding is not calculated in the duration of the TrackEntry and **SHOULD** be discarded during playback.

### 5.1.4. **Tracks Element**

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 unique ID to identify the Track.
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:

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

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

5.1.4.1.7. FlagHearingImpaired Element

id / type: 0x55AB / uinteger
range: 0-1
path: \Segment\Tracks\TrackEntry\FlagHearingImpaired
maxOccurs: 1
minver: 4
definition: Set to 1 if and only if that 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 that 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 that 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 that track is in the content's original language.

5.1.4.1.11. FlagCommentary Element

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

5.1.4.1.12. FlagLacing Element

id / type / default: 0x9C / integer / 1
range: 0-1
path: \\Segment\\Tracks\\TrackEntry\\FlagLacing
minOccurs / maxOccurs: 1 / 1
definition: Set to 1 if the track MAY contain blocks using lacing.
When set to 0 all blocks MUST have their lacing flags set to No lacing; see Section 10.3 on Block Lacing.

5.1.4.1.13. DefaultDuration Element

id / type: 0x23E383 / integer
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 / integer
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 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\BlockAdditionMapping\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\BlockAdditionMapping\BlockAddIDExtraDa
ta
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 [BCP47] form; 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: CodecDelay is The codec-built-in delay, expressed in
Matroska Ticks -- i.e., in nanoseconds; see Section 11.1. It
represents the amount of codec samples that will be discarded by
the decoder during playback. This timestamp value MUST be
subtracted from each frame timestamp in order to get the
timestamp that will be actually played. The value SHOULD be small
so the muxing of tracks with the same actual timestamp are in the
same Cluster.

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, SeekPreRoll is the duration of
the data 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 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 this chapter codec of the given chapter edition(s); see [Section 5.1.7.1.4.15](#).

**defined values:**

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Matroska</td>
<td>Chapter commands using the Matroska Script codec.</td>
</tr>
<tr>
<td>1</td>
<td>DVD-menu</td>
<td>Chapter commands using the DVD-like codec.</td>
</tr>
</tbody>
</table>

Table 4: TrackTranslateCodec values

5.1.4.1.27.3. **TrackTranslateEditionUID Element**

id / type: 0x66FC / uinteger

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

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

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

5.1.4.1.28. 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: Specify whether the video frames in this track are interlaced.

defined values:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>undetermined</td>
<td>Unknown status. This value SHOULD be avoided.</td>
</tr>
<tr>
<td>1</td>
<td>interlaced</td>
<td>Interlaced frames.</td>
</tr>
<tr>
<td>2</td>
<td>progressive</td>
<td>No interlacing.</td>
</tr>
</tbody>
</table>

Table 5: FlagInterlaced values

stream copy: True (Section 8)

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

defined values:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>progressive</td>
<td>Interlaced frames. This value SHOULD be avoided, setting FlagInterlaced to 2 is sufficient.</td>
</tr>
<tr>
<td>1</td>
<td>tff</td>
<td>Top field displayed first. Top field stored first.</td>
</tr>
<tr>
<td>2</td>
<td>undetermined</td>
<td>Unknown field order. This value SHOULD be avoided.</td>
</tr>
<tr>
<td>6</td>
<td>bff</td>
<td>Bottom field displayed first. Bottom field stored first.</td>
</tr>
<tr>
<td>9</td>
<td>bff(swapped)</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: FieldOrder values
Top field displayed first. Fields are interleaved in storage with the top line of the top field stored first.
Bottom field displayed first. Fields are interleaved in storage with the top line of the top field stored first.

Table 6: FieldOrder values

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

stream copy: True (Section 8)

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. There are some more details in Section 18.10.

restrictions:

Table 7: StereoMode values

stream copy: True (Section 8)

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: Indicate 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:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>none</td>
<td>The BlockAdditional Element with BlockAddID of &quot;1&quot; does not exist or SHOULD NOT be considered as containing such data.</td>
</tr>
<tr>
<td>1</td>
<td>present</td>
<td>The BlockAdditional Element with BlockAddID of &quot;1&quot; contains alpha channel data.</td>
</tr>
</tbody>
</table>

Table 8: AlphaMode values

stream copy: True (Section 8)

5.1.4.1.28.5. OldStereoMode Element

id / type: 0x53B9 / integer
path: \Segment\Tracks\TrackEntry\Video\OldStereoMode

maxOccurs: 1
maxver: 2

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

restrictions:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>mono</td>
</tr>
<tr>
<td>1</td>
<td>right eye</td>
</tr>
<tr>
<td>2</td>
<td>left eye</td>
</tr>
<tr>
<td>3</td>
<td>both eyes</td>
</tr>
</tbody>
</table>

Table 9: OldStereoMode values

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

5.1.4.1.28.6. PixelWidth Element

id / type: 0xB0 / integer
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
- **minOccur / 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
- **minOccur / 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
- **minOccur / 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
- **minOccur / 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
- **minOccur / 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:

attribute note
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

stream copy: True (Section 8)

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:

attribute note
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

stream copy: True (Section 8)

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 & DisplayHeight are interpreted.

restrictions:
Table 12: DisplayUnit values

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>pixels</td>
</tr>
<tr>
<td>1</td>
<td>centimeters</td>
</tr>
<tr>
<td>2</td>
<td>inches</td>
</tr>
<tr>
<td>3</td>
<td>display aspect ratio</td>
</tr>
<tr>
<td>4</td>
<td>unknown</td>
</tr>
</tbody>
</table>

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: Specify the uncompressed pixel format used for the Track's data as a FourCC. This value is similar in scope to the biCompression value of AVI's BITMAPINFO [AVIFormat]. There is no 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:

attribute note
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

stream copy: True (Section 8)

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

<table>
<thead>
<tr>
<th>id / type / default</th>
<th>0x55B1 / integer / 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>\Segment\Tracks\TrackEntry\Video\Colour\MatrixCoefficients</td>
</tr>
<tr>
<td>minOccurs / maxOccurs</td>
<td>1 / 1</td>
</tr>
<tr>
<td>minver</td>
<td>4</td>
</tr>
<tr>
<td>definition</td>
<td>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].</td>
</tr>
</tbody>
</table>

restrictions:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Identity</td>
</tr>
<tr>
<td>1</td>
<td>ITU-R BT.709</td>
</tr>
<tr>
<td>2</td>
<td>unspecified</td>
</tr>
<tr>
<td>3</td>
<td>reserved</td>
</tr>
<tr>
<td>4</td>
<td>US FCC 73.682</td>
</tr>
<tr>
<td>5</td>
<td>ITU-R BT.470BG</td>
</tr>
<tr>
<td>6</td>
<td>SMPTE 170M</td>
</tr>
<tr>
<td>7</td>
<td>SMPTE 240M</td>
</tr>
<tr>
<td>8</td>
<td>YCoCg</td>
</tr>
<tr>
<td>9</td>
<td>BT2020 Non-constant Luminance</td>
</tr>
<tr>
<td>10</td>
<td>BT2020 Constant Luminance</td>
</tr>
<tr>
<td>11</td>
<td>SMPTE ST 2085</td>
</tr>
<tr>
<td>12</td>
<td>Chroma-derived Non-constant Luminance</td>
</tr>
<tr>
<td>13</td>
<td>Chroma-derived Constant Luminance</td>
</tr>
<tr>
<td>14</td>
<td>ITU-R BT.2100-0</td>
</tr>
</tbody>
</table>

Table 14: MatrixCoefficients values

stream copy: True (Section 8)

5.1.4.1.28.18. BitsPerChannel Element

<table>
<thead>
<tr>
<th>id / type / default</th>
<th>0x55B2 / integer / 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>\Segment\Tracks\TrackEntry\Video\Colour\BitsPerChannel</td>
</tr>
<tr>
<td>minOccurs / maxOccurs</td>
<td>1 / 1</td>
</tr>
<tr>
<td>minver</td>
<td>4</td>
</tr>
<tr>
<td>definition</td>
<td>Number of decoded bits per channel. A value of 0 indicates that the BitsPerChannel is unspecified.</td>
</tr>
</tbody>
</table>

stream copy: True (Section 8)

5.1.4.1.28.19. ChromaSubsamplingHorz Element

<table>
<thead>
<tr>
<th>id / type</th>
<th>0x55B3 / integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>\Segment\Tracks\TrackEntry\Video\Colour\ChromaSubsamplingHorz</td>
</tr>
<tr>
<td>maxOccurs</td>
<td>1</td>
</tr>
</tbody>
</table>


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

\text{stream \ copy: \ True (Section \ 8)}

5.1.4.1.28.20. \text{ChromaSubsamplingVert Element}
\text{id / type:} \ 0x55B4 / \text{integer}
\text{path:} \ \text{\backslash Segment\backslash \TrackEntry\backslash Video\backslash \Colour\ChromaSubsamplingVert}
\text{maxOccurs:} \ 1
\text{minver:} \ 4
\text{definition:} \ The \ amount \ of \ pixels \ to \ remove \ in \ the \ Cr \ and \ Cb 
channels \ for \ every \ pixel \ not \ removed \ vertically. \ Example: \ For 
video \ with \ 4:2:0 \ chroma \ subsampling, \ the \ ChromaSubsamplingVert
\text{SHOULD} \ be \ set \ to \ 1.

\text{stream \ copy: \ True (Section \ 8)}

5.1.4.1.28.21. \text{CbSubsamplingHorz Element}
\text{id / type:} \ 0x55B5 / \text{integer}
\text{path:} \ \text{\backslash Segment\backslash \TrackEntry\backslash Video\backslash \Colour\CbSubsamplingHorz}
\text{maxOccurs:} \ 1
\text{minver:} \ 4
\text{definition:} \ The \ amount \ of \ pixels \ to \ remove \ in \ the \ Cb \ channel \ for 
every \ pixel \ not \ removed \ horizontally. \ This \ is \ additive \ with 
ChromaSubsamplingHorz. \ Example: \ For \ video \ with \ 4:2:1 \ chroma 
subsampling, \ the \ ChromaSubsamplingHorz \text{SHOULD} \ be \ set \ to \ 1 \ and 
CbSubsamplingHorz \text{SHOULD} \ be \ set \ to \ 1.

\text{stream \ copy: \ True (Section \ 8)}

5.1.4.1.28.22. \text{CbSubsamplingVert Element}
\text{id / type:} \ 0x55B6 / \text{integer}
\text{path:} \ \text{\backslash Segment\backslash \TrackEntry\backslash Video\backslash \Colour\CbSubsamplingVert}
\text{maxOccurs:} \ 1
\text{minver:} \ 4
\text{definition:} \ The \ amount \ of \ pixels \ to \ remove \ in \ the \ Cb \ channel \ for 
every \ pixel \ not \ removed \ vertically. \ This \ is \ additive \ with 
ChromaSubsamplingVert.

\text{stream \ copy: \ True (Section \ 8)}

5.1.4.1.28.23. \text{ChromaSitingHorz Element}
\text{id / type / default:} \ 0x55B7 / \text{integer / 0}
\text{path:} \ \text{\backslash Segment\backslash \TrackEntry\backslash Video\backslash \Colour\ChromaSitingHorz}
\text{minOccurs / maxOccurs:} \ 1 / 1
minver: 4
definition: How chroma is subsampled horizontally.

restrictions:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>unspecified</td>
</tr>
<tr>
<td>1</td>
<td>left collocated</td>
</tr>
<tr>
<td>2</td>
<td>half</td>
</tr>
</tbody>
</table>

Table 15: ChromaSitingHorz values

stream copy: True (Section 8)

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:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>unspecified</td>
</tr>
<tr>
<td>1</td>
<td>top collocated</td>
</tr>
<tr>
<td>2</td>
<td>half</td>
</tr>
</tbody>
</table>

Table 16: ChromaSitingVert values

stream copy: True (Section 8)

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:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>unspecified</td>
</tr>
</tbody>
</table>
value label
1 broadcast range
2 full range (no clipping)
3 defined by MatrixCoefficients / TransferCharacteristics

Table 17: Range values

stream copy: True (Section 8)

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:

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

stream copy: True (Section 8)

5.1.4.1.28.27. Primaries Element

id / type / default: 0x55BB / uinteger / 2
The colour primaries of the video. For clarity, the value and meanings for Primaries are adopted from Table 2 of [ITU-H.273].

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>reserved</td>
</tr>
<tr>
<td>1</td>
<td>ITU-R BT.709</td>
</tr>
<tr>
<td>2</td>
<td>unspecified</td>
</tr>
<tr>
<td>3</td>
<td>reserved2</td>
</tr>
<tr>
<td>4</td>
<td>ITU-R BT.470M</td>
</tr>
<tr>
<td>5</td>
<td>ITU-R BT.470BG - BT.601 625</td>
</tr>
<tr>
<td>6</td>
<td>ITU-R BT.601 525 - SMPTE 170M</td>
</tr>
<tr>
<td>7</td>
<td>SMPTE 240M</td>
</tr>
<tr>
<td>8</td>
<td>FILM</td>
</tr>
<tr>
<td>9</td>
<td>ITU-R BT.2020</td>
</tr>
<tr>
<td>10</td>
<td>SMPTE ST 428-1</td>
</tr>
<tr>
<td>11</td>
<td>SMPTE RP 432-2</td>
</tr>
<tr>
<td>12</td>
<td>SMPTE EG 432-2</td>
</tr>
<tr>
<td>22</td>
<td>EBU Tech. 3213-E - JEDEC P22 phosphors</td>
</tr>
</tbody>
</table>

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/m²).

