cellar S. Lhomme
Internet-Draft
Intended status: Standards Track D. Rice
Expires: January 7, 2017 July 6, 2016

Extensible Binary Meta Language
draft-lhomme-cellar-ebml-00

Abstract

This document defines the Extensible Binary Meta Language (EBML) format as a genearlized file format for any type of data in a hierarchical form. EBML is designed as a binary equivalent to XML and utilizes a storage-efficient approach to building nested Elements with identifiers, lengths, and values. Similar to how an XML Schema defines the structure and semantics of an XML Document, this document defines an EBML Schema to convey the semantics of an EBML Document.

Status of This Memo

Copyright Notice

Table of Contents

1. EBML specifications 2
1.1. Introduction 2
1.2. Notation and Conventions 3
1.3. Security Considerations 3
1.4. Structure 4
1.5. Variable Size Integer 4
1.5.1. VINT_WIDTH 4
1.5.2. VINT_MARKER 4
1.5.3. VINT_DATA 4
1.5.4. VINT Examples 5
1.6. Element ID 5
1.7. Element Data Size 6
1.8. EBML Element Types 8
1.9. EBML Document 11
1.9.1. EBML Header 12
1.9.2. EBML Body 12
1.10. EBML Stream 13
1.11. Elements semantic 13
1.11.1. EBML Schema 13
1.11.2. EBML Header Elements 22
1.11.3. Global elements (used everywhere in the format) . . 25
2. References 26
2.1. Normative References 27
2.2. Informative References 27
2.3. URIs 27
Authors' Addresses 27

1. EBML specifications

1.1. Introduction

EBML, short for Extensible Binary Meta Language, specifies a binary and octet (byte) aligned format inspired by the principle of XML.

The goal of the EBML Specification is to define a generic, binary, space-efficient format that may be utilized to define more complex formats (such as containers for multimedia content) using an EBML Schema. The definition of the EBML format recognizes the idea behind HTML and XML as a good one: separate structure and semantics allowing the same structural layer to be used with multiple, possibly widely differing semantic layers. Except for the EBML Header and a few global elements this specification does not define particular EBML format semantics; however this specification is intended to define how other EBML-based formats may be defined.

EBML uses a simple approach of building Elements upon three pieces of data (tag, length, and value) as this approach is well known, easy to parse, and allows selective data parsing. The EBML structure additionally allows for hierarchical arrangement to support complex structural formats in an efficient manner.

1.2. Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119](<https://tools.ietf.org/html/rfc2119>).

1.3. Security Considerations

EBML itself does not offer any kind of security. It has nothing to do with authentication, it does not provide confidentiality, only marginally useful and effective data integrity options (CRC elements).

EBML does not provide any kind of authorization.

Even if the semantic layer offers any kind of encryption, EBML itself may leak information at both the semantic layer (as declared via the DocType element) and within the EBML structure (you can derive the presence of EBML elements even with an unknown semantic layer with a heuristic approach; not without errors, of course, but with a certain degree of confidence).

Attacks on an EBML reader may include: - Invalid Element IDs that are longer than the limit stated in the EBMLMaxIDLength Element of the EBML Header. - Invalid Element IDs that are not encoded in the shortest-possible way. - Invalid Element IDs comprised of reserved values. - Invalid Element Data Size values that are longer than the limit stated in the EBMLMaxSizeLength Element of the EBML Header. - Invalid Element Data Size values (e.g. extending the length of the Element beyond the scope of the Parent Element; possibly triggering access-out-of-bounds issues). - Very high lengths in order to force out-of-memory situations resulting in a denial of service, access-out-of-bounds issues etc. - Missing Elements that are mandatory and have no declared default value. - Usage of "0x00" octets in EBML Elements with a string type. - Usage of invalid UTF-8 encoding in EBML Elements of UTF-8 type (e.g. in order to trigger access-out-of-bounds or buffer overflow issues). - Usage of invalid data in EBML Elements with a date type.

1.4. Structure

EBML uses a system of Elements to compose an EBML Document. Elements incorporate three parts: an Element ID, an Element Data Size, and Element Data. The Element Data, which is described by the Element ID, may include either binary data or one or many other Elements.

1.5. Variable Size Integer

The Element ID and Element Data Size are both encoded as a Variable Size Integer, developed according to a UTF-8 like system. The Variable Size Integer is composed of a VINT_WIDTH, VINT_MARKER, and VINT_DATA, in that order. Variable Size Integers shall be referred to as VINT for shorthand.

1.5.1. VINT_WIDTH

Each Variable Size Integer begins with a VINT_WIDTH which consists of zero or many zero-value bits. The count of consecutive zero-values of the VINT_WIDTH plus one equals the length in octets of the Variable Size Integer. For example, a Variable Size Integer that starts with a VINT_WIDTH which contains zero consecutive zero-value bits is one octet in length and a Variable Size Integer that starts with one consecutive zero-value bit is two octets in length. The VINT_WIDTH MUST only contain zero-value bits or be empty.

1.5.2. VINT_MARKER

The VINT_MARKER serves as a separator between the VINT_WIDTH and VINT_DATA. Each Variable Size Integer MUST contain exactly one VINT_MARKER. The VINT_MARKER MUST be one bit in length and contain a bit with a value of one. The first bit with a value of one within the Variable Size Integer is the VINT_MARKER.

1.5.3. VINT_DATA

The VINT_DATA portion of the Variable Size Integer includes all data that follows (but not including) the VINT_MARKER until end of the Variable Size Integer whose length is derived from the VINT_WIDTH. The bits required for the VINT_WIDTH and the VINT_MARKER combined use one bit per octet of the total length of the Variable Size Integer. Thus a Variable Size Integer of 1 octet length supplies 7 bits for VINT_DATA, a 2 octet length supplies 14 bits for VINT_DATA, and a 3 octet length supplies 21 bits for VINT_DATA. If the number of bits required for VINT_DATA are less than the bit size of VINT_DATA, then VINT_DATA may be zero-padded to the left to a size that fits. The VINT_DATA value MUST be expressed a big-endian unsigned integer.

1.5.4. VINT Examples

This table shows examples of Variable Size Integers at widths of 1 to 5 octets. The Representation column depicts a binary expression of Variable Size Integers where VINT_WIDTH is depicted by '0', the VINT_MARKER as '1', and the VINT_DATA as 'x'.

Octet Width	Size	Representation
1	2^7	1xxx xxxx
2	2^14	01xx xxxx xxxx xxxx
3	2^21	001x xxxx xxxx xxxx xxxx xxxx
4	2^28	0001 xxxx xxxx xxxx xxxx xxxx xxxx xxxx
5	2^35	0000 1xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note that data encoded as a Variable Size Integer may be rendered at octet widths larger than needed to store the data. In this table a binary value of 0b10 is shown encoded as different Variable Size Integers with widths from one octet to four octet. All four encoded examples have identical semantic meaning though the VINT_WIDTH and the padding of the VINT_DATA vary.

Binary Value	Octet Width	As Represented in Variable Size Integer
10	1	1000 0010
10	2	0100 0000 0000 0010
10	3	0010 0000 0000 0000 0000 0010
10	4	0001 0000 0000 0000 0000 0000 0000 0010

1.6. Element ID

The Element ID MUST be encoded as a Variable Size Integer. By default, EBML Element IDs may be encoded in lengths from one octet to four octets, although Element IDs of greater lengths may be used if the octet length of the EBML Document's longest Element ID is declared in the EBMLMaxIDLength Element of the EBML Header. The VINT_DATA component of the Element ID MUST NOT be set to either all zero values or all one values. The VINT_DATA component of the Element ID MUST be encoded at the shortest valid length. For example, an Element ID with binary encoding of 1011 1111 is valid, whereas an Element ID with binary encoding of 0100 0000 0011 1111 stores a semantically equal VINT_DATA but is invalid because a shorter VINT encoding is possible. The following table details this specific example further:

VINT_WIDTH	VINT_MARKER	VINT_DATA	Element ID Status
	1	0111111	Valid
0	1	00000000111111	Invalid

The octet length of an Element ID determines its EBML Class.

EBML Class	Octet Length	Number of Possible Element IDs
Class A	1	2^7 - 2 = 126
Class B	2	2^14 - 2^7 - 1 = 16,255
Class C	3	2^21 - 2^14 - 1 = 2,080,767
Class D	4	2^28 - 2^21 - 1 = 266,388,303

1.7. Element Data Size

The Element Data Size expresses the length in octets of Element Data. The Element Data Size itself MUST be encoded as a Variable Size Integer. By default, EBML Element Data Sizes can be encoded in lengths from one octet to eight octets, although Element Data Sizes of greater lengths MAY be used if the octet length of the EBML Document’s longest Element Data Size is declared in the EBMLMaxSizeLength Element of the EBML Header. Unlike the VINT_DATA of the Element ID, the VINT_DATA component of the Element Data Size is not required to be encoded at the shortest valid length. For example, an Element Data Size with binary encoding of 1011 1111 or a binary encoding of 0100 0000 0011 1111 are both valid Element Data Sizes and both store a semantically equal value.

Although an Element ID with all VINT_DATA bits set to zero is invalid, an Element Data Size with all VINT_DATA bits set to zero is allowed for EBML Data Types which do not mandate a non-zero length. An Element Data Size with all VINT_DATA bits set to zero indicates that the Element Data of the Element is zero octets in length. Such an Element is referred to as an Empty Element. If an Empty Element has a "default" value declared then that default value MUST be interpreted as the value of the Empty Element. If an Empty Element has no "default" value declared then the semantic meaning of Empty Element is defined as part of the definition of the EBML Element Types.

An Element Data Size with all VINT_DATA bits set to one is reserved as an indicator that the size of the Element is unknown. The only reserved value for the VINT_DATA of Element Data Size is all bits set to one. This rule allows for an Element to be written and read before the size of the Element is known; however unknown Element Data Size values SHOULD NOT be used unnecessarily. An Element with an unknown Element Data Size MUST be a Master-element in that it contains other EBML Elements as sub-elements. Master-elements MAY only use an unknown size if the "unknownsizeallowed" attribute of the EBML Schema is set to true. The end of a Master-element with unknown size is determined by the beginning of the next element that is not a valid sub-element of that Master-element. An Element with an unknown Element Data Size is referred to as an "Unknown-Sized Element".

For Element Data Sizes encoded at octet lengths from one to eight, this table depicts the range of possible values that can be encoded as an Element Data Size. An Element Data Size with an octet length of 8 is able to express a size of 2^56-2 or 72,057,594,037,927,934 octets (or about 72 petabytes). The maximum possible value that can be stored as Element Data Size is referred to as "VINTMAX".

Octet Length	Possible Value Range
1	0 to 2^7-2
2	0 to 2^14-2
3	0 to 2^21-2
4	0 to 2^28-2
5	0 to 2^35-2
6	0 to 2^42-2
7	0 to 2^49-2
8	0 to 2^56-2

If the length of Element Data equals 2^(n*7)-1 then the octet length of the Element Data Size MUST be at least n+1. This rule prevents an Element Data Size from being expressed as a reserved value. For example, an Element with an octet length of 127 MUST NOT be encoded in an Element Data Size encoding with a one octet length. The following table clarifies this rule by showing a valid and invalid expression of an Element Data Size with a VINT_DATA of 127 (which is equal to 2^(1*7)-1).

VINT_WIDTH	VINT_MARKER	VINT_DATA	Element Data Size Status
	1	1111111	Reserved (meaning Unknown)
0	1	00000001111111	Valid (meaning 127 octets)

1.8. EBML Element Types

Each defined EBML Element MUST have a declared Element Type. The Element Type defines a concept for storing data that may be constrained by length, endianness, and purpose.

Element Data Type	Signed Integer
Endianness	Big-endian
Length	A Signed Integer Element MUST declare a length that is no greater than 8 octets. An Signed Integer Element with a zero-octet length represents an integer value of zero.
Definition	A Signed Integer stores an integer (meaning that it can be written without a fractional component) which may be negative, positive, or zero. Because EBML limits Signed Integers to 8 octets in length a Signed Element may store a number from -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807.

Element Data Type	Unsigned Integer
Endianness	Big-endian
Length	A Unsigned Integer Element MUST declare a length that is no greater than 8 octets. An Unsigned Integer Element with a zero-octet length represents an integer value of zero.
Definition	An Unsigned Integer stores an integer (meaning that it can be written without a fractional component) which may be positive or zero. Because EBML limits Unsigned Integers to 8 octets in length an unsigned Element may store a number from 0 to 18,446,744,073,709,551,615.

Element Data Type	Float
Endianness	Big-endian
Length	A Float Element MUST declare of length of either 0 octets (0 bit), 4 octets (32 bit) or 8 octets (64 bit). A Float Element with a zero-octet length represents a numerical value of zero.
Definition	A Float Elements stores a floating-point number as defined in IEEE 754.

Element Data Type	String
Endianness	None
Length	A String Element may declare any length from zero to "VINTMAX".
Definition	A String Element may either be empty (zero-length) or contain Printable ASCII characters in the range of "0x20" to "0x7E". Octets with all bits set to zero may follow the string value when needed.

Element Data Type	UTF-8
Endianness	None
Length	A UTF-8 Element may declare any length from zero to "VINTMAX".
Definition	A UTF-8 Element shall contain only a valid Unicode string as defined in [RFC2279](<http://www.faqs.org/rfcs/rfc2279.html>). Octets with all bits set to zero may follow the UTF-8 value when needed.

Element Data Type	Date
Endianness	None
Length	A Date Element MUST declare a length of either 0 octets or 8 octets. A Date Element with a zero-octet length represents a timestamp of 2001-01-01T00:00:00.000000000 UTC.
Definition	The Date Element MUST contain a Signed Integer that expresses a point in time referenced in nanoseconds from the precise beginning of the third millennium of the Gregorian Calendar in Coordinated Universal Time (also known as 2001-01-01T00:00:00.000000000 UTC). This provides a possible expression of time from 1708-09-11T00:12:44.854775808 UTC to 2293-04-11T11:47:16.854775807 UTC.

Element Data Type	Master-element
Endianness	None
Length	A Master-element may declare any length from zero to "VINTMAX". The Master-element may also use an unknown length. See the section on Element Data Size for rules that apply to elements of unknown length.
Definition	The Master-element contains zero, one, or many other elements. Elements contained within a Master-element must be defined for use at levels greater than the level of the Master-element. For instance is a Master-element occurs on level 2 then all contained Elements must be valid at level 3. Element Data stored within Master-elements SHOULD only consist of EBML Elements and SHOULD NOT contain any data that is not part of an EBML Element. When EBML is used in transmission or streaming, data that is not part of an EBML Element is permitted to be present within a Master-element if "unknownsizeallowed" is enabled within that Master-element’s definition. In this case, the reader should skip data until a valid Element ID of the same level or the next greater level of the Master-element is found. What Element IDs are considered valid within a Master-element is identified by the EBML Schema for that version of the EBML Document Type. Any data contained with a Master-element that is not part of an Element SHOULD be ignored.

Element Data Type	Binary
Endianness	None
Length	A binary element may declare any length from zero to "VINTMAX".
Definition	The contents of a Binary element should not be interpreted by the EBML parser.

1.9. EBML Document

An EBML Document is comprised of only two components, an EBML Header and an EBML Body. An EBML Document MUST start with an EBML Header which declares significant characteristics of the entire EBML Body.

An EBML Document MAY only consist of EBML Elements and MUST NOT contain any data that is not part of an EBML Element. The initial EBML Element of an EBML Document and the Elements that follow it are considered Level 0 Elements. If an EBML Master-element is considered to be at level N and it contains one or many other EBML Elements then the contained Elements shall be considered at Level N+1. Thus a Level 2 Element would have to be contained by a Master-element (at Level 1) that is contained by another Master-element (at Level 0).

1.9.1. EBML Header

The EBML Header is a declaration that provides processing instructions and identification of the EBML Body. The EBML Header may be considered as analogous to an XML Declaration. All EBML Documents MUST begin with a valid EBML Header.

The EBML Header documents the EBML Schema (also known as the EBML DocType) that may be used to semantically interpret the structure and meaning of the EBML Document. Additionally the EBML Header documents the versions of both EBML and the EBML Schema that were used to write the EBML Document and the versions required to read the EBML Document.

The EBML Header consists of a single Master-element with an Element ID of 'EBML'. The EBML Header MUST ONLY contain EBML Elements that are defined as part of the EBML Specification.

All EBML Elements within the EBML Header MUST NOT utilize any Element ID with a length greater than 4 octets. All EBML Elements within the EBML Header MUST NOT utilize any Element Data Size with a length greater than 4 octets.

1.9.2. EBML Body

All data of an EBML Document following the EBML Header may be considered the EBML Body. The end of the EBML Body, as well as the end of the EBML Document that contains the EBML Body, is considered as whichever comes first: the beginning of a new level 0 EBML Header or the end of the file. The EBML Body MAY only consist of EBML Elements and MUST NOT contain any data that is not part of an EBML Element. Although the EBML specification itself defines precisely what EBML Elements are to be used within the EBML Header, the EBML specification does not name or define what EBML Elements are to be used within the EBML Body. The definition of what EBML Elements are to be used within the EBML Body is defined by an EBML Schema.

1.10. EBML Stream

An EBML Stream is a file that consists of one or many EBML Documents that are concatenated together. An occurrence of a Level 0 EBML Header marks the beginning of an EBML Document.

1.11. Elements semantic

1.11.1. EBML Schema

An EBML Schema is an XML Document that defines the properties, arrangement, and usage of EBML Elements that compose a specific EBML Document Type. The relationship of an EBML Schema to an EBML Document may be considered analogous to the relationship of an XML Schema [1] to an XML Document [2]. An EBML Schema MUST be clearly associated with one or many EBML Document Types. An EBML Schema must be expressed as well-formed XML. An EBML Document Type is identified by a unique string stored within the EBML Header element called DocType; for example "matroska" or "webm".

As an XML Document, the EBML Schema MUST use "<EBMLSchema>" as the top level element. The "<EBMLSchema>" element MAY contain "<element>" sub-elements. Each "<element>" defines one EBML Element through the use of several attributes which are defined in the section on Section 1.11.1.1. EBML Schemas MAY contain additional attributes to extend the semantics but MUST NOT conflict is the definitions of the "<element>" attributes defined within this specification.

Within the EBML Schema each EBML Element is defined to occur at a specific level. For any specificied EBML Element that is not at level 0, the Parent EBML Element refers to the EBML Master-element that that EBML Element is contained within. For any specifiied EBML Master-element the Child EBML Element refers to the EBML Elements that may be immediately contained within that Master-element. For any EBML Element that is not defined at level 0, the Parent EBML Element may be identified by the preceding "<element>" node which has a lower value as the defined "level" attribute. The only exception for this rule are Global EBML Elements which may occur within any Parent EBML Element within the restriction of the Global EBML Element’s range declaration.

An EBML Schema MUST declare exactly one Element at Level 0 (referred to as the Root Element) that MUST occur exactly once within an EBML Document. The Root Element MUST be mandatory (with minOccurs set to 1) and MUST be defined to occur exactly once (maxOccurs set to 1). Note that the EBML and Void Elements may also occur at Level 0 but are not considered to be Root Elements.

Elements defined to only occur at Level 1 are known as Top-Level Elements.

The EBML Schema does not itself document the EBML Header, but documents all data of the EBML Document that follows the EBML Header. The EBML Header itself is documented by this specification in the Section 1.11.2 section. The EBML Schema also does not document Global Elements that are defined by the EBML Specification (namely Void and CRC-32).

1.11.1.1. EBML Schema Element Attributes

Within an EBML Schema the "<EBMLSchema>" uses the following attributes to define the EBML Schema:

attribute name	required	definition
docType	Yes	The "docType" lists the official name of the EBML Document Type that is defined by the EBML Schema; for example, "<EBMLSchema docType="matroska">".
version	Yes	The "version" lists an incremental non-negative integer that specifies the version of the docType documented by the EBML Schema. Unlike XML Schemas, an EBML Schema documents all versions of a docType’s definition rather than using separate EBML Schemas for each version of a docType. Elements may be introduced and deprecated by using the "minver" and "maxver" attributes of .