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\Primary
  RChromaticityX
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\Primary
  RChromaticityY
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\Primary
  GChromaticityX
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\PrimaryGChromaticityY  
**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\PrimaryBChromaticityX  
**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\PrimaryBChromaticityY  
**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:** >= 0x0p+0
- **path:** 
  \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\LuminanceMax
- **maxOccurs:** 1
- **minver:** 4
- **definition:** Maximum luminance. Represented in candelas per square meter (cd/m²).

  stream copy: True (Section 8)
5.1.4.1.28.40. LuminanceMin Element

- **id / type:** 0x55DA / float
- **range:** >= 0x0p+0
- **path:** \Segment\Tracks\TrackEntry\Video\Colour\MasteringMetadata\LuminanceMin
- **maxOccurs:** 1
- **minver:** 4
- **definition:** Minimum luminance. Represented in candelas per square meter (cd/m²).

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, VR videos or flipping videos horizontally/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:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>rectangular</td>
</tr>
<tr>
<td>1</td>
<td>equirectangular</td>
</tr>
<tr>
<td>2</td>
<td>cubemap</td>
</tr>
<tr>
<td>3</td>
<td>mesh</td>
</tr>
</tbody>
</table>

Table 20: ProjectionType values

Stream copy: True (Section 8)

5.1.4.1.28.43. ProjectionPrivate Element

- **id / type:** 0x7672 / binary
- **path:** \Segment\Tracks\TrackEntry\Video\Projection\ProjectionPrivate
- **maxOccurs:** 1

Stream copy: True (Section 8)
**Miner:** 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 the 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

<table>
<thead>
<tr>
<th>id / type / default</th>
<th>0x7675 / float / 0x0p+0</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>&gt;= -0xB4p+0, &lt;= 0xB4p+0</td>
</tr>
<tr>
<td>path</td>
<td>\Segment\Tracks\TrackEntry\Video\Projection\ProjectionPoseRoll</td>
</tr>
<tr>
<td>minOccurs / maxOccurs</td>
<td>1 / 1</td>
</tr>
<tr>
<td>minver</td>
<td>4</td>
</tr>
</tbody>
</table>

**definition:** Specifies a roll rotation to the projection.

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

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

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

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

### 5.1.4.1.29. Audio Element

| id / type | 0xE1 / master |
|---------------------------------|
| path               | \Segment\Tracks\TrackEntry\Audio |

**definition:** Audio settings.

#### 5.1.4.1.29.1. SamplingFrequency Element

<table>
<thead>
<tr>
<th>id / type / default</th>
<th>0xB5 / float / 0x1.f4p+12</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>&gt; 0x0p+0</td>
</tr>
<tr>
<td>path</td>
<td>\Segment\Tracks\TrackEntry\Audio\SamplingFrequency</td>
</tr>
</tbody>
</table>

**definition:** Sampling frequency in Hz.

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

#### 5.1.4.1.29.2. OutputSamplingFrequency Element

<table>
<thead>
<tr>
<th>id / type</th>
<th>0x78B5 / float</th>
</tr>
</thead>
</table>
range:  > 0x0P+0
path:  \Segment\Tracks\TrackEntry\Audio\OutputSamplingFrequency
maxOccurs:  1
definition:  Real output sampling frequency in Hz (used for SBR techniques).

notes:

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

Table 21: OutputSamplingFrequency implementation notes

5.1.4.1.29.3. Channels Element

id / type / default:  0x0F / 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 look at 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 need 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\TrackPlaneUID  
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\TrackPlaneType  
minOccurs / maxOccurs: 1 / 1  
minver: 3  
definition: The kind of plane this track corresponds to.

restrictions:
<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left eye</td>
</tr>
<tr>
<td>1</td>
<td>right eye</td>
</tr>
<tr>
<td>2</td>
<td>background</td>
</tr>
</tbody>
</table>

Table 22: TrackPlaneType values

stream copy: True (Section 8)

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\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\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 over 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:

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

Table 23: ContentEncodingScope values

stream copy: True (Section 8)

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 what kind of transformation is applied.

restrictions:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Compression</td>
</tr>
<tr>
<td>1</td>
<td>Encryption</td>
</tr>
</tbody>
</table>

Table 24: ContentEncodingType values

stream copy: True ([Section 8])

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 not to 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:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>zlib</td>
<td>zlib compression [RFC1950].</td>
</tr>
<tr>
<td>1</td>
<td>bzlib</td>
<td>bzip2 compression [BZIP2], SHOULD NOT be used; see usage notes.</td>
</tr>
</tbody>
</table>
Table 25: ContentCompAlgo values

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>lzo1x</td>
<td>Lempel-Ziv-Oberhumer compression [LZO], SHOULD NOT be used; see usage notes.</td>
</tr>
<tr>
<td>3</td>
<td>Header Stripping</td>
<td>(Section 5.1.4.1.31.7) have been stripped from each frame.</td>
</tr>
</tbody>
</table>

Usage notes: Compression method "1" (bzlib) and "2" (lzo1x) are lacking proper documentation on the format which limits implementation possibilities. Due to licensing conflicts on commonly available libraries compression methods "2" (lzo1x) does not offer widespread interoperability. 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)

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 frames 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:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not encrypted</td>
<td>The data are not encrypted.</td>
</tr>
<tr>
<td>1</td>
<td>DES</td>
<td>Data Encryption Standard (DES) [FIPS.46-3].This value SHOULD be avoided.</td>
</tr>
<tr>
<td>2</td>
<td>3DES</td>
<td>Triple Data Encryption Algorithm [SP.800-67].This value SHOULD be avoided.</td>
</tr>
<tr>
<td>3</td>
<td>Twofish</td>
<td>Twofish Encryption Algorithm [Twofish].</td>
</tr>
<tr>
<td>4</td>
<td>Blowfish</td>
<td>Blowfish Encryption Algorithm [Blowfish].This value SHOULD be avoided.</td>
</tr>
<tr>
<td>5</td>
<td>AES</td>
<td>Advanced Encryption Standard (AES) [FIPS.197].</td>
</tr>
</tbody>
</table>

Table 26: ContentEncAlgo values

stream copy: True (Section 8)

5.1.4.1.31.10.  ContentEncKeyID Element

id / type: 0x47E2 / binary
path: \Segment\Tracks\TrackEntry\ContentEncodings\ContentEncoding\ContentEncryption\ContentEncKeyID
maxOccurs: 4
definition: For public key algorithms this is the ID of the public key 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:
attribute note
maxOccurs ContentEncAESSettings **MUST NOT** be set (maxOccurs=0) if ContentEncAlgo is not AES (5).

Table 27: ContentEncAESSettings implementation notes

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

### 5.1.4.1.31.12. AESSettingsCipherMode Element

id / type: 0x47E8 / integer

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:**

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AES-CTR</td>
<td>Counter <a href="#">SP.800-38A</a>.</td>
</tr>
<tr>
<td>2</td>
<td>AES-CBC</td>
<td>Cipher Block Chaining <a href="#">SP.800-38A</a>.</td>
</tr>
</tbody>
</table>

Table 28: AESSettingsCipherMode values

notes:

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

Table 29: AESSettingsCipherMode implementation notes

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

### 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:

attribute note
minOccurs
attribute note
   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: Contain 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
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**: Contain 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

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 [RFC6838] format.
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: Unique ID 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, look at the Chapters explanation in 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

stream copy: True ([Section 8](#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](#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 (apply 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 unique ID to identify the Chapter.

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

5.1.7.1.4.2. ChapterStringUID Element
**id / type:** 0x5654 / utf-8  
**path:** \Segment\Chapters\EditionEntry\+ChapterAtom\ChapterStringUID  
**maxOccurs:** 1  
**definition:** A unique string ID to identify the Chapter. For example it is used as the storage for [WebVTT] cue identifier values.

### 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:** The ChapterTimeEnd timestamp value being excluded, it MUST take in account the duration of the last frame it includes, especially for the ChapterAtom using the last frames of the Segment.

**notes:**

**attribute note**

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

Table 31: ChapterTimeEnd implementation notes

### 5.1.7.1.4.5. ChapterFlagHidden Element

**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:**

**attribute note**
ChapterSegmentUUID **MUST** be set (minOccurs=1) if

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

### 5.1.7.1.4.9. ChapterDisplay Element

**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 [BCP47] form; 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 the processing. A value of 0 means built-in Matroska processing (to be defined), a value of 1 means the DVD command set is used; see Section 20.3 on DVD menus. More codec IDs can be added later.

5.1.7.1.4.16. ChapProcessPrivate Element

id / type: 0x450D / binary
path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapProcess\ChapProcessPrivate
maxOccurs: 1
definition: Some optional data attached to the ChapProcessCodecID information. For ChapProcessCodecID = 1, it is the "DVD level" equivalent; see Section 20.3 on DVD menus.
5.1.7.1.4.17. ChapProcessCommand Element

id / type: 0x6911 / master
path: \Segment\Chapters\EditionEntry\+ChapterAtom\ChapProcess\ChapProcessCommand
definition: Contains all the commands associated to 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:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>during the whole chapter</td>
</tr>
<tr>
<td>1</td>
<td>before starting playback</td>
</tr>
<tr>
<td>2</td>
<td>after playback of the chapter</td>
</tr>
</tbody>
</table>

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:

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

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 like "ALBUM", "TRACK", "MOVIE", "CHAPTER", etc.

restrictions:

<table>
<thead>
<tr>
<th>value</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLECTION</td>
<td>TargetTypeValue 70</td>
</tr>
<tr>
<td>EDITION</td>
<td>TargetTypeValue 60</td>
</tr>
<tr>
<td>ISSUE</td>
<td>TargetTypeValue 60</td>
</tr>
<tr>
<td>VOLUME</td>
<td>TargetTypeValue 60</td>
</tr>
<tr>
<td>value</td>
<td>label</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>OPUS</td>
<td>TargetTypeValue 60</td>
</tr>
<tr>
<td>SEASON</td>
<td>TargetTypeValue 60</td>
</tr>
<tr>
<td>SEQUEL</td>
<td>TargetTypeValue 60</td>
</tr>
<tr>
<td>ALBUM</td>
<td>TargetTypeValue 50</td>
</tr>
<tr>
<td>OPERA</td>
<td>TargetTypeValue 50</td>
</tr>
<tr>
<td>CONCERT</td>
<td>TargetTypeValue 50</td>
</tr>
<tr>
<td>MOVIE</td>
<td>TargetTypeValue 50</td>
</tr>
<tr>
<td>EPISODE</td>
<td>TargetTypeValue 50</td>
</tr>
<tr>
<td>PART</td>
<td>TargetTypeValue 40</td>
</tr>
<tr>
<td>SESSION</td>
<td>TargetTypeValue 40</td>
</tr>
<tr>
<td>TRACK</td>
<td>TargetTypeValue 30</td>
</tr>
<tr>
<td>SONG</td>
<td>TargetTypeValue 30</td>
</tr>
<tr>
<td>CHAPTER</td>
<td>TargetTypeValue 30</td>
</tr>
<tr>
<td>SUBTRACK</td>
<td>TargetTypeValue 20</td>
</tr>
<tr>
<td>MOVEMENT</td>
<td>TargetTypeValue 20</td>
</tr>
<tr>
<td>SCENE</td>
<td>TargetTypeValue 20</td>
</tr>
<tr>
<td>SHOT</td>
<td>TargetTypeValue 10</td>
</tr>
</tbody>
</table>

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 unique ID to identify the Track(s) the tags belong to.
usage notes: If the value is 0 at this level, the tags apply to all tracks in the Segment. If set to any other value, it MUST match the TrackUID value of a track found in this Segment.

5.1.8.1.1.4. TagEditionUID Element

id / type / default: 0x63C9 / uinteger / 0
path: \
Segment\Tags\Tag\Targets\TagEditionUID
definition: A unique ID to identify the EditionEntry(s) the tags belong to.
usage notes: If the value is 0 at this level, the tags apply to all editions in the Segment. If set to any other value, it MUST match the EditionUID value of an edition found in this Segment.

5.1.8.1.1.5. TagChapterUID Element

id / type / default: 0x63C4 / uinteger / 0
path: \
Segment\Tags\Tag\Targets\TagChapterUID
definition: A unique ID to identify the Chapter(s) the tags belong to.
usage notes: If the value is 0 at this level, the tags apply to all chapters in the Segment. If set to any other value, it MUST match the ChapterUID value of a chapter found in this Segment.

5.1.8.1.1.6. TagAttachmentUID Element

id / type / default: 0x63C6 / uinteger / 0
path: \
Segment\Tags\Tag\Targets\TagAttachmentUID
definition: A unique ID to identify the Attachment(s) the tags belong to.

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

5.1.8.1.2. SimpleTag Element

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

5.1.8.1.2.3. TagLanguageBCP47 Element

id: 0x447B / string
path: \Segment\Tags\Tag\+SimpleTag\TagLanguageBCP47
maxOccurs: 1
minver: 4
definition: The language used in the TagString, in the BCP47 form; 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: 0x4487 / utf-8
path: \Segment\Tags\Tag\+SimpleTag\TagString
maxOccurs: 1
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

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

6.1. Top-Level Elements

The Info Element is the only REQUIRED Top-Level Element in a Matroska file. To be playable, Matroska MUST also contain at least one Tracks Element and Cluster Element. The first Info Element and the first Tracks Element MUST either be stored before the first Cluster Element or both SHALL be referenced by a SeekHead Element occurring before the first Cluster Element.