Within an EBML Schema the "<element>" uses the following attributes to define an EBML Element.

attribute name	required	definition
name	Yes	The official human-readable name of the EBML Element. The value of the name MUST be in the form of an NCName as defined by the XML Schema specification [3].
level	Yes	The level notes at what hierarchical depth the EBML

		Element may occur within an EBML Document. The Root Element of an EBML Document is at level 0 and the Elements that it may contain are at level 1. The level MUST be expressed as an integer. Note that Elements defined as "global" and "recursive" MAY occur at a level greater than or equal to the defined "level".
global	No	A boolean to express if an EBML Element MUST occur at its defined level or may occur within any Parent EBML Element. If the "global" attribute is not expressed for that Element then that element is to be considered not global.
id	Yes	The Element ID expressed in hexadecimal notation prefixed by a "0x". To reduce the risk of false positives while parsing EBML Streams, the IDs of the Root Element and Top-Level Elements SHOULD be at least 4 octets in length. Element IDs defined for use at Level 0 or Level 1 MAY use shorter octet lengths to facilitate padding and optimize edits to EBML Documents; for instance, the EBML Void Element uses an Element ID with a one octet length to allow its usage in more writing and editing scenarios.
minOccurs	No	An integer to express the minimal number of occurrences that the EBML Element MUST occur within its Parent Element if its Parent Element is used. If the Element has no Parent Level (as is the case with Elements at Level 0), then minOccurs refers to constaints on the Element’s occurrence within the EBML Document. If the minOccurs attribute is not expressed for that Element then that Element

		shall be considered to have a minOccurs value of 0. This value of minOccurs MUST be a positive integer. The semantic meaning of minOccurs within an EBML Schema is considered analogous to the meaning of minOccurs within an XML Schema [4]. Note that Elements with minOccurs set to "1" that also have a default value declared are not required to be stored but are required to be interpretted, see the Section 1.11.1.6.
maxOccurs	No	A value to express the maximum number of occurrences that the EBML Element MAY occur within its Parent Element if its Parent Element is used. If the Element has no Parent Level (as is the case with Elements at Level 0), then maxOccurs refers to constaints on the Element's occurrence within the EBML Document. This value may be either a positive integer or the term "unbounded" to indicate there is no maximum number of occurrences or the term "identical" to indicate that the Element is an Section 1.11.1.3. If the maxOccurs attribute is not expressed for that Element then that Element shall be considered to have a maxOccurs value of 1. The semantic meaning of maxOccurs within an EBML Schema is considered analogous to the meaning of minOccurs within an XML Schema [5], with EBML Schema adding the concept of Identically Recurring Elements.
range	No	For Elements which are of numerical types (Unsigned Integer, Signed Integer, Float, and Date) a numerical range may be specified. If specified that the value of the EBML Element

		MUST be within the defined range inclusively. See the Section 1.11.1.4 for rules applied to expression of range values.
default	No	A default value may be provided. If an Element is mandatory but not written within its Parent EBML Element, then the parser of the EBML Document MUST insert the defined default value of the Element. EBML Elements that are Master-elements MUST NOT declare a default value.
type	Yes	As defined within the Section 1.8, the type MUST be set to one of the following values: 'integer' (signed integer), 'uinteger' (unsigned integer), 'float', 'string', 'date', 'utf-8', 'master', or 'binary'.
unknownsizeallowed	No	A boolean to express if an EBML Element MAY be used as an "Unknown-Sized Element" (having all VINT_DATA bits of Element Data Size set to 1). The "unknownsizeallowed" attribute only applies to Master-elements. If the "unknownsizeallowed" attribute is not used it is assumed that the element is not allowed to use an unknown Element Data Size.
recursive	No	A boolean to express if an EBML Element MAY be stored recursively. In this case the Element MAY be stored at levels greater that defined in the "level" attribute if the Element is a Child Element of a Parent Element with the same Element ID. The "recursive" attribute only applies to Master-elements. If the "recursive" attribute is not used it is assumed that the element is not allowed to be used recursively.
minver	No	The "minver" (minimum version) attribute stores a non-negative

		integer that represents the first version of the docType to support the element. If the "minver" attribute is not used it is assumed that the element has a minimum version of "1".
maxver	No	The "maxver" (maximum version) attribute stores a non-negative integer that represents the last or most recent version of the docType to support the element. If the "maxver" attribute is not used it is assumed that the element has a maximum version equal to the value stored in the "version" attribute of .

The "<element>" nodes shall contain a description of the meaning and use of the EBML Element stored within one or many "<documentation>" sub-elements. The "<documentation>" sub-element may use a "lang" attribute which may be set to the RFC 5646 value of the language of the element’s documentation. The "<documentation>" sub-element may use a "type" attribute to distinguish the meaning of the documentation. Recommended values for the "<documentation>" sub-element’s "type" attribute include: "definition", "rationale", "usage notes", and "references".

The "<element>" nodes MUST be arranged hierarchically according to the permitted structure of the EBML Document Type. An "<element>" node that defines an EBML Element which is a Child Element of another Parent Element MUST be stored as an immediate sub-element of the "<element>" node that defines the Parent Element. "<element>" nodes that define Level 0 Elements and Global Elements should be sub-elements of "<EBMLSchema>".

1.11.1.2. EBML Schema Example

```
<?xml version="1.0" encoding="utf-8"?>
<EBMLSchema docType="files-in-ebml-demo" version="1">
 <!-- Root Element-->
 <element name="Files" level="0" id="0x1946696C" type="master">
  <documentation lang="en" type="definition">Container of data and
  attributes representing one or many files.</documentation>
  <element name="File" level="1" id="0x6146" type="master" minOccurs="1"
  maxOccurs="unbounded">
   <documentation lang="en" type="definition">An attached file.
   </documentation>
   <element name="FileName" level="2" id="0x614E" type="utf-8"
   minOccurs="1">
    <documentation lang="en" type="definition">Filename of the attached
    file.</documentation>
   </element>
   <element name="MimeType" level="2" id="0x464D" type="string"
     minOccurs="1">
    <documentation lang="en" type="definition">MIME type of the
    file.</documentation>
   </element>
   <element name="ModificationTimestamp" level="2" id="0x4654"
     type="date" minOccurs="1">
    <documentation lang="en" type="definition">Modification timestamp of
    the file.</documentation>
   </element>
   <element name="Data" level="2" id="0x4664" type="binary"
     minOccurs="1">
    <documentation lang="en" type="definition">The data of the
    file.</documentation>
   </element>
  </element>
 </element>
</EBMLSchema>
```

1.11.1.3. Identically Recurring Elements

An Identically Recurring Element is an Element that may occur within its Parent Element more than once but that each recurrence within that Parent Element MUST be identical both in storage and semantics. Identically Recurring Elements are permitted to be stored multiple times within the same Parent Element in order to increase data resilience and optimize the use of EBML in transmission. Identically Recurring Elements SHOULD include a CRC-32 Element as a Child Element; this is especially recommended when EBML is used for long-term storage or transmission. If a Parent Element contains more than one copy of an Identically Recurring Element which includes a CRC-32 Child Element then the first instance of the Identically Recurring Element with a valid CRC-32 value should be used for interpretation.

If a Parent Element contains more than one copy of an Identically Recurring Element which does not contain a CRC-32 Child Element or if CRC-32 Child Elements are present but none are valid then the first instance of the Identically Recurring Element should be used for interpretation.

1.11.1.4. Expression of range

The "range" attribute MUST only be used with EBML Elements that are either "signed integer", "unsigned integer", or "float". The "range" attribute does not support date EBML Elements. The "range" expression may contain whitespace for readability but whitespace within a "range" expression MUST NOT convey meaning. The expression of the "range" MUST adhere to one of the following forms:

- "x-y" where x and y are integers or floats and "y" must be greater than "x", meaning that the value must be greater than or equal to "x" and less than or equal to "y".

- ">x" where "x" is an integer or float, meaning that the value MUST be greater than "x".

- "x" where "x" is an integer or float, meaning that the value MUST be equal "x".

The "range" may use the prefix "not" to indicate that the expressed range is negated. Please also see the section on Section 1.11.1.5.

1.11.1.5. Textual expression of Floats

When a float value is represented textually in an EBML Schema, such as within a "default" or "range" value, the float values MUST be expressed as a Hexadecimal Floating-Point Constants as defined in the C11 standard ISO/IEC 9899:2011 [6] (see section 6.4.4.2 on Floating Constants). The following table provides examples of expressions of float ranges.

as decimal	as Hexadecimal Floating-Point Constants
0.0-1.0	"0x0p+1-0x1p+0"
1.0-256.0	"0x1p+0-0x1p+8"
0.857421875	"0x1.b7p-1"
-1.0--0.857421875	"-0x1p+0--0x1.b7p-1"

Within an expression of a float range, as in an integer range, the "-" (hyphen) character is the separator between the minimal and maximum value permitted by the range. Note that Hexadecimal Floating-Point Constants also use a "-" (hyphen) when indicating a negative binary power. Within a float range, when a "-" (hyphen) is immediately preceded by a letter "p", then the "-" (hyphen) is a part of the Hexadecimal Floating-Point Constant which notes negative binary power. Within a float range, when a "-" (hyphen) is not immediately preceded by a letter "p", then the "-" (hyphen) represents the separator between the minimal and maximum value permitted by the range.

1.11.1.6. Note on the Use of default attributes to define Mandatory EBML Elements

If a Mandatory EBML Element has a default value declared by an EBML Schema and the EBML Element’s value is equal to the declared default value then that Element is not required to be present within the EBML Document if its Parent EBML Element is present. In this case, the default value of the Mandatory EBML Element may be assumed although the EBML Element is not present within its Parent EBML Element. Also in this case the parser of the EBML Document MUST insert the defined default value of the Element.

If a Mandatory EBML Element has no default value declared by an EBML Schema and its Parent EBML Element is present than the EBML Element must be present as well. If a Mandatory EBML Element has a default value declared by an EBML Schema and its Parent EBML Element is present and the EBML Element’s value is NOT equal to the declared default value then the EBML Element MUST be used.

This table clarifies if a Mandatory EBML Element MUST be written, according to if the default value is declared, if the value of the EBML Element is equal to the declared default value, and if the Parent EBML Element is used.

Is the default value declared?	Is the value equal to default?	Is the Parent Element used?	Then is storing the EBML Element required?
Yes	Yes	Yes	No
Yes	Yes	No	No
Yes	No	Yes	Yes
Yes	No	No	No
No	n/a	Yes	Yes
No	n/a	No	No
No	n/a	Yes	Yes
No	n/a	No	No

1.11.1.7. Note on the Use of default attributes to define non-Mandatory EBML Elements

If an EBML Element is not Mandatory, has a defined default value, and is an Empty EBML Element then the EBML Element MUST be interpreted as expressing the default value.

1.11.2. EBML Header Elements

This specification here contains definitions of all EBML Elements of the EBML Header.

Name	EBML
Level	0
EBML ID	"0x1A45DFA3"
Mandatory	Yes
Multiple	No
Range	-
Default	-
Type	Master-element
Description	Set the EBML characteristics of the data to follow. Each EBML Document has to start with this.

Name	EBMLVersion
Level	1
EBML ID	"0x4286"
Mandatory	Yes
Multiple	No
Range	1
Default	1
Type	Unsigned Integer
Description	The version of EBML parser used to create the EBML Document.

Name	EBMLReadVersion
Level	1
EBML ID	"0x42F7"
Mandatory	Yes
Multiple	No
Range	1
Default	1
Type	Unsigned Integer
Description	The minimum EBML version a parser has to support to read this EBML Document.

Name	EBMLMaxIDLength
Level	1
EBML ID	"0x42F2"
Mandatory	Yes
Multiple	No
Range	>3
Default	4
Type	Unsigned Integer
Description	The EBMLMaxIDLength is the maximum length in octets of the Element IDs to be found within the EBML Body. An EBMLMaxIDLength value of four is recommended, though larger values are allowed.

Name	EBMLMaxSizeLength
Level	1
EBML ID	"0x42F3"
Mandatory	Yes
Multiple	No
Range	>0
Default	8
Type	Unsigned Integer
Description	The EBMLMaxSizeLength is the maximum length in octets of the expression of all Element Data Sizes to be found within the EBML Body. To be clear EBMLMaxSizeLength documents the maximum 'length' of all Element Data Size expressions within the EBML Body and not the maximum 'value' of all Element Data Size expressions within the EBML Body. Elements that have a Element Data Size expression which is larger in octets than what is expressed by EBMLMaxSizeLength SHALL be considered invalid.

Name	DocType
Level	1
EBML ID	"0x4282"
Mandatory	Yes
Multiple	No
Range	-
Default	matroska
Type	String
Description	A string that describes and identifies the content of the EBML Body that follows this EBML Header.

Name	DocTypeVersion
Level	1
EBML ID	"0x4287"
Mandatory	Yes
Multiple	No
Range	-
Default	1
Type	Unsigned Integer
Description	The version of DocType interpreter used to create the EBML Document.

Name	DocTypeReadVersion
Level	1
EBML ID	"0x4285"
Mandatory	Yes
Multiple	No
Range	-
Default	1
Type	Unsigned Integer
Description	The minimum DocType version an interpreter has to support to read this EBML Document.

1.11.3. Global elements (used everywhere in the format)

Name	CRC-32
Level	1+
Global	Yes
EBML ID	"0xBF"
Mandatory	No
Range	-
Default	-
Type	Binary
Description	The CRC-32 Element contains a 32 bit Cyclic Redundancy Check value of all the Element Data of the Parent Element as stored except for the CRC-32 Element itself. When the CRC-32 Element is present, the CRC-32 Element MUST be the first ordered Element within its Parent Element for easier reading. All Top-Level Elements of an EBML Document SHOULD include a CRC-32 Element as a Child Element. The CRC in use is the IEEE-CRC-32 algorithm as used in the ISO 3309 standard and in section 8.1.1.6.2 of ITU-T recommendation V.42, with initial value of "0xFFFFFFFF". The CRC value MUST be computed on a little endian bitstream and MUST use little endian storage.

Name	Void
Level	0+
Global	Yes
EBML ID	"0xEC"
Mandatory	No
Multiple	Yes
Range	-
Default	-
Type	Binary
Description	Used to void damaged data, to avoid unexpected behaviors when using damaged data. The content is discarded. Also used to reserve space in a sub-element for later use.

2. References

2.1. Normative References

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

2.2. Informative References

[RFC2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, DOI 10.17487/RFC2279, January 1998, <http://www.rfc-editor.org/info/rfc2279>.

2.3. URIs

[1] http://www.w3.org/XML/Schema#dev

[2] http://www.w3.org/TR/xml/

[3] http://www.w3.org/TR/1999/REC-xml-names-19990114/#ns-decl

[4] https://www.w3.org/TR/xmlschema-0/#ref6

[5] https://www.w3.org/TR/xmlschema-0/#ref6

[6] http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf

Authors' Addresses

Steve Lhomme

Dave Rice

cellar S. Lhomme
Internet-Draft
Intended status: Standards Track M. Bunkus
Expires: January 9, 2017
D. Rice
July 8, 2016

Matroska
draft-lhomme-cellar-matroska-00

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 9
2. Status of this document 10
3. Security Considerations 10
4. IANA Considerations 11
5. Notations and Conventions 11
6. Basis in EBML 11
  6.1. Added Constaints on EBML 11
  6.2. Matroska Design 11
    6.2.1. Language Codes 11
    6.2.2. Physical Types 11
    6.2.3. Block Structure 12
    6.2.4. Lacing 13
7. Matroska Schema 17
  7.1. EBML Schema Element Attributes 18
  7.2. Matroska Additions to Schema Element Attributes 22
  7.3. Matroska Schema 23
8. Segment 23
9. SeekHead 23
10. Seek 24
11. SeekID 24
12. SeekPosition 25
13. Info 25
14. SegmentUID 26
15. SegmentFilename 26
16. PrevUID 27
17. PrevFilename 27
18. NextUID 28
19. NextFilename 28
20. SegmentFamily 29
21. ChapterTranslate 29
22. ChapterTranslateEditionUID 30
23. ChapterTranslateCodec 31
24. ChapterTranslateID 31
25. TimecodeScale 32
26. Duration 32
27. DateUTC 32
28. Title 33
29. MuxingApp 33
30. WritingApp 34
31. Cluster 34
32. Timecode 35
33. SilentTracks 35
34. SilentTrackNumber 36
35. Position 36
36. PrevSize 37
37. SimpleBlock 37

38. BlockGroup 38
39. Block 39
40. BlockVirtual 39
41. BlockAdditions 40
42. BlockMore 40
43. BlockAddID 41
44. BlockAdditional 41
45. BlockDuration 42
46. ReferencePriority 42
47. ReferenceBlock 43
48. ReferenceVirtual 44
49. CodecState 44
50. DiscardPadding 44
51. Slices 45
52. TimeSlice 45
53. LaceNumber 46
54. FrameNumber 47
55. BlockAdditionID 48
56. Delay 49
57. SliceDuration 49
58. ReferenceFrame 50
59. ReferenceOffset 50
60. ReferenceTimeCode 50
61. EncryptedBlock 51
62. Tracks 52
63. TrackEntry 52
64. TrackNumber 53
65. TrackUID 53
66. TrackType 54
67. FlagEnabled 54
68. FlagDefault 54
69. FlagForced 55
70. FlagLacing 56
71. MinCache 56
72. MaxCache 56
73. DefaultDuration 57
74. DefaultDecodedFieldDuration 57
75. TrackTimecodeScale 58
76. TrackOffset 59
77. MaxBlockAdditionID 59
78. Name 59
79. Language 60
80. CodecID 60
81. CodecPrivate 61
82. CodecName 61
83. AttachmentLink 62
84. CodecSettings 62
85. CodecInfoURL 63

86. CodecDownloadURL 63
87. CodecDecodeAll 64
88. TrackOverlay 64
89. CodecDelay 65
90. SeekPreRoll 66
91. TrackTranslate 66
92. TrackTranslateEditionUID 66
93. TrackTranslateCodec 67
94. TrackTranslateTrackID 68
95. Video 68
96. FlagInterlaced 69
97. FieldOrder 69
98. StereoMode 70
99. AlphaMode 71
100. OldStereoMode 72
101. PixelWidth 73
102. PixelHeight 73
103. PixelCropBottom 74
104. PixelCropTop 74
105. PixelCropLeft 75
106. PixelCropRight 75
107. DisplayWidth 76
108. DisplayHeight 76
109. DisplayUnit 77
110. AspectRatioType 78
111. ColourSpace 78
112. GammaValue 79
113. FrameRate 79
114. Colour 80
115. MatrixCoefficients 80
116. BitsPerChannel 81
117. ChromaSubsamplingHorz 82
118. ChromaSubsamplingVert 83
119. CbSubsamplingHorz 84
120. CbSubsamplingVert 85
121. ChromaSitingHorz 86
122. ChromaSitingVert 87
123. Range 87
124. TransferCharacteristics 88
125. Primaries 89
126. MaxCLL 90
127. MaxFALL 91
128. MasteringMetadata 92
129. PrimaryRChromaticityX 92
130. PrimaryRChromaticityY 93
131. PrimaryGChromaticityX 94
132. PrimaryGChromaticityY 95
133. PrimaryBChromaticityX 96

134. PrimaryBChromaticityY 97
135. WhitePointChromaticityX 98
136. WhitePointChromaticityY 99
137. LuminanceMax 100
138. LuminanceMin 101
139. Audio 102
140. SamplingFrequency 103
141. OutputSamplingFrequency 103
142. Channels 104
143. ChannelPositions 104
144. BitDepth 105
145. TrackOperation 105
146. TrackCombinePlanes 106
147. TrackPlane 106
148. TrackPlaneUID 107
149. TrackPlaneType 107
150. TrackJoinBlocks 108
151. TrackJoinUID 109
152. TrickTrackUID 109
153. TrickTrackSegmentUID 109
154. TrickTrackFlag 110
155. TrickMasterTrackUID 110
156. TrickMasterTrackSegmentUID 111
157. ContentEncodings 112
158. ContentEncoding 112
159. ContentEncodingOrder 113
160. ContentEncodingScope 113
161. ContentEncodingType 114
162. ContentCompression 115
163. ContentCompAlgo 116
164. ContentCompSettings 117
165. ContentEncryption 118
166. ContentEncAlgo 119
167. ContentEncKeyID 120
168. ContentSignature 121
169. ContentSigKeyID 122
170. ContentSigAlgo 123
171. ContentSigHashAlgo 124
172. Cues 125
173. CuePoint 126
174. CueTime 126
175. CueTrackPositions 126
176. CueTrack 127
177. CueClusterPosition 128
178. CueRelativePosition 128
179. CueDuration 129
180. CueBlockNumber 129
181. CueCodecState 130

182. CueReference 131
183. CueRefTime 131
184. CueRefCluster 132
185. CueRefNumber 132
186. CueRefCodecState 133
187. Attachments 134
188. AttachedFile 134
189. FileDescription 134
190. FileName 135
191. FileMimeType 135
192. FileData 136
193. FileUID 136
194. FileReferral 137
195. FileUsedStartTime 137
196. FileUsedEndTime 138
197. Chapters 138
198. EditionEntry 139
199. EditionUID 139
200. EditionFlagHidden 140
201. EditionFlagDefault 140
202. EditionFlagOrdered 141
203. ChapterAtom 141
204. ChapterUID 142
205. ChapterStringUID 143
206. ChapterTimeStart 143
207. ChapterTimeEnd 144
208. ChapterFlagHidden 144
209. ChapterFlagEnabled 145
210. ChapterSegmentUID 145
211. ChapterSegmentEditionUID 146
212. ChapterPhysicalEquiv 147
213. ChapterTrack 147
214. ChapterTrackNumber 148
215. ChapterDisplay 148
216. ChapString 149
217. ChapLanguage 150
218. ChapCountry 150
219. ChapProcess 151
220. ChapProcessCodecID 152
221. ChapProcessPrivate 152
222. ChapProcessCommand 153
223. ChapProcessTime 154
224. ChapProcessData 154
225. Tags 155
226. Tag 156
227. Targets 156
228. TargetTypeValue 156
229. TargetType 157