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

When using Medium Linking, chapters are used to reference other Segments to play in a given order 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) itemize the identity and position of all Top-Level Elements.

Editing, removing, or adding Elements to a Matroska file often requires that some existing Elements be voided or extended. Transforming the existing Elements into Void Elements as padding can be used as a method to avoid moving large amounts of data around.
6.2. CRC-32

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

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

6.3. SeekHead

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

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

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

6.4. Cues (index)

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

6.5. Info

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

6.6. Chapters Element

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

### 6.7. Attachments

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

### 6.8. Tags

The Tags Element is most subject to changes after the file was originally created. For easier editing, the Tags Element 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. So it's better if the Tags Element is found early in the stream. When editing the Tags Element, the original Tags Element at the beginning can be overwritten with a Void Element and a new Tags Element written at the end of the Segment Element. The file and Segment sizes will only marginally change.

### 7. Matroska versioning

Matroska is based upon the principle that a reading application does not have to support 100% of the specifications in order to be able to play the file. A Matroska file therefore contains version indicators that tell a reading application what to expect. It is possible and valid to have the version fields indicate that the file contains Matroska Elements from a higher specification version number while signaling that a reading application **MUST** only support a lower version number properly in order to play it back (possibly with a reduced feature set).

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

**DocTypeVersion** **MUST** be equal to or greater than the highest Matroska version number of any Element present in the Matroska file. For
example, a file using the SimpleBlock Element (Section 5.1.3.4) 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 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 file. If that property is not set, elements may or may not keep the same value between the source and destination.

9. DefaultDecodedFieldDuration

The DefaultDecodedFieldDuration Element can signal to the displaying application how often fields of a video sequence will be available for displaying. It can be used for both interlaced and progressive content.

If the video sequence is signaled as interlaced Section 5.1.4.1.2.8.1, 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 ns/(48/1.001) = 20854167 ns
*PAL broadcast/DVD: 1000000000 ns/(50/1.000) = 20000000 ns
*N/ATSC broadcast: 1000000000 ns/(60/1.001) = 16683333 ns
*hard-telecined DVD: 1000000000 ns/(60/1.001) = 16683333 ns (60 
encoded interlaced fields per second)
*soft-telecined DVD: 1000000000 ns/(60/1.001) = 16683333 ns (48 
encoded interlaced fields per second, with "repeat_first_field = 
1")

10. Cluster Blocks

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

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

*in a Block which is contained inside a BlockGroup,
 *or in a SimpleBlock which 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,
 *optionally the lacing information,
 *the consecutive frame(s)

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

The TrackNumber is coded using the 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. We only provide the definitions for TrackNumber sizes of 1 and 2. The other variants can be deduced by extending the size of the TrackNumber by multiples of 8 bits.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               |                               |       |I|LAC|U|
|  Track Number |         Timestamp             | Rsvrd |N|ING|N|
|               |                               |       |V|   |U|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Block Header with 1 octet TrackNumber

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Track Number         |         Timestamp             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       |I|LAC|U|
| Rsvrd |N|ING|N|                     ...
|       |V|   |U|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Block Header with 2 octets 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
using 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 correspond 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 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. We only provide the definitions for TrackNumber sizes of 1 and 2. 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

0  | 1  | 2  | 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------------------------------------+
|               |                               |K|     |I|LAC|D|
|  Track Number |         Timestamp             |E|Rsvrd|N|ING|I|
|               |                               |Y|     |V|   |S|
+-------------------------------------------------------------+

Figure 13: SimpleBlock Header with 1 octet TrackNumber
Figure 14: SimpleBlock Header with 2 octets 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
using 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 correspond 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 3 types of lacing:

1. Xiph, inspired by what is found in the Ogg container [RFC3533]
2. EBML, which is the same with sizes coded differently
3. fixed-size, where the size is not coded
When lacing is not used, i.e. to store a single frame, the lacing bits 5 and 6 of the Block or SimpleBlock MUST be set to zero.

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

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

10.3.1. 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. The bits 5-6 of the Block Header flags are set to 0b00.

The Block for an 800 octets frame is as follows:

<table>
<thead>
<tr>
<th>Block Octets</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-803</td>
<td>&lt;frame&gt;</td>
<td>Single frame data</td>
</tr>
</tbody>
</table>

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]. The bits 5-6 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 frames are stored with Xiph lacing in a Block as follows:

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

Table 37: Xiph lacing example

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

### 10.3.3. EBML lacing

The EBML lacing encodes the frame size with an EBML-like encoding [RFC8794]. The bits 5-6 of the Block Header flags are set to 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 Variable-Size Integer value, also known as VINT in [RFC8794]. 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 $2^{(({7n})-1)}$, where n is the octet size of the VINT.

<table>
<thead>
<tr>
<th>Bit Representation of signed VINT</th>
<th>Possible Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xxx xxxx</td>
<td>$2^7$ values from $-(2^6-1)$ to $2^6$</td>
</tr>
<tr>
<td>01xx xxxx xxxx xxxx</td>
<td>$2^{14}$ values from $-(2^{13}-1)$ to $2^{13}$</td>
</tr>
<tr>
<td>001x xxxx xxxx xxxx xxxx</td>
<td>$2^{21}$ values from $-(2^{20}-1)$ to $2^{20}$</td>
</tr>
<tr>
<td>0001 xxxx xxxx xxxx xxxx xxxx xxx</td>
<td></td>
</tr>
</tbody>
</table>


### Bit Representation of signed VINT

<table>
<thead>
<tr>
<th>Bit Representation</th>
<th>Possible Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 1xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx</td>
<td>$2^{28}$ values from $-(2^{27}-1)$ to $2^{27}$</td>
</tr>
<tr>
<td>xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx</td>
<td>$2^{35}$ values from $-(2^{34}-1)$ to $2^{34}$</td>
</tr>
</tbody>
</table>

Table 38: EBML Lacing signed VINT bits usage

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

<table>
<thead>
<tr>
<th>Block Octets</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0x02</td>
<td>Number of frames minus 1</td>
</tr>
<tr>
<td>5-6</td>
<td>0x43</td>
<td>Size of the first frame (800 = 0x320 + 0x4000)</td>
</tr>
<tr>
<td>7-8</td>
<td>0x5E</td>
<td>Size of the second frame (500 - 800 = -300 = -0x12C + 0x1FFF + 0x4000)</td>
</tr>
<tr>
<td>8-807</td>
<td>&lt;frame1&gt;</td>
<td>First frame data</td>
</tr>
<tr>
<td>808-1307</td>
<td>&lt;frame2&gt;</td>
<td>Second frame data</td>
</tr>
<tr>
<td>1308-2307</td>
<td>&lt;frame3&gt;</td>
<td>Third frame data</td>
</tr>
</tbody>
</table>

Table 39: EBML lacing example

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

#### 10.3.4. Fixed-size lacing

The Fixed-size lacing doesn't store the frame size, only the number of frames in the lace. Each frame **MUST** have the same size. The frame size of each frame is deduced from the total size of the Block. The bits 5-6 of the Block Header flags are set to 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 3 frames of 800 octets each:

<table>
<thead>
<tr>
<th>Block Octets</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0x02</td>
<td>Number of frames minus 1</td>
</tr>
<tr>
<td>5-804</td>
<td>&lt;frame1&gt;</td>
<td>First frame data</td>
</tr>
<tr>
<td>805-1604</td>
<td>&lt;frame2&gt;</td>
<td>Second frame data</td>
</tr>
<tr>
<td>1605-2404</td>
<td>&lt;frame3&gt;</td>
<td>Third frame data</td>
</tr>
</tbody>
</table>

Table 40: Fixed-size lacing example
This gives a Block of 2405 octets. When reading the Block we find that there are 3 frames (Octet 4). The data start at Octet 5, so the size of each frame is \((2405 - 5) / 3 = 800\).

### 10.3.5. 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. But each of these frames **MUST** be contiguous -- i.e. the decoded data **MUST NOT** contain any gap between them. If there is a gap in the stream, the frames around the gap **MUST NOT** be in the same Block.

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

### 10.4. Random Access Points

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

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

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

*BlockGroup with a frame that references another frame, with the EBML tree shown as XML:
Frames that are RAP -- i.e. they don't depend on other frames -- MUST set the keyframe flag if they are in a SimpleBlock or their parent BlockGroup MUST NOT contain a ReferenceBlock.

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

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

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

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

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

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

```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:

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

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