230. TagTrackUID 158
231. TagEditionUID 158
232. TagChapterUID 159
233. TagAttachmentUID 160
234. SimpleTag 160
235. TagName 161
236. TagLanguage 161
237. TagDefault 162
238. TagString 162
239. TagBinary 163
240. Beginning of File 163
241. Block Timecodes 163
242. Default decoded field duration 164
243. Default Values 164
244. DRM 165
245. EBML Class 165
246. Encryption 165
247. Image cropping 165
248. Matroska version indicators 166
249. Mime Types 166
250. Octet 166
251. Overlay Track 167
252. Position References 167
253. Raw Timecode 167
254. Linked Segments 167
  254.1. Hard Linking 167
  254.2. Soft Linking 168
  254.3. Medium Linking 168
255. Timecode Types 169
256. TimecodeScale 169
257. TimecodeScale Rounding 170
258. Track Flags 172
  258.1. Default flag 172
  258.2. Forced flag 172
259. TrackTimecodeScale 172
260. Unknown elements 174
261. Multi-planar and 3D videos 174
262. Track Operation 175
263. Matroska Element Ordering Guidelines 175
  263.1. Top-Level Elements 176
  263.2. CRC-32 176
  263.3. SeekHead 176
  263.4. Cues (index) 177
  263.5. Info 177
  263.6. Chapters 177
  263.7. Attachments 177
  263.8. Tags 177
  263.9. Optimum layout from a muxer 178

263.10. Optimum layout after editing tags 178
263.11. Optimum layout with Cues at the front 178
263.12. Cluster Timecode 179
264. CodecID 179
264.1. Example 1 : basic chaptering 187
264.2. Example 2 : nested chapters 188
264.2.1. The Micronauts "Bleep To Bleep" 188
264.3. Edition and chapter flags 189
264.3.1. Chapter flags 189
264.3.2. Edition flags 189
264.4. Menu features 189
264.4.1. Matroska Script (0) 190
264.4.2. DVD menu (1) 190
265. Subtitles 191
266. Images Subtitles 192
267. SRT Subtitles 195
268. SSA/ASS Subtitles 196
269. USF Subtitles 199
270. WebVTT 199
270.1. Storage of WebVTT in Matroska 199
270.1.1. CodecID: codec identification 199
270.1.2. CodecPrivate: storage of gloal WebVTT blocks . . . 199
270.1.3. Storage of non-global WebVTT blocks 199
270.1.4. Storage of Cues in Matroska blocks 200
270.1.5. BlockAdditions: storing non-global WebVTT blocks,
Cue Settings Lists and Cue identifiers 200
270.2. Examples of transformation 200
270.2.1. Example WebVTT file 200
270.2.2. CodecPrivate 201
270.2.3. Storage of Cue 1 201
270.2.4. Storage of Cue 2 201
270.2.5. Storage of Cue 3 201
270.2.6. Storage of Cue 4 201
270.3. Storage of WebVTT in Matroska vs. WebM 201
271. Tagging 202
271.1. Why official tags matter 202
271.2. Tag translations 203
271.3. Tag Formatting 203
271.4. Target types 203
271.5. Official tags 204
271.6. Notes 204
272. Matroska Streaming 205
273. File Access 205
274. Live Streaming 206
275. Menu Specifications 206
276. Introduction 206
277. Requirements 207
277.1. Highlights/Hotspots 207

277.2. Playback features 207
277.3. Player requirements 208
278. Working Graph 208
279. Ideas 208
280. Data Structure 208
281. Overhead 209
281.1. Example 1 - low bitrate audio 209
281.1.1. Conclusions 212
281.2. Example 2 - medium bitrate audio 212
281.3. Example 3 - high bitrate audio 212
281.4. Example 4 - low bitrate video 212
281.4.1. First example - all I frames (CBR) 213
281.4.2. Second example - 1 I frame for 1 B frame 215
281.5. Example 5 - medium bitrate video 216
281.6. Example 6 - high bitrate video 216
281.7. Example 7 - low bitrate video + low bitrate audio . . . 216
281.8. Example 8 - medium bitrate video + medium bitrate audio 216
281.9. Example 9 - high bitrate video + high bitrate audio . . 216
282.1. URIs 216
Authors' Addresses 220

1. Introduction

Matroska aims to become THE standard of multimedia container formats. It was derived from a project called MCF [1], but differentiates from it significantly because it is based on EBML [2] (Extensible Binary Meta Language), a binary derivative of XML. EBML enables significant advantages in terms of future format extensibility, without breaking file support in old parsers.

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

- It is NOT a video or audio compression format (video codec)

- It is an envelope for which there can be many audio, video and subtitles streams, allowing the user to store a complete movie or CD in a single file.

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

- Fast seeking in the file

- Chapter entries

- Full metadata (tags) support

- Selectable subtitle/audio/video streams
- Modularly expandable
- Error resilience (can recover playback even when the stream is damaged)
- Streamable over the internet and local networks (HTTP, CIFS, FTP, etc)
- Menus (like DVDs have)

Matroska is an open standards project. This means for personal use it is absolutely free to use and that the technical specifications describing the bitstream are open to everybody, even to companies that would like to support it in their products.

2. Status of this document

This document is a work-in-progress specification defining the Matroska file format as part of the IETF Cellar working group [3]. But since it's quite complete it is used as a reference for the development of libmatroska. Legacy versions of the specification can be found here [4] (PDF doc by Alexander Noe -- outdated).

For a simplified diagram of the layout of a Matroska file, see the Diagram page [5].

A more refined and detailed version of the EBML specifications is being worked on here [6].

The table found below is now generated from the "source" of the Matroska specification. This XML file [7] is also used to generate the semantic data used in libmatroska and libmatroska2. We encourage anyone to use and monitor its changes so your code is spec-proof and always up to date.

Note that versions 1, 2 and 3 have been finalized. Version 4 is currently work in progress. There MAY be further additions to v4.

3. Security Considerations

Matroska inherits security considerations from EBML. Other security considerations are to be determined.

4. IANA Considerations

To be determined.

5. Notations and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [8].

6. Basis in EBML

Matroska is a Document Type of EBML (Extensible Binary Meta Language). This specification is dependent on the EBML Specification [9]. For an understanding of Matroska's EBML Schema, see in particular the sections of the EBML Specification covering EBML Element Types [10], EBML Schema [11], and EBML Structure [12].

6.1. Added Constaints on EBML

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

- o The "docType" of the "EBML Header" MUST be 'matroska'.
- o The "EBMLMaxIDLength" of the "EBML Header" MUST be "4".
- o The "EBMLMaxSizeLength" of the "EBML Header" MUST be "8" or less.

6.2. Matroska Design

All top-levels elements (Segment and direct sub-elements) are coded on 4 octets, i.e. class D elements.

6.2.1. Language Codes

Language codes can be either the 3 letters bibliographic ISO-639-2 [13] form (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). Country codes are the same as used for internet domains [14].

6.2.2. Physical Types

Each level can have different meanings for audio and video. The ORIGINAL_MEDIUM tag can be used to specify a string for ChapterPhysicalEquiv = 60. Here is the list of possible levels for both audio and video :

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

6.2.3. Block Structure

Size = 1 + (1-8) + 4 + (4 + (4)) octets. So from 6 to 21 octets.

Bit 0 is the most significant bit.

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

There can be many Blocks in a BlockGroup provided they all have the same timecode. It is used with different parts of a frame with different priorities.

| Block Header | | Offset | Player | Description | | 0x00+ | MUST | Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits

set to increase the range). | | 0x01+ | MUST | Timecode (relative to Cluster timecode, signed int16) | | 0x03+ | - |

| Flags | | Bit | Player | Description | | 0-3 | - | Reserved, set to 0 | | 4 | - | Invisible, the codec SHOULD decode this frame but not display it | | 5-6 | MUST | Lacing

- 00 : no lacing
- 01 : Xiph lacing
- 11 : EBML lacing
- 10 : fixed-size lacing

| | 7 | - | not used |

| | Lace (when lacing bit is set) | | 0x00 | MUST | Number of frames in the lace-1 (uint8) | | 0x01 / 0xXX | MUST* | Lace-coded size of each frame of the lace, except for the last one (multiple uint8). *This is not used with Fixed-size lacing as it is calculated automatically from (total size of lace) / (number of frames in lace). | | (possibly) Laced Data | | 0x00 | MUST | Consecutive laced frames |

6.2.4. Lacing

Lacing is a mechanism to save space when storing data. It is typically used for small blocks of data (refered to as frames in matroska). There are 3 types of lacing : the Xiph one inspired by what is found in the Ogg container, the EBML one which is the same with sizes coded differently and the fixed-size one where the size is not coded. As an example is better than words...

Let’s say you want 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, you can store them in a lace to save space. They will then be solved in the same block as follows:

6.2.4.1. Xiph lacing

- Block head (with lacing bits set to 01)
- Lacing head: Number of frames in the lace -1, i.e. 2 (the 800 and 500 octets one)

- Lacing sizes: only the 2 first ones will be coded, 800 gives 255;255;255;35, 500 gives 255;245. The size of the last frame is deduced from the total size of the Block.

- Data in frame 1

- Data in frame 2

- Data in frame 3

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.

6.2.4.2. EBML lacing

In this case the size is not coded as blocks of 255 bytes, but as a difference with the previous size and this size is coded as in EBML. The first size in the lace is unsigned as in EBML. The others use a range shifting to get a sign on each value :

- Block head (with lacing bits set to 11)

- Lacing head: Number of frames in the lace -1, i.e. 2 (the 800 and 400 octets one)

- Lacing sizes: only the 2 first ones will be coded, 800 gives 0x320 0x4000 = 0x4320, 500 is coded as -300 : - 0x12C + 0x1FFF + 0x4000 = 0x5ED3. The size of the last frame is deduced from the total size of the Block.

- Data in frame 1

- Data in frame 2

- Data in frame 3

6.2.4.3. Fixed-size lacing

In this case only the number of frames in the lace is saved, the size of each frame is deduced from the total size of the Block. For example, for 3 frames of 800 octets each :

- Block head (with lacing bits set to 10)

- Lacing head: Number of frames in the lace -1, i.e. 2

- Data in frame 1

- o Data in frame 2
- o Data in frame 3

6.2.4.4. SimpleBlock Structure

The SimpleBlock is very inspired by the [Block structure](({{site.baseurl}}/index.html#block-structure). The main differences are the added Keyframe flag and Discardable flag. Otherwise everything is the same.

Size = 1 + (1-8) + 4 + (4 + (4)) octets. So from 6 to 21 octets.

Bit 0 is the most significant bit.

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

There can be many Blocks in a BlockGroup provided they all have the same timecode. It is used with different parts of a frame with different priorities.

| SimpleBlock Header | | Offset | Player | Description | | 0x00+ | MUST | Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits set to increase the range). | | 0x01+ | MUST | Timecode (relative to Cluster timecode, signed int16) | | 0x03+ | - |

| Flags | | Bit | Player | Description | | 0 | - | Keyframe, set when the Block contains only keyframes | | 1-3 | - | Reserved, set to 0 | | 4 | - | Invisible, the codec SHOULD decode this frame but not display it | | 5-6 | MUST | Lacing

- o 00 : no lacing
- o 01 : Xiph lacing
- o 11 : EBML lacing
- o 10 : fixed-size lacing

| | 7 | - | Discardable, the frames of the Block can be discarded during playing if needed |

| | Lace (when lacing bit is set) | | 0x00 | MUST | Number of frames in the lace-1 (uint8) | | 0x01 / 0xXX | MUST* | Lace-coded size of each frame of the lace, except for the last one (multiple uint8). *This is not used with Fixed-size lacing as it is calculated automatically from (total size of lace) / (number of frames in lace). | | (possibly) Laced Data | | 0x00 | MUST | Consecutive laced frames |

6.2.4.5. EncryptedBlock Structure

The EncryptedBlock is very inspired by the [SimpleBlock structure](({{site.baseurl}}/index.html#simpleblock_structure). The main differences is that the raw data are Transformed. That means the data after the lacing definition (if present) have been processed before put into the Block. The laced sizes apply on the decoded (Inverse Transform) data. This size of the Transformed data MAY NOT match the size of the initial chunk of data.

The other difference is that the number of frames in the lace are not saved if "no lacing" is specified (bits 5 and 6 set to 0).

The Transformation is specified by a TransformID in the Block (MUST be the same for all frames within the EncryptedBlock).

Size = 1 + (1-8) + 4 + (4 + (4)) octets. So from 6 to 21 octets.

Bit 0 is the most significant bit.

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

There can be many Blocks in a BlockGroup provided they all have the same timecode. It is used with different parts of a frame with different priorities.

| EncryptedBlock Header | | Offset | Player | Description | | 0x00+ | MUST | Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits set to increase the range). | | 0x01+ | MUST | Timecode (relative to Cluster timecode, signed int16) | | 0x03+ | - |

| Flags | | Bit | Player | Description | | 0 | - | Keyframe, set when the Block contains only keyframes | | 1-3 | - | Reserved, set to 0 | | 4 | - | Invisible, the codec SHOULD decode this frame but not display it | | 5-6 | MUST | Lacing

- o 00 : no lacing

- o 01 : Xiph lacing

- o 11 : EBML lacing

- o 10 : fixed-size lacing

| | 7 | - | Discardable, the frames of the Block can be discarded during playing if needed |

| | Lace (when lacing bit is set) | | 0x00 | MUST* | Number of frames in the lace-1 (uint8) _Only available if bit 5 or bit 6 of the EncryptedBlock flag is set to one. | | 0x01 / 0xXX | MUST_ | Lace-coded size of each frame of the lace, except for the last one (multiple uint8). *This is not used with Fixed-size lacing as it is calculated automatically from (total size of lace) / (number of frames in lace). | | (possibly) Laced Data | | 0x00 | MUST | TransformID (EBML coded integer value). Value 0 = Null Transform | | 0x01+ | MUST | Consecutive laced frames |

6.2.4.6. Virtual Block

The data in matroska is stored in coding order. But that means if you seek to a particular point and a frame has been referenced far away, you won’t know while playing and you might miss this frame (true for independent frames and overlapping of dependent frames). So the idea is to have a placeholder for the original frame in the timecode (display) order.

The structure is a scaled down version of the normal Block [15].

| Virtual Block Header | | Offset | Player | Description | | 0x00+ | MUST | Track Number (Track Entry). It is coded in EBML like form (1 octet if the value is < 0x80, 2 if < 0x4000, etc) (most significant bits set to increase the range). | | 0x01+ | MUST | Timecode (relative to Cluster timecode, signed int16) | | 0x03+ | - |

| Flags | | Bit | Player | Description | | 7-0 | - | Reserved, set to 0 |

|

7. Matroska Schema

This specification includes an "EBML Schema" which defines the Elements and structure of Matroska as an EBML Document Type. The

EBML Schema defines every valid Matroska element in a manner defined by the EBML specification.

For convenience the section of the EBML specification that defines EBML Schema Element Attributes is restated here:

7.1. EBML Schema Element Attributes

Within an EBML Schema the "<EBMLSchema>" uses the following attributes to define the EBML Schema:

attribute name	required	definition
docType	Yes	The "docType" lists the official name of the EBML Document Type that is defined by the EBML Schema; for example, "<EBMLSchema docType="matroska">".
version	Yes	The "version" lists an incremental non-negative integer that specifies the version of the docType documented by the EBML Schema. Unlike XML Schemas, an EBML Schema documents all versions of a docType’s definition rather than using separate EBML Schemas for each version of a docType. Elements may be introduced and deprecated by using the "minver" and "maxver" attributes of .

Within an EBML Schema the "<element>" uses the following attributes to define an EBML Element.

attribute name	required	definition
name	Yes	The official human-readable name of the EBML Element. The value of the name MUST be in the form of an NCName as defined by the XML Schema specification [16].
level	Yes	The level notes at what hierarchical depth the EBML Element may occur within an EBML Document. The Root Element of an EBML Document is at level 0 and the Elements that it may contain

		are at level 1. The level MUST be expressed as an integer. Note that Elements defined as "global" and "recursive" MAY occur at a level greater than or equal to the defined "level".
global	No	A boolean to express if an EBML Element MUST occur at its defined level or may occur within any Parent EBML Element. If the "global" attribute is not expressed for that Element then that element is to be considered not global.
id	Yes	The Element ID expressed in hexadecimal notation prefixed by a '0x'. To reduce the risk of false positives while parsing EBML Streams, the IDs of the Root Element and Top-Level Elements SHOULD be at least 4 octets in length. Element IDs defined for use at Level 0 or Level 1 MAY use shorter octet lengths to facilitate padding and optimize edits to EBML Documents; for instance, the EBML Void Element uses an Element ID with a one octet length to allow its usage in more writing and editing scenarios.
minOccurs	No	An integer to express the minimal number of occurrences that the EBML Element MUST occur within its Parent Element if its Parent Element is used. If the Element has no Parent Level (as is the case with Elements at Level 0), then minOccurs refers to constaints on the Element's occurrence within the EBML Document. If the minOccurs attribute is not expressed for that Element then that Element shall be considered to have a minOccurs value of 0. This value of minOccurs MUST be a positive integer. The semantic meaning of

		minOccurs within an EBML Schema is considered analogous to the meaning of minOccurs within an XML Schema [17]. Note that Elements with minOccurs set to "1" that also have a default value declared are not required to be stored but are required to be interpretted, see the "Note on the Use of default attributes to define Mandatory EBML Elements".
maxOccurs	No	A value to express the maximum number of occurrences that the EBML Element MAY occur within its Parent Element if its Parent Element is used. If the Element has no Parent Level (as is the case with Elements at Level 0), then maxOccurs refers to constaints on the Element's occurrence within the EBML Document. This value may be either a positive integer or the term "unbounded" to indicate there is no maximum number of occurrences or the term "identical" to indicate that the Element is an "Identically Recurring Element". If the maxOccurs attribute is not expressed for that Element then that Element shall be considered to have a maxOccurs value of 1. The semantic meaning of maxOccurs within an EBML Schema is considered analogous to the meaning of minOccurs within an XML Schema [18], with EBML Schema adding the concept of Identically Recurring Elements.
range	No	For Elements which are of numerical types (Unsigned Integer, Signed Integer, Float, and Date) a numerical range may be specified. If specified that the value of the EBML Element MUST be within the defined range inclusively. See the "section of

		Expressions of range" for rules applied to expression of range values.
default	No	A default value may be provided. If an Element is mandatory but not written within its Parent EBML Element, then the parser of the EBML Document MUST insert the defined default value of the Element. EBML Elements that are Master-elements MUST NOT declare a default value.
type	Yes	As defined within the "section on EBML Element Types", the type MUST be set to one of the following values: 'integer' (signed integer), 'uinteger' (unsigned integer), 'float', 'string', 'date', 'utf-8', 'master', or 'binary'.
unknownsizeallowed	No	A boolean to express if an EBML Element MAY be used as an "Unknown-Sized Element" (having all VINT_DATA bits of Element Data Size set to 1). The "unknownsizeallowed" attribute only applies to Master-elements. If the "unknownsizeallowed" attribute is not used it is assumed that the element is not allowed to use an unknown Element Data Size.
recursive	No	A boolean to express if an EBML Element MAY be stored recursively. In this case the Element MAY be stored at levels greater that defined in the "level" attribute if the Element is a Child Element of a Parent Element with the same Element ID. The "recursive" attribute only applies to Master-elements. If the "recursive" attribute is not used it is assumed that the element is not allowed to be used recursively.
minver	No	The "minver" (minimum version) attribute stores a non-negative

		integer that represents the first version of the docType to support the element. If the "minver" attribute is not used it is assumed that the element has a minimum version of "1".
maxver	No	The "maxver" (maximum version) attribute stores a non-negative integer that represents the last or most recent version of the docType to support the element. If the "maxver" attribute is not used it is assumed that the element has a maximum version equal to the value stored in the "version" attribute of .

The "<element>" nodes shall contain a description of the meaning and use of the EBML Element stored within one or many "<documentation>" sub-elements. The "<documentation>" sub-element may use a "lang" attribute which may be set to the RFC 5646 value of the language of the element's documentation. The "<documentation>" sub-element may use a "type" attribute to distinguish the meaning of the documentation. Recommended values for the "<documentation>" sub-element's "type" attribute include: "definition", "rationale", "usage notes", and "references".

The "<element>" nodes MUST be arranged hierarchically according to the permitted structure of the EBML Document Type. An "<element>" node that defines an EBML Element which is a Child Element of another Parent Element MUST be stored as an immediate sub-element of the "<element>" node that defines the Parent Element. "<element>" nodes that define Level 0 Elements and Global Elements should be sub-elements of "<EBMLSchema>".

7.2. Matroska Additions to Schema Element Attributes

In addition to the EBML Schema definition provided by the EBML Specification, Matroska adds the following additional attributes:

attribute name	required	definition
webm	No	A boolean to express if the Matroska Element is also supported within version 2 of the "webm" specification. Please consider the webm specification [19] as the authoritative on "webm".

7.3. Matroska Schema

Here the definition of each Matroska Element is provided.