```xml
<Cluster>
  <Timestamp>123456</Timestamp>
  <BlockGroup>
    <!-- References itself to mark it should not be used as RAP -->
    <ReferenceBlock>0</ReferenceBlock>
    <BlockDuration>20</BlockDuration>
    <Block/>
  </BlockGroup>
...
</Cluster>
```

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

11. Timestamps

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

11.1. Timestamp Ticks

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

11.1.1. Matroska Ticks

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

The elements storing values in Matroska Ticks/nanoseconds are:

- TrackEntry\DefaultDuration; defined in Section 5.1.4.1.13
- 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:

\[
\text{timestamp in nanosecond} = \text{element value} \times \text{TimestampScale}
\]

This allows storing smaller integer values in the elements.

When using the default value of TimestampScale of "1,000,000", one Segment Tick represents one millisecond.
The elements storing values in Segment Ticks are:

*Cluster\Timestamp; defined in Section 5.1.3.1
*Info\Duration is stored as a floating-point but the same formula applies; defined in Section 5.1.2.10
*CuePoint\CueTrackPositions\CueDuration; defined in Section 5.1.5.1.2.4

11.1.3. Track Ticks

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

\[
\text{timestamp in nanoseconds} = \text{element value} \times \text{TrackTimestampScale} \times \text{TimestampScale}
\]

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

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

The elements storing values in Track Ticks are:

*Cluster\BlockGroup\Block and Cluster\SimpleBlock timestamps; detailed in Section 11.2
*Cluster\BlockGroup\BlockDuration; defined in Section 5.1.3.5.3
*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 in account. So 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 you have to use the following formula:

\[
( \text{Cluster\Timestamp} + ( \text{block timestamp} \times \text{TrackTimestampScale} ) ) \times \text{TimestampScale}
\]

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

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

\[
( ( \text{Cluster\Timestamp} + ( \text{block timestamp} \times \text{TrackTimestampScale} ) ) \times \text{TimestampScale} ) - \text{CodecDelay}
\]

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, which 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 nanosecond to get the proper nanosecond timestamp of a Block. This allows some clever TimestampScale values to have more refined timestamp precision per frame.

### 12. Language Codes

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

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

13. Country Codes

Country codes are the [BCP47] two-letter region subtag, 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 subfields (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 are encrypted. But each ContentEncAlgo Element and its sub elements like AESSettingsCipherMode really define how the encrypted should be exactly 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 stream.

15. **Image Presentation**

15.1. **Cropping**

The PixelCrop Elements (PixelCropTop, PixelCropBottom, PixelCropRight, and PixelCropLeft) indicate when, and by how much, encoded videos frames **SHOULD** be cropped for display. These Elements allow edges of the frame that are not intended for display, such as the sprockets of a full-frame film scan or the VANC area of a digitized analog videotape, to be stored but hidden. PixelCropTop and PixelCropBottom store an integer of how many rows of pixels **SHOULD** be cropped from the top and bottom of the image (respectively). PixelCropLeft and PixelCropRight store an integer of how many columns of pixels **SHOULD** be cropped from the left and right of the image (respectively).

For example, a pillar-boxed video that stores a 1440x1080 visual image within the center of a padded 1920x1080 encoded image may set both PixelCropLeft and PixelCropRight to "240", so that a Matroska Player should crop off 240 columns of pixels from the left and right of the encoded image to present the image with the pillar-boxes hidden.

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

15.2. **Rotation**

The ProjectionPoseRoll Element (see 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 representation 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:
16. Segment Position

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

16.1. Segment Position Exception

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

16.2. Example of Segment Position

This table presents an example of Segment Position by showing a hexadecimal representation of a very small Matroska file with labels to show the offsets in octets. The file contains a Segment Element with an Element ID of "0x1853067" and a MuxingApp Element with an Element ID of "0x4D80".

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

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

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

17. Linked Segments

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

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

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

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

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

17.1. Hard Linking

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

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

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

With Hard Linking, the chapters of any Segment within the Linked Segment **MUST** only reference the current Segment. The NextUUID and PrevUUID reference the respective SegmentUUID values of the next and previous Segments.
The first Segment of a Linked Segment **MUST NOT** have a PrevUUID Element. The last Segment of a Linked Segment **MUST NOT** have a NextUUID Element.

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

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

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

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

<table>
<thead>
<tr>
<th>file name</th>
<th>SegmentUUID</th>
<th>PrevUUID</th>
<th>NextUUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>start.mkv</td>
<td>71000c23cd310998</td>
<td>Invalid</td>
<td>a77b3598941cb803</td>
</tr>
<tr>
<td></td>
<td>53fbc94dd984a5dd</td>
<td></td>
<td>eac0fcdafe44fac9</td>
</tr>
<tr>
<td>middle.mkv</td>
<td>a77b3598941cb803</td>
<td>71000c23cd310998</td>
<td>6c92285fa63e827</td>
</tr>
<tr>
<td></td>
<td>eac0fcdafe44fac9</td>
<td>53fbc94dd984a5dd</td>
<td>b198d120ea3ac674</td>
</tr>
<tr>
<td>end.mkv</td>
<td>6c92285fa63e827</td>
<td>a77b3598941cb803</td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td>b198d120ea3ac674</td>
<td>eac0fcdafe44fac9</td>
<td></td>
</tr>
</tbody>
</table>

Table 41: Usual Hard Linking UIDs

An other example where only the NextUUID Element is used:

<table>
<thead>
<tr>
<th>file name</th>
<th>SegmentUUID</th>
<th>PrevUUID</th>
<th>NextUUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>start.mkv</td>
<td>71000c23cd310998</td>
<td>Invalid</td>
<td>a77b3598941cb803</td>
</tr>
<tr>
<td></td>
<td>53fbc94dd984a5dd</td>
<td></td>
<td>eac0fcdafe44fac9</td>
</tr>
<tr>
<td>middle.mkv</td>
<td>a77b3598941cb803</td>
<td>n/a</td>
<td>6c92285fa63e827</td>
</tr>
<tr>
<td></td>
<td>eac0fcdafe44fac9</td>
<td></td>
<td>b198d120ea3ac674</td>
</tr>
<tr>
<td>end.mkv</td>
<td>6c92285fa63e827</td>
<td>n/a</td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td>b198d120ea3ac674</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 42: Hard Linking without PrevUUID

An example where only the PrevUUID Element is used:

<table>
<thead>
<tr>
<th>file name</th>
<th>SegmentUUID</th>
<th>PrevUUID</th>
<th>NextUUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>start.mkv</td>
<td>Invalid</td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 43: Hard Linking without NextUUID

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

<table>
<thead>
<tr>
<th>file name</th>
<th>SegmentUUID</th>
<th>PrevUUID</th>
<th>NextUUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>middle.mkv</td>
<td>71000c23cd310998</td>
<td>71000c23cd310998</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>53fbc94dd984a5dd</td>
<td>53fbc94dd984a5dd</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a77b3598941cb803</td>
<td>a77b3598941cb803</td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td>eac0fcdafe44fac9</td>
<td>eac0fcdafe44fac9</td>
<td></td>
</tr>
<tr>
<td>end.mkv</td>
<td>6c92285fa6d3e827</td>
<td>6c92285fa6d3e827</td>
<td>b198d120ea3ac674</td>
</tr>
<tr>
<td></td>
<td>b198d120ea3ac674</td>
<td>b198d120ea3ac674</td>
<td></td>
</tr>
</tbody>
</table>

Table 44: Hard Linking with mixed UID links

<table>
<thead>
<tr>
<th>file name</th>
<th>SegmentUUID</th>
<th>PrevUUID</th>
<th>NextUUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>start.mkv</td>
<td>71000c23cd310998</td>
<td>Invalid</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>53fbc94dd984a5dd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>middle.mkv</td>
<td>a77b3598941cb803</td>
<td>71000c23cd310998</td>
<td>6c92285fa6d3e827</td>
</tr>
<tr>
<td></td>
<td>eac0fcdafe44fac9</td>
<td>53fbc94dd984a5dd</td>
<td>b198d120ea3ac674</td>
</tr>
<tr>
<td>end.mkv</td>
<td>6c92285fa6d3e827</td>
<td>n/a</td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td>b198d120ea3ac674</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 optionally a ChapterTimeEnd element. The second chapter references content within the Segment of program.mkv. A Matroska Player SHOULD recognize the Linked Segment created by the use of ChapterSegmentUUID in an enabled Edition and present the reference content of the two Segments as a single presentation.
The ChapterSegmentUUID represents the Segment that holds the content to play in place of the Linked Chapter. The ChapterSegmentUUID **MUST NOT** be the SegmentUUID of its own Segment.

There are 2 ways to use a chapter link:

*Linked-Duration linking,*

*Linked-Edition linking*

### 17.2.1. Linked-Duration

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

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

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

### 17.2.2. Linked-Edition

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

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

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

### 18. Track Flags

#### 18.1. Default flag

The "default track" flag is a hint for a Matroska Player indicating that a given track **SHOULD** be eligible to be automatically selected as the default track for a given language. If no tracks in a given language have the default track flag set, then all tracks in that language are eligible for automatic selection. This can be used to indicate that a track provides "regular service" suitable for users with default settings, as opposed to specialized services, such as commentary, hearing-impaired captions, or descriptive audio.

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

The "forced" flag tells the Matroska Player that it **SHOULD** display this subtitle track, even if user preferences usually would not call for any subtitles to be displayed alongside the current selected audio track. This can be used to indicate that a track contains translations of onscreen text, or of dialogue spoken in a different language than the track's primary one.

18.3. Hearing-impaired flag

The "hearing impaired" flag tells the Matroska Player that it **SHOULD** prefer this track when selecting a default track for a hearing-impaired user, and that it **MAY** prefer to select a different track when selecting a default track for a non-hearing-impaired user.

18.4. Visual-impaired flag

The "visual impaired" flag tells the Matroska Player that it **SHOULD** prefer this track when selecting a default track for a visually-impaired user, and that it **MAY** prefer to select a different track when selecting a default track for a non-visually-impaired user.

18.5. Descriptions flag

The "descriptions" flag tells the Matroska Player that this track is suitable to play via a text-to-speech system for a visually-impaired user, and that it **SHOULD NOT** automatically select this track when selecting a default track for a non-visually-impaired user.

18.6. Original flag

The "original" flag tells the Matroska Player that this track is in the original language, and that it **SHOULD** prefer it if configured to prefer original-language tracks of this track's type.

18.7. Commentary flag

The "commentary" flag tells the Matroska Player that this track contains commentary on the content.

18.8. Track Operation

TrackOperation allows combining multiple tracks to make a virtual one. It uses two separate system to combine tracks. One to create a 3D "composition" (left/right/background planes) and one to simplify join two tracks together to make a single track.

A track created with TrackOperation is a proper track with a UID and all its flags. However, the codec ID is meaningless because each
"sub" track needs to be decoded by its own decoder before the "operation" is applied. The Cues Elements corresponding to such a virtual track **SHOULD** be the 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 amount of pixels, for example 3840x1080 for full HD side by side, and the DisplayWidth/DisplayHeight in pixels is the amount of pixels for one plane (1920x1080 for that full HD stream). Old stereo 3D were displayed using anaglyph (cyan and red colors separated). For compatibility with such movies, there is a value of the StereoMode that corresponds to AnaGlyph.

There is also a "packed" mode (values 13 and 14) which consists of packing two frames together in a Block using lacing. The first frame is the left eye and the other frame is the right eye (or vice versa). The frames **SHOULD** be decoded in that order and are possibly dependent on each other (P and B frames).

For separate tracks, Matroska needs to define exactly which track does what. TrackOperation with TrackCombinePlanes do that. For more details look at [Section 18.8](#) on how TrackOperation works.
The 3D support is still in infancy and may evolve to support more features.

The StereoMode used to be part of Matroska v2 but it didn't meet the requirement for multiple tracks. There was also a bug in libmatroska prior to 0.9.0 that would save/read it as 0x53B9 instead of 0x53B8; see OldStereoMode (Section 5.1.4.1.28.5). 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, 3: both eyes, the only values that can be found in OldStereoMode. They are not compatible with the StereoMode values found in Matroska v3 and above.