% concatenate with Matroska EBML Schema converted to markdown %

8. Segment

Element Name	Segment
Element ID	0x18538067
Element Type	master
Version	1-4
Parent Element	None
Child Elements	Section 9 Section 13 Section 31 Section 62 Section 172 Section 187 Section 197 Section 225
Element Context	None
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The Root Element that contains all other Top-Level Elements (Elements defined only at Level 1). A Matroska file is composed of 1 Segment.

9. SeekHead

Element Name	SeekHead
Element ID	0x114D9B74
Element Type	master
Version	1-4
Parent Element	Section 8
Child Elements	Section 10
Element Context	/Section 8/SeekHead
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains the position of other Top-Level Elements.

10. Seek

Element Name	Seek
Element ID	0x4DBB
Element Type	master
Version	1-4
Parent Element	Section 9
Child Elements	Section 11 Section 12
Element Context	/Section 8/Section 9/Seek
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains a single seek entry to an EBML Element.

11. SeekID

Element Name	SeekID
Element ID	0x53AB
Element Type	binary
Version	1-4
Parent Element	Section 10
Child Elements	None
Element Context	/Section 8/Section 9/Section 10/SeekID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The binary ID corresponding to the Element name.

12. SeekPosition

Element Name	SeekPosition
Element ID	0x53AC
Element Type	uinteger
Version	1-4
Parent Element	Section 10
Child Elements	None
Element Context	/Section 8/Section 9/Section 10/SeekPosition
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The position of the Element in the Segment in octets (0 = first level 1 Element).

13. Info

Element Name	Info
Element ID	0x1549A966
Element Type	master
Version	1-4
Parent Element	Section 8
Child Elements	Section 14 Section 15 Section 16 Section 17 Section 18 Section 19 Section 20 Section 21 Section 25 Section 26 Section 27 Section 28 Section 29 Section 30
Element Context	/Section 8/Info
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains general information about the Segment.

14. SegmentUID

Element Name	SegmentUID
Element ID	0x73A4
Element Type	binary
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/SegmentUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A randomly generated unique ID to identify the Segment amongst many others (128 bits).

15. SegmentFilename

Element Name	SegmentFilename
Element ID	0x7384
Element Type	utf-8
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/SegmentFilename
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A filename corresponding to this Segment.

16. PrevUID

Element Name	PrevUID
Element ID	0x3CB923
Element Type	binary
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/PrevUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A unique ID to identify the previous Segment of a Linked Segment (128 bits).

17. PrevFilename

Element Name	PrevFilename
Element ID	0x3C83AB
Element Type	utf-8
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/PrevFilename
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A filename corresponding to the file of the previous Linked Segment.

18. NextUID

Element Name	NextUID
Element ID	0x3EB923
Element Type	binary
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/NextUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A unique ID to identify the next Segment of a Linked Segment (128 bits).

19. NextFilename

Element Name	NextFilename
Element ID	0x3E83BB
Element Type	utf-8
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/NextFilename
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A filename corresponding to the file of the next Linked Segment.

20. SegmentFamily

Element Name	SegmentFamily
Element ID	0x4444
Element Type	binary
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/SegmentFamily
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	A randomly generated unique ID that all Segments of a Linked Segment MUST share (128 bits).

21. ChapterTranslate

Element Name	ChapterTranslate
Element ID	0x6924
Element Type	master
Version	1-4
Parent Element	Section 13
Child Elements	Section 22 Section 23 Section 24
Element Context	/Section 8/Section 13/ChapterTranslate
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	A tuple of corresponding ID used by chapter codecs to represent this Segment.

22. ChapterTranslateEditionUID

Element Name	ChapterTranslateEditionUID
Element ID	0x69FC
Element Type	uinteger
Version	1-4
Parent Element	Section 21
Child Elements	None
Element Context	/Section 8/Section 13/Section 21/ChapterTranslateEditionUID
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Specify an edition UID on which this correspondance applies. When not specified, it means for all editions found in the Segment.

23. ChapterTranslateCodec

Element Name	ChapterTranslateCodec
Element ID	0x69BF
Element Type	uinteger
Version	1-4
Parent Element	Section 21
Child Elements	None
Element Context	/Section 8/Section 13/Section 21/ChapterTranslateCodec
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The chapter codec using this ID (0: Matroska Script, 1: DVD-menu).

24. ChapterTranslateID

Element Name	ChapterTranslateID
Element ID	0x69A5
Element Type	binary
Version	1-4
Parent Element	Section 21
Child Elements	None
Element Context	/Section 8/Section 13/Section 21/ChapterTranslateID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The binary value used to represent this Segment in the chapter codec data. The format depends on the ChapProcessCodecID used.

25. TimecodeScale

Element Name	TimecodeScale
Element ID	0x2AD7B1
Element Type	uinteger
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/TimecodeScale
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Timestamp scale in nanoseconds (1.000.000 means all timestamps in the Segment are expressed in milliseconds).

26. Duration

Element Name	Duration
Element ID	0x4489
Element Type	float
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/Duration
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Duration of the Segment in nanoseconds based on TimecodeScale.

27. DateUTC

Element Name	DateUTC
Element ID	0x4461
Element Type	date
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/DateUTC
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The date and time that the Segment was created by the muxing application or library.

28. Title

Element Name	Title
Element ID	0x7BA9
Element Type	utf-8
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/Title
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	General name of the Segment.

29. MuxingApp

Element Name	MuxingApp
Element ID	0x4D80
Element Type	utf-8
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/MuxingApp
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Muxing application or library (example: "libmatroska-0.4.3").

30. WritingApp

Element Name	WritingApp
Element ID	0x5741
Element Type	utf-8
Version	1-4
Parent Element	Section 13
Child Elements	None
Element Context	/Section 8/Section 13/WritingApp
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Writing application (example: "mkvmerge-0.3.3").

31. Cluster

Element Name	Cluster
Element ID	0x1F43B675
Element Type	master
Version	1-4
Parent Element	Section 8
Child Elements	Section 32 Section 33 Section 35 Section 36 Section 37 Section 38 Section 61
Element Context	/Section 8/Cluster
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	The Top-Level Element containing the (monolithic) Block structure.

32. Timecode

Element Name	Timecode
Element ID	0xE7
Element Type	uinteger
Version	1-4
Parent Element	Section 31
Child Elements	None
Element Context	/Section 8/Section 31/Timecode
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Absolute timestamp of the cluster (based on TimecodeScale).

33. SilentTracks

Element Name	SilentTracks
Element ID	0x5854
Element Type	master
Version	1-4
Parent Element	Section 31
Child Elements	Section 34
Element Context	/Section 8/Section 31/SilentTracks
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
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.

34. SilentTrackNumber

Element Name	SilentTrackNumber
Element ID	0x58D7
Element Type	uinteger
Version	1-4
Parent Element	Section 33
Child Elements	None
Element Context	/Section 8/Section 31/Section 33/SilentTrackNumber
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
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.

35. Position

Element Name	Position
Element ID	0xA7
Element Type	uinteger
Version	1-4
Parent Element	Section 31
Child Elements	None
Element Context	/Section 8/Section 31/Position
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The Position of the Cluster in the Segment (0 in live broadcast streams). It might help to resynchronise offset on damaged streams.

36. PrevSize

Element Name	PrevSize
Element ID	0xAB
Element Type	uinteger
Version	1-4
Parent Element	Section 31
Child Elements	None
Element Context	/Section 8/Section 31/PrevSize
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Size of the previous Cluster, in octets. Can be useful for backward playing.

37. SimpleBlock

Element Name	SimpleBlock
Element ID	0xA3
Element Type	binary
Version	2-4
Parent Element	Section 31
Child Elements	None
Element Context	/Section 8/Section 31/SimpleBlock
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Similar to Block but without all the extra information, mostly used to reduced overhead when no extra feature is needed. (see SimpleBlock Structure)

38. BlockGroup

Element Name	BlockGroup
Element ID	0xA0
Element Type	master
Version	1-4
Parent Element	Section 31
Child Elements	Section 39 Section 40 Section 41 Section 45 Section 46 Section 47 Section 48 Section 49 Section 50 Section 51 Section 58
Element Context	/Section 8/Section 31/BlockGroup
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Basic container of information containing a single Block or BlockVirtual, and information specific to that Block/VirtualBlock.

39. Block

Element Name	Block
Element ID	0xA1
Element Type	binary
Version	1-4
Parent Element	Section 38
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Block
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Block containing the actual data to be rendered and a timestamp relative to the Cluster Timecode. (see Block Structure)

40. BlockVirtual

Element Name	BlockVirtual
Element ID	0xA2
Element Type	binary
Version	1-4
Parent Element	Section 38
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/BlockVirtual
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A Block with no data. It MUST be stored in the stream at the place the real Block would be in display order. (see Block Virtual)

41. BlockAdditions

Element Name	BlockAdditions
Element ID	0x75A1
Element Type	master
Version	1-4
Parent Element	Section 38
Child Elements	Section 42
Element Context	/Section 8/Section 31/Section 38/BlockAdditions
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Contain additional blocks to complete the main one. An EBML parser that has no knowledge of the Block structure could still see and use/skip these data.

42. BlockMore

Element Name	BlockMore
Element ID	0xA6
Element Type	master
Version	1-4
Parent Element	Section 41
Child Elements	Section 43 Section 44
Element Context	/Section 8/Section 31/Section 38/Section 41/BlockMore
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contain the BlockAdditional and some parameters.

43. BlockAddID

Element Name	BlockAddID
Element ID	0xEE
Element Type	uinteger
Version	1-4
Parent Element	Section 42
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Section 41/Section 42/BlockAddID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	An ID to identify the BlockAdditional level.

44. BlockAdditional

Element Name	BlockAdditional
Element ID	0xA5
Element Type	binary
Version	1-4
Parent Element	Section 42
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Section 41/Section 42/BlockAdditional
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Interpreted by the codec as it wishes (using the BlockAddID).

45. BlockDuration

Element Name	BlockDuration
Element ID	0x9B
Element Type	uinteger
Version	1-4
Parent Element	Section 38
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/BlockDuration
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The duration of the Block (based on TimecodeScale). This Element is mandatory when DefaultDuration is set for the track (but can be omitted as other default values). 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). This Element can be useful at the end of a Track (as there is not other Block available), or when there is a break in a track like for subtitle tracks. When set to 0 that means the frame is not a keyframe.

46. ReferencePriority

Element Name	ReferencePriority
Element ID	0xFA
Element Type	uinteger
Version	1-4
Parent Element	Section 38
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/ReferencePriority
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	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.

47. ReferenceBlock

Element Name	ReferenceBlock
Element ID	0xFB
Element Type	integer
Version	1-4
Parent Element	Section 38
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/ReferenceBlock
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Timestamp of another frame used as a reference (ie: B or P frame). The timestamp is relative to the block it’s attached to.