19. Default track selection

This section provides some example sets of Tracks and hypothetical user settings, along with indications of which ones a similarly-configured Matroska Player SHOULD automatically select for playback by default in such a situation. A player MAY provide additional settings with more detailed controls for more nuanced scenarios. These examples are provided as guidelines to illustrate the intended usages of the various supported Track flags, and their expected behaviors.

Track names are shown in English for illustrative purposes; actual files may have titles in the language of each track, or provide titles in multiple languages.

19.1. Audio Selection

Example track set:

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Lang</th>
<th>Layout</th>
<th>Original</th>
<th>Default</th>
<th>Other flags</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Video</td>
<td>und</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>Audio</td>
<td>eng</td>
<td>5.1</td>
<td>1</td>
<td>1</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>3</td>
<td>Audio</td>
<td>eng</td>
<td>2.0</td>
<td>1</td>
<td>1</td>
<td>None</td>
<td>Visual-impaired</td>
</tr>
<tr>
<td>4</td>
<td>Audio</td>
<td>eng</td>
<td>2.0</td>
<td>1</td>
<td>0</td>
<td>Visual-impaired</td>
<td>Descriptive audio</td>
</tr>
<tr>
<td>5</td>
<td>Audio</td>
<td>esp</td>
<td>5.1</td>
<td>0</td>
<td>1</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>6</td>
<td>Audio</td>
<td>esp</td>
<td>2.0</td>
<td>0</td>
<td>0</td>
<td>Visual-impaired</td>
<td>Descriptive audio</td>
</tr>
<tr>
<td>7</td>
<td>Audio</td>
<td>eng</td>
<td>2.0</td>
<td>1</td>
<td>0</td>
<td>None</td>
<td>Commentary</td>
</tr>
<tr>
<td>8</td>
<td>Audio</td>
<td>eng</td>
<td>2.0</td>
<td>1</td>
<td>0</td>
<td>None</td>
<td>Karaoke</td>
</tr>
</tbody>
</table>

Table 45: Audio Tracks for default selection
Here we have a file with 7 audio tracks, of which 5 are in English and 2 are in Spanish.

The English tracks all have the Original flag, indicating that English is the original content language.

Generally the player will first consider the track languages: if the player has an option to prefer original-language audio and the user has enabled it, then it should prefer one of the Original-flagged tracks. If configured to specifically prefer audio tracks in English or Spanish, the player should select one of the tracks in the corresponding language. The player may also wish to prefer an Original-flagged track if no tracks matching any of the user's explicitly-preferred languages are available.

Two of the tracks have the Visual-impaired flag. If the player has been configured to prefer such tracks, it should select one; otherwise, it should avoid them if possible.

If selecting an English track, when other settings have left multiple possible options, it may be useful to exclude the tracks that lack the Default flag: here, one provides descriptive service for the visually impaired (which has its own flag and may be automatically selected by user configuration, but is unsuitable for users with default-configured players), one is a commentary track (which has its own flag, which the player may or may not have specialized handling for), and the last contains karaoke versions of the music that plays during the film, which is an unusual specialized audio service that Matroska has no built-in support for indicating, so it's indicated in the track name instead. By not setting the Default flag on these specialized tracks, the file's author hints that they should not be automatically selected by a default-configured player.

Having narrowed its choices down, our example player now may have to select between tracks 2 and 3. The only difference between these tracks is their channel layouts: 2 is 5.1 surround, while 3 is stereo. If the player is aware that the output device is a pair of headphones or stereo speakers, it may wish to prefer the stereo mix automatically. On the other hand, if it knows that the device is a surround system, it may wish to prefer the surround mix.

If the player finishes analyzing all of the available audio tracks and finds that multiple seems equally and maximally preferable, it SHOULD default to the first of the group.

19.2. Subtitle selection

Example track set:
Table 46: Subtitle Tracks for default selection

Here we have 2 audio tracks and 5 subtitle tracks. As we can see, French is the original language.

We'll start by discussing the case where the user prefers French (or Original-language) audio (or has explicitly selected the French audio track), and also prefers French subtitles.

In this case, if the player isn't configured to display captions when the audio matches their preferred subtitle languages, the player doesn't need to select a subtitle track at all.

If the user has indicated that they want captions to be displayed, the selection simply comes down to whether Hearing-impaired subtitles are preferred.

The situation for a user who prefers Portuguese subtitles starts out somewhat analogous. If they select the original French audio (either by explicit audio language preference, preference for Original-language tracks, or by explicitly selecting that track), then the selection once again comes down to the hearing-impaired preference.

However, the case where the Portuguese audio track is selected has an important catch: a Forced track in Portuguese is present. This may contain translations of onscreen text from the video track, or of portions of the audio that are not translated (music, for instance). This means that even if the user's preferences wouldn't normally call for captions here, the Forced track should be selected nonetheless, rather than selecting no track at all. On the other hand, if the user's preferences do call for captions, the non-Forced tracks should be preferred, as the Forced track will not contain captioning for the dialogue.
20. Chapters

The Matroska Chapters system can have multiple Editions and each Edition can consist of Simple Chapters where a chapter start time is used as marker in the timeline only. An Edition can be more complex with Ordered Chapters where a chapter end time stamp is additionally used or much more complex with Linked Chapters. The Matroska Chapters system can also have a menu structure, borrowed from the DVD menu system [DVD-Video], or have 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 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.

<table>
<thead>
<tr>
<th>Edition</th>
<th>FlagDefault</th>
<th>Default Edition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edition 1</td>
<td>true</td>
<td>X</td>
</tr>
<tr>
<td>Edition 2</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>Edition 3</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

Table 47: Default edition, all default

<table>
<thead>
<tr>
<th>Edition</th>
<th>FlagDefault</th>
<th>Default Edition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edition 1</td>
<td>false</td>
<td>X</td>
</tr>
<tr>
<td>Edition 2</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>Edition 3</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

Table 48: Default edition, no default
### EditionFlagOrdered

The EditionFlagOrdered Flag is a significant feature as it enables an Edition of Ordered Chapters which defines and arranges a virtual timeline rather than simply labeling points within the timeline. For example, with Editions of Ordered Chapters a single Matroska file can present multiple edits of a film without duplicating content. Alternatively, if a videotape is digitized in full, one Ordered Edition could present the full content (including colorbars, countdown, slate, a feature presentation, and black frames), while another Edition of Ordered Chapters can use Chapters that only mark the intended presentation with the colorbars and other ancillary visual information excluded. If an Edition of Ordered Chapters is enabled, then the Matroska Player **MUST** play those Chapters in their stored order from the timestamp marked in the ChapterTimeStart Element to the timestamp marked in to ChapterTimeEnd Element.

If the EditionFlagOrdered Flag evaluates to "0", Simple Chapters are used and only the ChapterTimeStart of a Chapter is used as chapter mark to jump to the predefined point in the timeline. With Simple Chapters, a Matroska Player **MUST** ignore certain Chapter Elements. In that case these elements are informational only.

The following list shows the different Chapter elements only found in Ordered Chapters.

**Ordered Chapter elements**

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

**Table 50: elements only found in ordered chapters**

Furthermore, there are other EBML Elements which could be used if the EditionFlagOrdered evaluates to "1".
20.1.3.1. Ordered-Edition and Matroska Segment-Linking

*Hard Linking: Ordered-Chapters supersedes the Hard Linking.

*Medium Linking: Ordered Chapters are used in a normal way and can be combined with the ChapterSegmentUUID element which establishes a link to another Segment.

See Section 17 on the Linked Segments for more information about Hard Linking and Medium Linking.

20.2. ChapterAtom

The ChapterAtom is also called a Chapter.

20.2.1. ChapterTimeStart

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

20.2.2. ChapterTimeEnd

The timestamp of the end of Chapter with nanosecond accuracy, not scaled by TimestampScale. The timestamp defined by the ChapterTimeEnd is not part of the Chapter. A Matroska Player calculates the duration of this Chapter using the difference between the ChapterTimeEnd and ChapterTimeStart. The end timestamp MUST be greater than or equal to the start timestamp.

When the ChapterTimeEnd timestamp is equal to the ChapterTimeStart timestamp, the timestamps is included in the Chapter. It can be useful to put markers in a file or add chapter commands with ordered chapter commands without having to play anything; see Section 5.1.7.1.4.14.

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Start timestamp</th>
<th>End timestamp</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1</td>
<td>0</td>
<td>10000000000</td>
<td>10000000000</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>10000000000</td>
<td>5000000000</td>
<td>4000000000</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>60000000000</td>
<td>60000000000</td>
<td>0</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>90000000000</td>
<td>80000000000</td>
<td>Invalid (-1000000000)</td>
</tr>
</tbody>
</table>

Table 51: ChapterTimeEnd usage possibilities

20.2.3. Nested Chapters

A ChapterAtom element can contain other ChapterAtom elements. That element is a Parent Chapter and the ChapterAtom elements it contains are Nested Chapters.
Nested Chapters can be useful to tag small parts of a Segment that already have tags or add Chapter Codec commands on smaller parts of a Segment that already have Chapter Codec commands.

The ChapterTimeStart of a Nested Chapter **MUST** be greater than or equal to the ChapterTimeStart its Parent Chapter.

If the Parent Chapter of a Nested Chapter has a ChapterTimeEnd, the ChapterTimeStart of that Nested Chapter **MUST** be smaller than or equal to the ChapterTimeEnd of the Parent Chapter.

### 20.2.4. Nested Chapters in Ordered Chapters

The ChapterTimeEnd of the lowest level of Nested Chapters **MUST** be set for Ordered Chapters.

When used with Ordered Chapters, the ChapterTimeEnd value of a Parent Chapter is useless for playback as the proper playback sections are described in its Nested Chapters. The ChapterTimeEnd **SHOULD NOT** be set in Parent Chapters and **MUST** be ignored for playback.

### 20.2.5. ChapterFlagHidden

Each Chapter ChapterFlagHidden flag works independently of parent chapters. A Nested Chapter with a ChapterFlagHidden that evaluates to "0" remains visible in the user interface even if the Parent Chapter ChapterFlagHidden flag is set to "1".

<table>
<thead>
<tr>
<th>Chapter + Nested Chapter</th>
<th>ChapterFlagHidden</th>
<th>visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>Nested Chapter 1.1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>Nested Chapter 1.2</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>Nested Chapter 2.1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>Nested Chapter 2.2</td>
<td>1</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 52: ChapterFlagHidden nested visibility

### 20.3. Menu features

The menu features are handled like a chapter codec. That means each codec has a type, some private data and some data in the chapters.

The type of the menu system is defined by the ChapProcessCodecID parameter. For now, only 2 values are supported : 0 matroska script, 1 menu borrowed from the DVD [DVD-Video]. The private data depend on the type of menu system (stored in ChapProcessPrivate), idem for the data in the chapters (stored in ChapProcessData).
The menu system, as well as Chapter Codecs in general, can do actions on the Matroska Player like jumping to another Chapter or Edition, selecting different tracks and possibly more. The scope of all the possibilities of Chapter Codecs is not covered in this document as it depends on the Chapter Codec features and its integration in a Matroska Player.

### 20.4. Physical Types

Each level can have different meanings for audio and video. The ORIGINAL_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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Audio</th>
<th>Video</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>SET / PACKAGE</td>
<td>SET / PACKAGE</td>
<td>the collection of different media</td>
</tr>
<tr>
<td>60</td>
<td>CD / 12&quot; / 10&quot; / 7&quot; / TAPE / MINIDISC / DAT</td>
<td>DVD / VHS / LASERDISC</td>
<td>the physical medium like a CD or a DVD when the original medium (LP/DVD) has different sides</td>
</tr>
<tr>
<td>50</td>
<td>SIDE</td>
<td>SIDE</td>
<td>another physical level on DVDs</td>
</tr>
<tr>
<td>40</td>
<td>-</td>
<td>LAYER</td>
<td>as found on CDs and DVDs</td>
</tr>
<tr>
<td>30</td>
<td>SESSION</td>
<td>SESSION</td>
<td>as found on audio CDs</td>
</tr>
<tr>
<td>20</td>
<td>TRACK</td>
<td>-</td>
<td>the first logical level of the side/medium</td>
</tr>
<tr>
<td>10</td>
<td>INDEX</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Table 53: ChapterPhysicalEquiv meaning per track type

### 20.5. Chapter Examples

#### 20.5.1. Example 1: basic chaptering

In this example a movie is split in different chapters. It could also just be an audio file (album) on which each track corresponds to a chapter.

*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:

```xml
<EBML version="1.0.0">
  <!-- EBML tree structure here -->
</EBML>
```
<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>
  </EditionEntry>
</Chapters>
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_0+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:
<Chipots>
  <EditionEntry>
    <EditionUID>1281690858003401414</EditionUID>
    <ChapterAtom>
      <ChapterUID>1</ChapterUID>
      <ChapterTimeStart>0</ChapterTimeStart>
      <ChapterTimeEnd>748000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChipString>Baby wants to Bleep/Rock</ChipString>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>2</ChapterUID>
      <ChapterTimeStart>0</ChapterTimeStart>
      <ChapterTimeEnd>278000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChipString>Baby wants to bleep (pt.1)</ChipString>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>3</Chapter UID>
      <ChapterTimeStart>278000000</ChapterTimeStart>
      <ChapterTimeEnd>432000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChipString>Baby wants to rock</ChipString>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>4</ChapterUID>
      <ChapterTimeStart>432000000</ChapterTimeStart>
      <ChapterTimeEnd>633000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChipString>Baby wants to bleep (pt.2)</ChipString>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>5</ChapterUID>
      <ChapterTimeStart>633000000</ChapterTimeStart>
      <ChapterTimeEnd>748000000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChipString>Baby wants to bleep (pt.3)</ChipString>
      </ChapterDisplay>
    </ChapterAtom>
    <ChapterAtom>
      <ChapterUID>6</ChapterUID>
      <ChapterTimeStart>750000000</ChapterTimeStart>
      <ChapterTimeEnd>1178500000</ChapterTimeEnd>
      <ChapterDisplay>
        <ChipString>Bleeper_0+2</ChipString>
      </ChapterDisplay>
    </ChapterAtom>
  </EditionEntry>
</Chipots>
<ChapterDisplay><ChapterAtom><ChapterUID>7</ChapterUID><ChapterTimeStart>1180500000</ChapterTimeStart><ChapterTimeEnd>1340000000</ChapterTimeEnd><ChapterDisplay><ChapString>Baby wants to bleep (pt. 4)</ChapString></ChapterDisplay></ChapterAtom><ChapterAtom><ChapterUID>8</ChapterUID><ChapterTimeStart>1342000000</ChapterTimeStart><ChapterTimeEnd>1518000000</ChapterTimeEnd><ChapterDisplay><ChapString>Bleep to bleep</ChapString></ChapterDisplay></ChapterAtom><ChapterAtom><ChapterUID>9</ChapterUID><ChapterTimeStart>1520000000</ChapterTimeStart><ChapterTimeEnd>2015000000</ChapterTimeEnd><ChapterDisplay><ChapString>Baby wants to bleep (k)</ChapString></ChapterDisplay></ChapterAtom><ChapterAtom><ChapterUID>10</ChapterUID><ChapterTimeStart>2017000000</ChapterTimeStart><ChapterTimeEnd>2668000000</ChapterTimeEnd><ChapterDisplay><ChapString>Bleeper</ChapString></ChapterDisplay></ChapterAtom></EditionEntry></Chapters>
21. Attachments

Matroska supports storage of related files and data in the Attachments Element (a Top-Level Element). Attachment Elements can be used to store related cover art, font files, transcripts, reports, error recovery files, picture, or text-based annotations, copies of specifications, or other ancillary files related to the Segment.

Matroska Readers MUST NOT execute files stored as Attachment Elements.

21.1. Cover Art

This section defines a set of guidelines for the storage of cover art in Matroska files. A Matroska Reader MAY use embedded cover art to display a representational still-image depiction of the multimedia contents of the Matroska file.

Only JPEG and PNG [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 -- for example, 960x600 for landscape, 600x800 for portrait, or 600x600 for square. The dimension of the small cover SHOULD be 120 pixels on the smallest side -- for example, 192x120 or 120x160.

Versions of cover art can be differentiated by the filename, which is stored in the FileName Element. The default filename of the normal cover in square or portrait mode is cover.(jpg|png). When stored, the normal cover SHOULD be the first Attachment in storage order. The small cover SHOULD be prefixed with "small_", such as small_cover.(jpg|png). The landscape variant SHOULD be suffixed with "_land", such as cover_land.(jpg|png). The filenames are case-sensitive.

The following table provides examples of file names for cover art in Attachments.
21.2. Font files

Font files **MAY** be added to a Matroska file as Attachments so that the font file may be used to display an associated subtitle track. This allows the presentation of a Matroska file to be consistent in various environments where the needed fonts might not be available on the local system.

Depending on the font format in question, each font file can contain multiple font variants. Each font variant has a name which will be referred to as Font Name from now on. This Font Name can be different 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/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: TTF Font Type,
- *font/otf: OpenType Layout (OTF) Font Type,
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: TTF fonts, equivalent to font/ttf,
*application/vnd.ms-opentype: OpenType Layout fonts, equivalent to font/otf
*application/font-sfnt: Generic SFNT Font Type, equivalent to font/sfnt
*application/font-woff: WOFF 1.0, equivalent to font/woff

There may also be some font attachments with the application/octet-stream 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 which each MUST reference an absolute timestamp (via the CueTime Element), a Track (via the CueTrack Element), and a Segment Position (via the CueClusterPosition Element). Additional non-mandated
Elements are part of the CuePoint Element such as CueDuration, CueRelativePosition, CueCodecState and others which provide any Matroska Reader with additional information to use in the optimization of seeking performance.

22.1. Recommendations

The following recommendations are provided to optimize Matroska performance.

*Unless Matroska is used as a live stream, it SHOULD contain a Cues Element.

*For each video track, each keyframe SHOULD be referenced by a CuePoint Element.

*It is RECOMMENDED to not reference non-keyframes of video tracks in Cues unless it references a Cluster Element which contains a CodecState Element but no keyframes.

*For each subtitle track present, each subtitle frame SHOULD be referenced by a CuePoint Element with a CueDuration Element.

*References to audio tracks MAY be skipped in CuePoint Elements if a video track is present. When included the CuePoint Elements SHOULD reference audio keyframes at most once every 500 milliseconds.

*If the referenced frame is not stored within the first SimpleBlock, or first BlockGroup within its Cluster Element, then the CueRelativePosition Element SHOULD be written to reference where in the Cluster the reference frame is stored.

*If a CuePoint Element references Cluster Element that includes a CodecState Element, then that CuePoint Element MUST use a CueCodecState Element.

*CuePoint Elements SHOULD be numerically sorted in storage order by the value of the CueTime Element.

23. Matroska Streaming

In Matroska, there are two kinds of streaming: file access and livestreaming.

23.1. File Access

File access can simply be reading a file located on your computer, but also includes accessing a file from an HTTP (web) server or CIFS (Windows share) server. These protocols are usually safe from
reading errors and seeking in the stream is possible. However, when a file is stored far away or on a slow server, seeking can be an expensive operation and should be avoided. The guidelines in Section 25, when followed, help reduce the number of seeking operations for regular playback and also have the playback start quickly without a lot of data needed to read first (like a Cues Element, Attachment Element or SeekHead Element).

Matroska, having a small overhead, is well suited for storing music/videos on file servers without a big impact on the bandwidth used. Matroska does not require the index to be loaded before playing, which allows playback to start very quickly. The index can be loaded only when seeking is requested the first time.

23.2. Livestreaming

Livestreaming is the equivalent of television broadcasting on the internet. There are 2 families of servers for livestreaming: RTP/RTSP and HTTP. Matroska is not meant to be used over RTP. RTP already has timing and channel mechanisms that would be wasted if doubled in Matroska. Additionally, having the same information at the RTP and Matroska level would be a source of confusion if they do not match. Livestreaming of Matroska over 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 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 and not 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. So 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, 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) element 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 contains no more than 5 seconds or 5 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 ordering of Elements are better than others. Here are few optimum layouts for different use case:

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. For players that want to seek in the file they 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. SeekHead and Cues are useless for example. All elements other than the Clusters MUST be placed before the Clusters.

*Info
*Tracks
*Attachments (rare)
*Tags
*Clusters
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 is 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, 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 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

This document creates a new IANA registry called the "Matroska Element IDs" registry.

To register a new Element ID in this registry, one needs an Element ID, a Change Controller (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-long Element IDs are possible only if declared in the EBML header.

Element IDs are described in Section 5 of [RFC8794] with errata 7189 and 7191.

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 Elements IDs registry are similar to the ones found in the EBML Element IDs registry defined in Section 17.1 of [RFC8794].

EBML 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 the 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 Section 28 have an assigned Matroska Element ID for historical reasons. These elements are not in use and SHOULD NOT be reused unless there is no other IDs available with the desired size. Such IDs are considered as reclaimed to the IANA registry as they could be used for other things in the future.

Matroska Element IDs Values found in this document are assigned as initial values as follows:

<table>
<thead>
<tr>
<th>Element ID</th>
<th>Element Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80</td>
<td>ChapterDisplay</td>
<td>Described in Section 5.1.7.1.4.9</td>
</tr>
<tr>
<td>0x83</td>
<td>TrackType</td>
<td>Described in Section 5.1.4.1.3</td>
</tr>
<tr>
<td>0x85</td>
<td>ChapString</td>
<td>Described in Section 5.1.7.1.4.10</td>
</tr>
<tr>
<td>0x86</td>
<td>CodecID</td>
<td>Described in Section 5.1.4.1.21</td>
</tr>
<tr>
<td>Element ID</td>
<td>Element Name</td>
<td>Reference</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>0x88</td>
<td>FlagDefault</td>
<td>Described in Section 5.1.4.1.5</td>
</tr>
<tr>
<td>0x8E</td>
<td>Slices</td>
<td>reclaimed (Section 28.5)</td>
</tr>
<tr>
<td>0x91</td>
<td>ChapterTimeStart</td>
<td>Described in Section 5.1.7.1.4.3</td>
</tr>
<tr>
<td>0x92</td>
<td>ChapterTimeEnd</td>
<td>Described in Section 5.1.7.1.4.4</td>
</tr>
<tr>
<td>0x96</td>
<td>CueRefTime</td>
<td>Described in Section 5.1.5.1.2.8</td>
</tr>
<tr>
<td>0x97</td>
<td>CueRefCluster</td>
<td>reclaimed (Section 28.37)</td>
</tr>
<tr>
<td>0x98</td>
<td>ChapterFlagHidden</td>
<td>Described in Section 5.1.7.1.4.5</td>
</tr>
<tr>
<td>0x9A</td>
<td>FlagInterlaced</td>
<td>Described in Section 5.1.4.1.28.1</td>
</tr>
<tr>
<td>0x9B</td>
<td>BlockDuration</td>
<td>Described in Section 5.1.3.5.3</td>
</tr>
<tr>
<td>0x9C</td>
<td>FlagLacing</td>
<td>Described in Section 5.1.4.1.12</td>
</tr>
<tr>
<td>0x9D</td>
<td>FieldOrder</td>
<td>Described in Section 5.1.4.1.28.2</td>
</tr>
<tr>
<td>0x9F</td>
<td>Channels</td>
<td>Described in Section 5.1.4.1.29.3</td>
</tr>
<tr>
<td>0xA0</td>
<td>BlockGroup</td>
<td>Described in Section 5.1.3.5</td>
</tr>
<tr>
<td>0xA1</td>
<td>Block</td>
<td>Described in Section 5.1.3.5.1</td>
</tr>
<tr>
<td>0xA2</td>
<td>BlockVirtual</td>
<td>reclaimed (Section 28.3)</td>
</tr>
<tr>
<td>0xA3</td>
<td>SimpleBlock</td>
<td>Described in Section 5.1.3.4</td>
</tr>
<tr>
<td>0xA4</td>
<td>CodecState</td>
<td>Described in Section 5.1.3.5.6</td>
</tr>
<tr>
<td>0xA5</td>
<td>BlockAdditional</td>
<td>Described in Section 5.1.3.5.2.2</td>
</tr>
<tr>
<td>0xA6</td>
<td>BlockMore</td>
<td>Described in Section 5.1.3.5.2.1</td>
</tr>
<tr>
<td>0xA7</td>
<td>Position</td>
<td>Described in Section 5.1.3.2</td>
</tr>
<tr>
<td>0xAA</td>
<td>CodecDecodeAll</td>
<td>reclaimed (Section 28.22)</td>
</tr>
<tr>
<td>0xAB</td>
<td>PrevSize</td>
<td>Described in Section 5.1.3.3</td>
</tr>
<tr>
<td>0xAE</td>
<td>TrackEntry</td>
<td>Described in Section 5.1.4.1</td>
</tr>
<tr>
<td>0xAF</td>
<td>EncryptedBlock</td>
<td>reclaimed (Section 28.15)</td>
</tr>
<tr>
<td>0xB0</td>
<td>PixelWidth</td>
<td>Described in Section 5.1.4.1.28.6</td>
</tr>
<tr>
<td>0xB2</td>
<td>CueDuration</td>
<td>Described in Section 5.1.5.1.2.4</td>
</tr>
<tr>
<td>0xB3</td>
<td>CueTime</td>
<td>Described in Section 5.1.5.1.1</td>
</tr>
<tr>
<td>0xB5</td>
<td>SamplingFrequency</td>
<td>Described in Section 5.1.4.1.29.1</td>
</tr>
<tr>
<td>0xB6</td>
<td>ChapterAtom</td>
<td>Described in Section 5.1.7.1.4</td>
</tr>
<tr>
<td>Element ID</td>
<td>Element Name</td>
<td>Reference</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>0xB7</td>
<td>CueTrackPositions</td>
<td>Described in Section 5.1.5.1.2</td>
</tr>
<tr>
<td>0xB9</td>
<td>FlagEnabled</td>
<td>Described in Section 5.1.4.1.4</td>
</tr>
<tr>
<td>0xBA</td>
<td>PixelHeight</td>
<td>Described in Section 5.1.4.1.28.7</td>
</tr>
<tr>
<td>0xBB</td>
<td>CuePoint</td>
<td>Described in Section 5.1.5.1.1</td>
</tr>
<tr>
<td>0xC0</td>
<td>TrickTrackUID</td>
<td>Reclaimed (Section 28.28)</td>
</tr>
<tr>
<td>0xC1</td>
<td>TrickTrackSegmentUID</td>
<td>Reclaimed (Section 28.29)</td>
</tr>
<tr>
<td>0xC4</td>
<td>TrickMasterTrackSegmentUID</td>
<td>Reclaimed (Section 28.32)</td>
</tr>
<tr>
<td>0xC6</td>
<td>TrickTrackFlag</td>
<td>Reclaimed (Section 28.30)</td>
</tr>
<tr>
<td>0xC7</td>
<td>TrickMasterTrackUID</td>
<td>Reclaimed (Section 28.31)</td>
</tr>
<tr>
<td>0xC8</td>
<td>ReferenceFrame</td>
<td>Reclaimed (Section 28.12)</td>
</tr>
<tr>
<td>0xC9</td>
<td>ReferenceOffset</td>
<td>Reclaimed (Section 28.13)</td>
</tr>
<tr>
<td>0xCA</td>
<td>ReferenceTimestamp</td>
<td>Reclaimed (Section 28.14)</td>
</tr>
<tr>
<td>0xCB</td>
<td>BlockAdditionID</td>
<td>Reclaimed (Section 28.9)</td>
</tr>
<tr>
<td>0xCC</td>
<td>LaceNumber</td>
<td>Reclaimed (Section 28.7)</td>
</tr>
<tr>
<td>0xCD</td>
<td>FrameNumber</td>
<td>Reclaimed (Section 28.8)</td>
</tr>
<tr>
<td>0xCE</td>
<td>Delay</td>
<td>Reclaimed (Section 28.10)</td>
</tr>
<tr>
<td>0xCF</td>
<td>SliceDuration</td>
<td>Reclaimed (Section 28.11)</td>
</tr>
<tr>
<td>0xD7</td>
<td>TrackNumber</td>
<td>Described in Section 5.1.4.1.1</td>
</tr>
<tr>
<td>0xDB</td>
<td>CueReference</td>
<td>Described in Section 5.1.5.1.2.7</td>
</tr>
<tr>
<td>0xE0</td>
<td>Video</td>
<td>Described in Section 5.1.4.1.28</td>
</tr>
<tr>
<td>0xE1</td>
<td>Audio</td>
<td>Described in Section 5.1.4.1.29</td>
</tr>
<tr>
<td>0xE2</td>
<td>TrackOperation</td>
<td>Described in Section 5.1.4.1.30</td>
</tr>
<tr>
<td>0xE3</td>
<td>TrackCombinePlanes</td>
<td>Described in Section 5.1.4.1.30.1</td>
</tr>
<tr>
<td>0xE4</td>
<td>TrackPlane</td>
<td>Described in Section 5.1.4.1.30.2</td>
</tr>
<tr>
<td>0xE5</td>
<td>TrackPlaneUID</td>
<td>Described in Section 5.1.4.1.30.3</td>
</tr>
<tr>
<td>0xE6</td>
<td>TrackPlaneType</td>
<td>Described in Section 5.1.4.1.30.4</td>
</tr>
<tr>
<td>0xE7</td>
<td>Timestamp</td>
<td>Described in Section 5.1.3.1</td>
</tr>
<tr>
<td>0xE8</td>
<td>TimeSlice</td>
<td>Reclaimed (Section 28.6)</td>
</tr>
<tr>
<td>0xE9</td>
<td>TrackJoinBlocks</td>
<td>Described in Section 5.1.4.1.30.5</td>
</tr>
<tr>
<td>0xEA</td>
<td>CueCodecState</td>
<td>Described in Section 5.1.5.1.2.6</td>
</tr>
<tr>
<td>0xEB</td>
<td>CueRefCodecState</td>
<td>Reclaimed (Section 28.39)</td>
</tr>
<tr>
<td>0xED</td>
<td>TrackJoinUID</td>
<td>Described in Section 5.1.4.1.30.6</td>
</tr>
<tr>
<td>0xEE</td>
<td>BlockAddID</td>
<td></td>
</tr>
<tr>
<td>Element ID</td>
<td>Element Name</td>
<td>Reference</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>0xF0</td>
<td>CueRelativePosition</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.3.5.2.3</td>
</tr>
<tr>
<td>0xF1</td>
<td>CueClusterPosition</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.5.1.2.2</td>
</tr>
<tr>
<td>0xF7</td>
<td>CueTrack</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.5.1.2.1</td>
</tr>
<tr>
<td>0xFA</td>
<td>ReferencePriority</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.3.5.4</td>
</tr>
<tr>
<td>0xFB</td>
<td>ReferenceBlock</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.3.5.5</td>
</tr>
<tr>
<td>0xFD</td>
<td>ReferenceVirtual</td>
<td>Reclaimed (Section 28.4)</td>
</tr>
<tr>
<td>0x41A4</td>
<td>BlockAddIDName</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.4.1.17.2</td>
</tr>
<tr>
<td>0x41E4</td>
<td>BlockAdditionMapping</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.4.1.17</td>
</tr>
<tr>
<td>0x41E7</td>
<td>BlockAddIDType</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.4.1.17.3</td>
</tr>
<tr>
<td>0x41ED</td>
<td>BlockAddIDEExtraData</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.4.1.17.4</td>
</tr>
<tr>
<td>0x41F0</td>
<td>BlockAddIDValue</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.4.1.17.1</td>
</tr>
<tr>
<td>0x4254</td>
<td>ContentCompAlgo</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.4.1.31.6</td>
</tr>
<tr>
<td>0x4255</td>
<td>ContentCompSettings</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.4.1.31.7</td>
</tr>
<tr>
<td>0x437C</td>
<td>ChapLanguage</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.7.1.4.11</td>
</tr>
<tr>
<td>0x437D</td>
<td>ChapLanguageBCP47</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.7.1.4.12</td>
</tr>
<tr>
<td>0x437E</td>
<td>ChapCountry</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.7.1.4.13</td>
</tr>
<tr>
<td>0x4444</td>
<td>SegmentFamily</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.2.7</td>
</tr>
<tr>
<td>0x4461</td>
<td>DateUTC</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.2.11</td>
</tr>
<tr>
<td>0x447A</td>
<td>TagLanguage</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.8.1.2.2</td>
</tr>
<tr>
<td>0x447B</td>
<td>TagLanguageBCP47</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.8.1.2.3</td>
</tr>
<tr>
<td>0x4484</td>
<td>TagDefault</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.8.1.2.4</td>
</tr>
<tr>
<td>0x4485</td>
<td>TagBinary</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.8.1.2.6</td>
</tr>
<tr>
<td>0x4487</td>
<td>TagString</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.8.1.2.5</td>
</tr>
<tr>
<td>0x4489</td>
<td>Duration</td>
<td>Described in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 5.1.2.10</td>
</tr>
<tr>
<td>Element ID</td>
<td>Element Name</td>
<td>Reference</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>0x44B4</td>
<td>TagDefaultBogus</td>
<td>Reclaimed (Section 28.43)</td>
</tr>
<tr>
<td>0x450D</td>
<td>ChapProcessPrivate</td>
<td>Section 5.1.7.1.4.16</td>
</tr>
<tr>
<td>0x45A3</td>
<td>TagName</td>
<td>Section 5.1.8.1.2.1</td>
</tr>
<tr>
<td>0x45B9</td>
<td>EditionEntry</td>
<td>Section 5.1.7.1</td>
</tr>
<tr>
<td>0x45BC</td>
<td>EditionUID</td>
<td>Section 5.1.7.1.1</td>
</tr>
<tr>
<td>0x45DB</td>
<td>EditionFlagDefault</td>
<td>Section 5.1.7.1.2</td>
</tr>
<tr>
<td>0x45DD</td>
<td>EditionFlagOrdered</td>
<td>Section 5.1.7.1.3</td>
</tr>
<tr>
<td>0x465C</td>
<td>FileData</td>
<td>Section 5.1.6.1.4</td>
</tr>
<tr>
<td>0x4660</td>
<td>FileMediaType</td>
<td>Section 5.1.6.1.3</td>
</tr>
<tr>
<td>0x4661</td>
<td>FileUsedStartTime</td>
<td>Reclaimed (Section 28.41)</td>
</tr>
<tr>
<td>0x4662</td>
<td>FileUsedEndTime</td>
<td>Reclaimed (Section 28.42)</td>
</tr>
<tr>
<td>0x466E</td>
<td>FileName</td>
<td>Section 5.1.6.1.2</td>
</tr>
<tr>
<td>0x4675</td>
<td>FileReferral</td>
<td>Reclaimed (Section 28.40)</td>
</tr>
<tr>
<td>0x467E</td>
<td>FileDescription</td>
<td>Section 5.1.6.1.1</td>
</tr>
<tr>
<td>0x46AE</td>
<td>FileUID</td>
<td>Section 5.1.6.1.5</td>
</tr>
<tr>
<td>0x47E1</td>
<td>ContentEncAlgo</td>
<td>Section 5.1.4.1.31.9</td>
</tr>
<tr>
<td>0x47E2</td>
<td>ContentEncKeyID</td>
<td>Section 5.1.4.1.31.10</td>
</tr>
<tr>
<td>0x47E3</td>
<td>ContentSignature</td>
<td>Reclaimed (Section 28.33)</td>
</tr>
<tr>
<td>0x47E4</td>
<td>ContentSigKeyID</td>
<td>Reclaimed (Section 28.34)</td>
</tr>
<tr>
<td>0x47E5</td>
<td>ContentSigAlgo</td>
<td>Reclaimed (Section 28.35)</td>
</tr>
<tr>
<td>0x47E6</td>
<td>ContentSigHashAlgo</td>
<td>Reclaimed (Section 28.36)</td>
</tr>
<tr>
<td>0x47E7</td>
<td>ContentEncAESSettings</td>
<td>Section 5.1.4.1.31.11</td>
</tr>
<tr>
<td>0x47E8</td>
<td>AESSettingsCipherMode</td>
<td>Section 5.1.4.1.31.12</td>
</tr>
<tr>
<td>0x4D80</td>
<td>MuxingApp</td>
<td>Section 5.1.2.13</td>
</tr>
<tr>
<td>0x4DBB</td>
<td>Seek</td>
<td>Section 5.1.1.1</td>
</tr>
<tr>
<td>0x5031</td>
<td>ContentEncodingOrder</td>
<td>Section 5.1.4.1.31.2</td>
</tr>
<tr>
<td>0x5032</td>
<td>ContentEncodingScope</td>
<td>Section 5.1.4.1.31.3</td>
</tr>
<tr>
<td>0x5033</td>
<td>ContentEncodingType</td>
<td>Section 5.1.4.1.31.4</td>
</tr>
<tr>
<td>0x5034</td>
<td>ContentCompression</td>
<td>Section 5.1.4.1.31.5</td>
</tr>
<tr>
<td>Element ID</td>
<td>Element Name</td>
<td>Reference</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>0x5035</td>
<td>ContentEncryption</td>
<td>Described in Section 5.1.4.1.31.8</td>
</tr>
<tr>
<td>0x535F</td>
<td>CueRefNumber</td>
<td>Reclaimed (Section 28.38)</td>
</tr>
<tr>
<td>0x536E</td>
<td>Name</td>
<td>Described in Section 5.1.4.1.18</td>
</tr>
<tr>
<td>0x5378</td>
<td>CueBlockNumber</td>
<td>Described in Section 5.1.5.1.2.5</td>
</tr>
<tr>
<td>0x537F</td>
<td>TrackOffset</td>
<td>Reclaimed (Section 28.18)</td>
</tr>
<tr>
<td>0x53AB</td>
<td>SeekID</td>
<td>Described in Section 5.1.1.1.1</td>
</tr>
<tr>
<td>0x53AC</td>
<td>SeekPosition</td>
<td>Described in Section 5.1.1.1.2</td>
</tr>
<tr>
<td>0x53B8</td>
<td>StereoMode</td>
<td>Described in Section 5.1.4.1.28.3</td>
</tr>
<tr>
<td>0x53B9</td>
<td>OldStereoMode</td>
<td>Described in Section 5.1.4.1.28.5</td>
</tr>
<tr>
<td>0x53C0</td>
<td>AlphaMode</td>
<td>Described in Section 5.1.4.1.28.4</td>
</tr>
<tr>
<td>0x54AA</td>
<td>PixelCropBottom</td>
<td>Described in Section 5.1.4.1.28.8</td>
</tr>
<tr>
<td>0x54B0</td>
<td>DisplayWidth</td>
<td>Described in Section 5.1.4.1.28.12</td>
</tr>
<tr>
<td>0x54B2</td>
<td>DisplayUnit</td>
<td>Described in Section 5.1.4.1.28.14</td>
</tr>
<tr>
<td>0x54B3</td>
<td>AspectRatioType</td>
<td>Reclaimed (Section 28.24)</td>
</tr>
<tr>
<td>0x54BA</td>
<td>DisplayHeight</td>
<td>Described in Section 5.1.4.1.28.13</td>
</tr>
<tr>
<td>0x54BB</td>
<td>PixelCropTop</td>
<td>Described in Section 5.1.4.1.28.9</td>
</tr>
<tr>
<td>0x54CC</td>
<td>PixelCropLeft</td>
<td>Described in Section 5.1.4.1.28.10</td>
</tr>
<tr>
<td>0x54DD</td>
<td>PixelCropRight</td>
<td>Described in Section 5.1.4.1.28.11</td>
</tr>
<tr>
<td>0x55AA</td>
<td>FlagForced</td>
<td>Described in Section 5.1.4.1.6</td>
</tr>
<tr>
<td>0x55AB</td>
<td>FlagHearingImpaired</td>
<td>Described in Section 5.1.4.1.7</td>
</tr>
<tr>
<td>0x55AC</td>
<td>FlagVisualImpaired</td>
<td>Described in Section 5.1.4.1.8</td>
</tr>
<tr>
<td>0x55AD</td>
<td>FlagTextDescriptions</td>
<td>Described in Section 5.1.4.1.9</td>
</tr>
<tr>
<td>0x55AE</td>
<td>FlagOriginal</td>
<td>Described in Section 5.1.4.1.10</td>
</tr>
<tr>
<td>0x55AF</td>
<td>FlagCommentary</td>
<td>Described in Section 5.1.4.1.11</td>
</tr>
<tr>
<td>0x55B0</td>
<td>Colour</td>
<td>Described in Section 5.1.4.1.28.16</td>
</tr>
<tr>
<td>0x55B1</td>
<td>MatrixCoefficients</td>
<td></td>
</tr>
<tr>
<td>Element ID</td>
<td>Element Name</td>
<td>Reference</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>0x55B2</td>
<td>BitsPerChannel</td>
<td>Described in 5.1.4.1.28.17</td>
</tr>
<tr>
<td>0x55B3</td>
<td>ChromaSubsamplingHorz</td>
<td>Described in 5.1.4.1.28.19</td>
</tr>
<tr>
<td>0x55B4</td>
<td>ChromaSubsamplingVert</td>
<td>Described in 5.1.4.1.28.20</td>
</tr>
<tr>
<td>0x55B5</td>
<td>CbSubsamplingHorz</td>
<td>Described in 5.1.4.1.28.21</td>
</tr>
<tr>
<td>0x55B6</td>
<td>CbSubsamplingVert</td>
<td>Described in 5.1.4.1.28.22</td>
</tr>
<tr>
<td>0x55B7</td>
<td>ChromaSitingHorz</td>
<td>Described in 5.1.4.1.28.23</td>
</tr>
<tr>
<td>0x55B8</td>
<td>ChromaSitingVert</td>
<td>Described in 5.1.4.1.28.24</td>
</tr>
<tr>
<td>0x55B9</td>
<td>Range</td>
<td>Described in 5.1.4.1.28.25</td>
</tr>
<tr>
<td>0x55BA</td>
<td>TransferCharacteristics</td>
<td>Described in 5.1.4.1.28.26</td>
</tr>
<tr>
<td>0x55BB</td>
<td>Primaries</td>
<td>Described in 5.1.4.1.28.27</td>
</tr>
<tr>
<td>0x55BC</td>
<td>MaxCLL</td>
<td>Described in 5.1.4.1.28.28</td>
</tr>
<tr>
<td>0x55BD</td>
<td>MaxFALL</td>
<td>Described in 5.1.4.1.28.29</td>
</tr>
<tr>
<td>0x55D0</td>
<td>MasteringMetadata</td>
<td>Described in 5.1.4.1.28.30</td>
</tr>
<tr>
<td>0x55D1</td>
<td>PrimaryRChromaticityX</td>
<td>Described in 5.1.4.1.28.31</td>
</tr>
<tr>
<td>0x55D2</td>
<td>PrimaryRChromaticityY</td>
<td>Described in 5.1.4.1.28.32</td>
</tr>
<tr>
<td>0x55D3</td>
<td>PrimaryGChromaticityX</td>
<td>Described in 5.1.4.1.28.33</td>
</tr>
<tr>
<td>0x55D4</td>
<td>PrimaryGChromaticityY</td>
<td>Described in 5.1.4.1.28.34</td>
</tr>
<tr>
<td>0x55D5</td>
<td>PrimaryBChromaticityX</td>
<td>Described in 5.1.4.1.28.35</td>
</tr>
<tr>
<td>0x55D6</td>
<td>PrimaryBChromaticityY</td>
<td>Described in 5.1.4.1.28.36</td>
</tr>
<tr>
<td>0x55D7</td>
<td>WhitePointChromaticityX</td>
<td>Described in 5.1.4.1.28.37</td>
</tr>
<tr>
<td>0x55D8</td>
<td>WhitePointChromaticityY</td>
<td>Described in 5.1.4.1.28.38</td>
</tr>
<tr>
<td>0x55D9</td>
<td>LuminanceMax</td>
<td>Described in 5.1.4.1.28.39</td>
</tr>
<tr>
<td>0x55DA</td>
<td>LuminanceMin</td>
<td>Described in 5.1.4.1.28.40</td>
</tr>
<tr>
<td>Element ID</td>
<td>Element Name</td>
<td>Reference</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>0x55EE</td>
<td>MaxBlockAdditionID</td>
<td>Described in Section 5.1.4.1.16</td>
</tr>
<tr>
<td>0x5654</td>
<td>ChapterStringUID</td>
<td>Described in Section 5.1.7.1.4.2</td>
</tr>
<tr>
<td>0x56AA</td>
<td>CodecDelay</td>
<td>Described in Section 5.1.4.1.25</td>
</tr>
<tr>
<td>0x56BB</td>
<td>SeekPreRoll</td>
<td>Described in Section 5.1.4.1.26</td>
</tr>
<tr>
<td>0x5741</td>
<td>WritingApp</td>
<td>Described in Section 5.1.2.14</td>
</tr>
<tr>
<td>0x5854</td>
<td>SilentTracks</td>
<td>Reclaimed (Section 28.1)</td>
</tr>
<tr>
<td>0x58D7</td>
<td>SilentTrackNumber</td>
<td>Reclaimed (Section 28.2)</td>
</tr>
<tr>
<td>0x61A7</td>
<td>AttachedFile</td>
<td>Described in Section 5.1.6.1</td>
</tr>
<tr>
<td>0x6240</td>
<td>ContentEncoding</td>
<td>Described in Section 5.1.4.1.31.1</td>
</tr>
<tr>
<td>0x6264</td>
<td>BitDepth</td>
<td>Described in Section 5.1.4.1.29.4</td>
</tr>
<tr>
<td>0x63A2</td>
<td>CodecPrivate</td>
<td>Described in Section 5.1.4.1.22</td>
</tr>
<tr>
<td>0x63C0</td>
<td>Targets</td>
<td>Described in Section 5.1.8.1.1</td>
</tr>
<tr>
<td>0x63C3</td>
<td>ChapterPhysicalEquiv</td>
<td>Described in Section 5.1.7.1.4.8</td>
</tr>
<tr>
<td>0x63C4</td>
<td>TagChapterUID</td>
<td>Described in Section 5.1.8.1.1.5</td>
</tr>
<tr>
<td>0x63C5</td>
<td>TagTrackUID</td>
<td>Described in Section 5.1.8.1.1.3</td>
</tr>
<tr>
<td>0x63C6</td>
<td>TagAttachmentUID</td>
<td>Described in Section 5.1.8.1.1.6</td>
</tr>
<tr>
<td>0x63C9</td>
<td>TagEditionUID</td>
<td>Described in Section 5.1.8.1.1.4</td>
</tr>
<tr>
<td>0x63CA</td>
<td>TargetType</td>
<td>Described in Section 5.1.8.1.1.2</td>
</tr>
<tr>
<td>0x6624</td>
<td>TrackTranslate</td>
<td>Described in Section 5.1.4.1.27</td>
</tr>
<tr>
<td>0x66A5</td>
<td>TrackTranslateTrackID</td>
<td>Described in Section 5.1.4.1.27.1</td>
</tr>
<tr>
<td>0x66BF</td>
<td>TrackTranslateCodec</td>
<td>Described in Section 5.1.4.1.27.2</td>
</tr>
<tr>
<td>0x66FC</td>
<td>TrackTranslateEditionUID</td>
<td>Described in Section 5.1.4.1.27.3</td>
</tr>
<tr>
<td>0x67C8</td>
<td>SimpleTag</td>
<td>Described in Section 5.1.8.1.2</td>
</tr>
<tr>
<td>0x68CA</td>
<td>TargetTypeValue</td>
<td>Described in Section 5.1.8.1.1.1</td>
</tr>
<tr>
<td>0x6911</td>
<td>ChapProcessCommand</td>
<td>Described in Section 5.1.7.1.4.17</td>
</tr>
<tr>
<td>0x6922</td>
<td>ChapProcessTime</td>
<td></td>
</tr>
<tr>
<td>Element ID</td>
<td>Element Name</td>
<td>Reference</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>0x6924</td>
<td>ChapterTranslate</td>
<td>Described in Section 5.1.7.1.4.18</td>
</tr>
<tr>
<td>0x6933</td>
<td>ChapProcessData</td>
<td>Described in Section 5.1.7.1.4.19</td>
</tr>
<tr>
<td>0x6944</td>
<td>ChapProcess</td>
<td>Described in Section 5.1.7.1.4.14</td>
</tr>
<tr>
<td>0x6955</td>
<td>ChapProcessCodecID</td>
<td>Described in Section 5.1.7.1.4.15</td>
</tr>
<tr>
<td>0x69A5</td>
<td>ChapterTranslateID</td>
<td>Described in Section 5.1.2.8.1</td>
</tr>
<tr>
<td>0x69BF</td>
<td>ChapterTranslateCodec</td>
<td>Described in Section 5.1.2.8.2</td>
</tr>
<tr>
<td>0x69FC</td>
<td>ChapterTranslateEditionUID</td>
<td>Described in Section 5.1.2.8.3</td>
</tr>
<tr>
<td>0x6D80</td>
<td>ContentEncodings</td>
<td>Described in Section 5.1.4.1.31</td>
</tr>
<tr>
<td>0x6DE7</td>
<td>MinCache</td>
<td>Reclaimed (Section 28.16)</td>
</tr>
<tr>
<td>0x6DF8</td>
<td>MaxCache</td>
<td>Reclaimed (Section 28.17)</td>
</tr>
<tr>
<td>0x6E67</td>
<td>ChapterSegmentUUID</td>
<td>Described in Section 5.1.7.1.4.6</td>
</tr>
<tr>
<td>0x6EBC</td>
<td>ChapterSegmentEditionUID</td>
<td>Described in Section 5.1.7.1.4.7</td>
</tr>
<tr>
<td>0x6FAB</td>
<td>TrackOverlay</td>
<td>Reclaimed (Section 28.23)</td>
</tr>
<tr>
<td>0x7373</td>
<td>Tag</td>
<td>Described in Section 5.1.8.1</td>
</tr>
<tr>
<td>0x7384</td>
<td>SegmentFilename</td>
<td>Described in Section 5.1.2.2</td>
</tr>
<tr>
<td>0x73A4</td>
<td>SegmentUUID</td>
<td>Described in Section 5.1.2.1</td>
</tr>
<tr>
<td>0x73C4</td>
<td>ChapterUID</td>
<td>Described in Section 5.1.7.1.4.1</td>
</tr>
<tr>
<td>0x73C5</td>
<td>TrackUID</td>
<td>Described in Section 5.1.4.1.2</td>
</tr>
<tr>
<td>0x7446</td>
<td>AttachmentLink</td>
<td>Described in Section 5.1.4.1.24</td>
</tr>
<tr>
<td>0x75A1</td>
<td>BlockAdditions</td>
<td>Described in Section 5.1.3.5.2</td>
</tr>
<tr>
<td>0x75A2</td>
<td>DiscardPadding</td>
<td>Described in Section 5.1.3.5.7</td>
</tr>
<tr>
<td>0x7670</td>
<td>Projection</td>
<td>Described in Section 5.1.4.1.28.41</td>
</tr>
<tr>
<td>0x7671</td>
<td>ProjectionType</td>
<td>Described in Section 5.1.4.1.28.42</td>
</tr>
<tr>
<td>0x7672</td>
<td>ProjectionPrivate</td>
<td>Described in Section 5.1.4.1.28.43</td>
</tr>
<tr>
<td>0x7673</td>
<td>ProjectionPoseYaw</td>
<td>Described in Section 5.1.4.1.28.44</td>
</tr>
<tr>
<td>0x7674</td>
<td>ProjectionPosePitch</td>
<td>Described in Section 5.1.4.1.28.45</td>
</tr>
<tr>
<td>0x7675</td>
<td>ProjectionPoseRoll</td>
<td></td>
</tr>
<tr>
<td>Element ID</td>
<td>Element Name</td>
<td>Reference</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>0x78B5</td>
<td>OutputSamplingFrequency</td>
<td>Described in Section 5.1.4.1.28.46</td>
</tr>
<tr>
<td>0x7BA9</td>
<td>Title</td>
<td>Described in Section 5.1.2.12</td>
</tr>
<tr>
<td>0x7D7B</td>
<td>ChannelPositions</td>
<td>Reclaimed (Section 28.27)</td>
</tr>
<tr>
<td>0x22B59C</td>
<td>Language</td>
<td>Described in Section 5.1.4.1.19</td>
</tr>
<tr>
<td>0x22B59D</td>
<td>LanguageBCP47</td>
<td>Described in Section 5.1.4.1.20</td>
</tr>
<tr>
<td>0x23314F</td>
<td>TrackTimestampScale</td>
<td>Described in Section 5.1.4.1.15</td>
</tr>
<tr>
<td>0x234E7A</td>
<td>DefaultDecodedFieldDuration</td>
<td>Described in Section 5.1.4.1.14</td>
</tr>
<tr>
<td>0x2383E3</td>
<td>FrameRate</td>
<td>Reclaimed (Section 28.26)</td>
</tr>
<tr>
<td>0x23E838</td>
<td>DefaultDuration</td>
<td>Described in Section 5.1.4.1.13</td>
</tr>
<tr>
<td>0x258688</td>
<td>CodecName</td>
<td>Described in Section 5.1.4.1.23</td>
</tr>
<tr>
<td>0x26B240</td>
<td>CodecDownloadURL</td>
<td>Reclaimed (Section 28.21)</td>
</tr>
<tr>
<td>0x2AD7B1</td>
<td>TimestampScale</td>
<td>Described in Section 5.1.2.9</td>
</tr>
<tr>
<td>0x2EB524</td>
<td>UncompressedFourCC</td>
<td>Described in Section 5.1.4.1.28.15</td>
</tr>
<tr>
<td>0x2FB523</td>
<td>GammaValue</td>
<td>Reclaimed (Section 28.25)</td>
</tr>
<tr>
<td>0x3A9697</td>
<td>CodecSettings</td>
<td>Reclaimed (Section 28.19)</td>
</tr>
<tr>
<td>0x3B4040</td>
<td>CodecInfoURL</td>
<td>Reclaimed (Section 28.20)</td>
</tr>
<tr>
<td>0x3C83AB</td>
<td>PrevFilename</td>
<td>Described in Section 5.1.2.4</td>
</tr>
<tr>
<td>0x3CB923</td>
<td>PrevUUID</td>
<td>Described in Section 5.1.2.3</td>
</tr>
<tr>
<td>0x3E83BB</td>
<td>NextFilename</td>
<td>Described in Section 5.1.2.6</td>
</tr>
<tr>
<td>0x3EB923</td>
<td>NextUUID</td>
<td>Described in Section 5.1.2.5</td>
</tr>
<tr>
<td>0x1043A770</td>
<td>Chapters</td>
<td>Described in Section 5.1.7</td>
</tr>
<tr>
<td>0x114D9B74</td>
<td>SeekHead</td>
<td>Described in Section 5.1.1</td>
</tr>
<tr>
<td>0x1254C367</td>
<td>Tags</td>
<td>Described in Section 5.1.8</td>
</tr>
<tr>
<td>0x1549A966</td>
<td>Info</td>
<td>Described in Section 5.1.2</td>
</tr>
<tr>
<td>0x1654AE6B</td>
<td>Tracks</td>
<td>Described in Section 5.1.4</td>
</tr>
<tr>
<td>0x18538067</td>
<td>Segment</td>
<td>Described in Section 5.1</td>
</tr>
<tr>
<td>0x1941A469</td>
<td>Attachments</td>
<td>Described in Section 5.1.6</td>
</tr>
<tr>
<td>0x1C538B6B</td>
<td>Cues</td>
<td>Described in Section 5.1.5</td>
</tr>
<tr>
<td>0x1F43B675</td>
<td>Cluster</td>
<td>Described in Section 5.1.3</td>
</tr>
</tbody>
</table>

Table 55: IDs and Names for Matroska Element IDs assigned by this document
27.2. Chapter Codec IDs Registry

This document creates a new IANA 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].

ChapProcessCodecID values of "0" and "1" are RESERVED to the IETF for future use.

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 a "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 [RFC6838] format instead.

Please register three media types, the [RFC6838] templates are below:

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 this document and RFC8794
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: THISRFC
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
Provisional registration? (standards tree only): No

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 this document and RFC8794
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: THISRFC
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
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 this document and RFC8794
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: THISRFC
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
Provisional registration? (standards tree only): No


As Matroska evolved since 2002 many parts that were considered for use in the format were never used and often incorrectly designed. Many of the elements that were then defined are not found in any known files but were part of public specs. DivX also had a few custom elements that were designed for custom features.

We list these elements that have a known ID that SHOULD NOT be reused to avoid colliding with existing files. They 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.

28.1. SilentTracks Element

type / id: master / 0x5854
path: \Segment\Cluster\SilentTracks
documentation: The list of tracks that are not used in that part of the stream. It is useful when using overlay tracks on seeking or to decide what track to use.

28.2. SilentTrackNumber Element

type / id: uinteger / 0x58D7
path: \Segment\Cluster\SilentTracks\SilentTrackNumber
documentation: One of the track number that are not used from now on in the stream. It could change later if not specified as silent in a further Cluster.

28.3. 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.

28.4. 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.

28.5. Slices Element

type / id: master / 0x8E
path: \Segment\Cluster\BlockGroup\Slices
documentation: Contains slices description.

28.6. 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.

28.7. 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.

28.8. 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 (allow you to generate many frames from the same Block/Frame).

28.9. 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).

28.10. 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.

28.11. 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.

28.12. ReferenceFrame Element

type / id: master / 0xC8
path: \Segment\Cluster\BlockGroup\ReferenceFrame
documentation: Contains information about the last reference frame. See [DivXTrickTrack].

28.13. 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].

28.15. 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 (encrypt and/or signed).

28.16. MinCache Element

type / id: integer / 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.

28.17. MaxCache Element

type / id: integer / 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.

28.18. 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.

28.19. CodecSettings Element

type / id: utf-8 / 0x3A9697
path: \Segment\Tracks\TrackEntry\CodecSettings
documentation: A string describing the encoding setting used.

28.20. CodecInfoURL Element

type / id: string / 0x3B4040
path: \Segment\Tracks\TrackEntry\CodecInfoURL
documentation: A URL to find information about the codec used.

28.21. CodecDownloadURL Element
28.22. CodecDecodeAll Element

type / id:  uinteger / 0xAA
path:  \Segment\Tracks\TrackEntry\CodecDecodeAll
documentation:  Set to 1 if the codec can decode potentially damaged data.

28.23. 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 u-integer). That means 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 not found it should be the second, etc.

28.24. AspectRatioType Element

type / id:  uinteger / 0x54B3
path:  \Segment\Tracks\TrackEntry\Video\AspectRatioType
documentation:  Specify the possible modifications to the aspect ratio.

28.25. GammaValue Element

type / id:  float / 0x2FB523
path:  \Segment\Tracks\TrackEntry\Video\GammaValue
documentation:  Gamma Value.

28.26. 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.

28.27. ChannelPositions Element

type / id:  binary / 0x7D7B
path:  \Segment\Tracks\TrackEntry\Audio\ChannelPositions
documentation:  Table of horizontal angles for each successive channel.
28.28. 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].

28.29. 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].

28.30. 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].

28.31. 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].

28.32. 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].

28.33. ContentSignature Element

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

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

28.35. ContentSigAlgo Element

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

28.36. ContentSigHashAlgo Element

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

28.37. CueRefCluster Element

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

28.38. 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.

28.39. 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.
28.40. 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.

28.41. 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 is based on TimestampScale. See [DivXWorldFonts].

28.42. 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 is based on TimestampScale. See [DivXWorldFonts].

28.43. 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.

29. Normative References


30. Informative References


Authors' Addresses

Steve Lhomme
Email: slhomme@matroska.org

Moritz Bunkus
Email: moritz@bunkus.org

Dave Rice
Email: dave@dericed.com