48. ReferenceVirtual

Element Name	ReferenceVirtual
Element ID	0xFD
Element Type	integer
Version	1-4
Parent Element	Section 38
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/ReferenceVirtual
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Relative position of the data that would otherwise be in position of the virtual block.

49. CodecState

Element Name	CodecState
Element ID	0xA4
Element Type	binary
Version	2-4
Parent Element	Section 38
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/CodecState
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The new codec state to use. Data interpretation is private to the codec. This information SHOULD always be referenced by a seek entry.

50. DiscardPadding

Element Name	DiscardPadding
Element ID	0x75A2
Element Type	integer
Version	4-4
Parent Element	Section 38
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/DiscardPadding
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Duration in nanoseconds of the silent data added to the Block (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.

51. Slices

Element Name	Slices
Element ID	0x8E
Element Type	master
Version	1-4
Parent Element	Section 38
Child Elements	Section 52
Element Context	/Section 8/Section 31/Section 38/Slices
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Contains slices description.

52. TimeSlice

Element Name	TimeSlice
Element ID	0xE8
Element Type	master
Version	1-4
Parent Element	Section 51
Child Elements	Section 53 Section 54 Section 55 Section 56 Section 57
Element Context	/Section 8/Section 31/Section 38/Section 51/TimeSlice
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains extra time information about the data contained in the Block. While there are a few files in the wild with this Element, it is no longer in use and has been deprecated. Being able to interpret this Element is not REQUIRED for playback.

53. LaceNumber

Element Name	LaceNumber
Element ID	0xCC
Element Type	uinteger
Version	1-4
Parent Element	Section 52
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Section 51/Section 52/LaceNumber
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The reverse number of the frame in the lace (0 is the last frame, 1 is the next to last, etc). While there are a few files in the wild with this Element, it is no longer in use and has been deprecated. Being able to interpret this Element is not REQUIRED for playback.

54. FrameNumber

Element Name	FrameNumber
Element ID	0xCD
Element Type	uinteger
Version	1-4
Parent Element	Section 52
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Section 51/Section 52/FrameNumber
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
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).

55. BlockAdditionID

Element Name	BlockAdditionID
Element ID	0xCB
Element Type	uinteger
Version	1-4
Parent Element	Section 52
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Section 51/Section 52/BlockAdditionID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The ID of the BlockAdditional Element (0 is the main Block).

56. Delay

Element Name	Delay
Element ID	0xCE
Element Type	uinteger
Version	1-4
Parent Element	Section 52
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Section 51/Section 52/Delay
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The (scaled) delay to apply to the Element.

57. SliceDuration

Element Name	SliceDuration
Element ID	0xCF
Element Type	uinteger
Version	1-4
Parent Element	Section 52
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Section 51/Section 52/SliceDuration
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The (scaled) duration to apply to the Element.

58. ReferenceFrame

Element Name	ReferenceFrame
Element ID	0xC8
Element Type	master
Version	0-4
Parent Element	Section 38
Child Elements	Section 59 Section 60
Element Context	/Section 8/Section 31/Section 38/ReferenceFrame
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX trick track extenstions

59. ReferenceOffset

Element Name	ReferenceOffset
Element ID	0xC9
Element Type	uinteger
Version	0-4
Parent Element	Section 58
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Section 58/ReferenceOffset
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX trick track extenstions

60. ReferenceTimeCode

Element Name	ReferenceTimeCode
Element ID	0xCA
Element Type	uinteger
Version	0-4
Parent Element	Section 58
Child Elements	None
Element Context	/Section 8/Section 31/Section 38/Section 58/ReferenceTimeCode
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX trick track extenstions

61. EncryptedBlock

Element Name	EncryptedBlock
Element ID	0xAF
Element Type	binary
Version	1-4
Parent Element	Section 31
Child Elements	None
Element Context	/Section 8/Section 31/EncryptedBlock
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Similar to SimpleBlock but the data inside the Block are Transformed (encrypt and/or signed). (see EncryptedBlock Structure)

62. Tracks

Element Name	Tracks
Element ID	0x1654AE6B
Element Type	master
Version	1-4
Parent Element	Section 8
Child Elements	Section 63
Element Context	/Section 8/Tracks
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	A Top-Level Element of information with many tracks described.

63. TrackEntry

Element Name	TrackEntry
Element ID	0xAE
Element Type	master
Version	1-4
Parent Element	Section 62
Child Elements	Section 64 Section 65 Section 66 Section 67 Section 68 Section 69 Section 70 Section 71 Section 72 Section 73 Section 74 Section 75 Section 76 Section 77 Section 78 Section 79 Section 80 Section 81 Section 82 Section 83 Section 84 Section 85 Section 86 Section 87 Section 88 Section 89 Section 90 Section 91 Section 95 Section 139 Section 145 Section 152 Section 153 Section 154 Section 155 Section 156 Section 157
Element Context	/Section 8/Section 62/TrackEntry
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Describes a track with all Elements.

64. TrackNumber

Element Name	TrackNumber
Element ID	0xD7
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrackNumber
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The track number as used in the Block Header (using more than 127 tracks is not encouraged, though the design allows an unlimited number).

65. TrackUID

Element Name	TrackUID
Element ID	0x73C5
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrackUID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A unique ID to identify the Track. This SHOULD be kept the same when making a direct stream copy of the Track to another file.

66. TrackType

Element Name	TrackType
Element ID	0x83
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrackType
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A set of track types coded on 8 bits (1: video, 2: audio, 3: complex, 0x10: logo, 0x11: subtitle, 0x12: buttons, 0x20: control).

67. FlagEnabled

Element Name	FlagEnabled
Element ID	0xB9
Element Type	uinteger
Version	2-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/FlagEnabled
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Set if the track is usable. (1 bit)

68. FlagDefault

Element Name	FlagDefault
Element ID	0x88
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/FlagDefault
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Set if that track (audio, video or subs) SHOULD be active if no language found matches the user preference. (1 bit)

69. FlagForced

Element Name	FlagForced
Element ID	0x55AA
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/FlagForced
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Set if that track MUST be active during playback. There can be many forced track for a kind (audio, video or subs), the player SHOULD select the one which language matches the user preference or the default + forced track. Overlay MAY happen between a forced and non-forced track of the same kind. (1 bit)

70. FlagLacing

Element Name	FlagLacing
Element ID	0x9C
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/FlagLacing
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Set if the track MAY contain blocks using lacing. (1 bit)

71. MinCache

Element Name	MinCache
Element ID	0x6DE7
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/MinCache
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
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.

72. MaxCache

Element Name	MaxCache
Element ID	0x6DF8
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/MaxCache
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The maximum cache size necessary to store referenced frames in and the current frame. 0 means no cache is needed.

73. DefaultDuration

Element Name	DefaultDuration
Element ID	0x23E383
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/DefaultDuration
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Number of nanoseconds (not scaled via TimecodeScale) per frame ('frame' in the Matroska sense -- one Element put into a (Simple)Block).

74. DefaultDecodedFieldDuration

Element Name	DefaultDecodedFieldDuration
Element ID	0x234E7A
Element Type	uinteger
Version	4-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/DefaultDecodedFieldDuration
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The period in nanoseconds (not scaled by TimcodeScale)

between two successive fields at the output of the decoding process (see the notes)

75. TrackTimecodeScale

Element Name	TrackTimecodeScale
Element ID	0x23314F
Element Type	float
Version	1-3 DEPRECATED
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrackTimecodeScale
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DEPRECATED, DO NOT USE. 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).

76. TrackOffset

Element Name	TrackOffset
Element ID	0x537F
Element Type	integer
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrackOffset
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A value to add to the Block's Timestamp. This can be used to adjust the playback offset of a track.

77. MaxBlockAdditionID

Element Name	MaxBlockAdditionID
Element ID	0x55EE
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/MaxBlockAdditionID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The maximum value of BlockAddID. A value 0 means there is no BlockAdditions for this track.

78. Name

Element Name	Name
Element ID	0x536E
Element Type	utf-8
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Name
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A human-readable track name.

79. Language

Element Name	Language
Element ID	0x22B59C
Element Type	string
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Language
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Specifies the language of the track in the Matroska languages form.

80. CodecID

Element Name	CodecID
Element ID	0x86
Element Type	string
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/CodecID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	An ID corresponding to the codec, see the codec page for more info.

81. CodecPrivate

Element Name	CodecPrivate
Element ID	0x63A2
Element Type	binary
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/CodecPrivate
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Private data only known to the codec.

82. CodecName

Element Name	CodecName
Element ID	0x258688
Element Type	utf-8
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/CodecName
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A human-readable string specifying the codec.

83. AttachmentLink

Element Name	AttachmentLink
Element ID	0x7446
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/AttachmentLink
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The UID of an attachment that is used by this codec.

84. CodecSettings

Element Name	CodecSettings
Element ID	0x3A9697
Element Type	utf-8
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/CodecSettings
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A string describing the encoding setting used.

85. CodecInfoURL

Element Name	CodecInfoURL
Element ID	0x3B4040
Element Type	string
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/CodecInfoURL
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	A URL to find information about the codec used.

86. CodecDownloadURL

Element Name	CodecDownloadURL
Element ID	0x26B240
Element Type	string
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/CodecDownloadURL
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	A URL to download about the codec used.

87. CodecDecodeAll

Element Name	CodecDecodeAll
Element ID	0xAA
Element Type	uinteger
Version	2-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/CodecDecodeAll
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The codec can decode potentially damaged data (1 bit).

88. TrackOverlay

Element Name	TrackOverlay
Element ID	0x6FAB
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrackOverlay
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
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 (see 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.

89. CodecDelay

Element Name	CodecDelay
Element ID	0x56AA
Element Type	uinteger
Version	4-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/CodecDelay
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	CodecDelay is The codec-built-in delay in nanoseconds. This value MUST be subtracted from each block timestamp in order to get the actual timestamp. The value SHOULD be small so the muxing of tracks with the same actual timestamp are in the same Cluster.

90. SeekPreRoll

Element Name	SeekPreRoll
Element ID	0x56BB
Element Type	uinteger
Version	4-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/SeekPreRoll
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	After a discontinuity, SeekPreRoll is the duration in nanoseconds of the data the decoder MUST decode before the decoded data is valid.

91. TrackTranslate

Element Name	TrackTranslate
Element ID	0x6624
Element Type	master
Version	1-4
Parent Element	Section 63
Child Elements	Section 92 Section 93 Section 94
Element Context	/Section 8/Section 62/Section 63/TrackTranslate
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	The track identification for the given Chapter Codec.

92. TrackTranslateEditionUID

Element Name	TrackTranslateEditionUID
Element ID	0x66FC
Element Type	uinteger
Version	1-4
Parent Element	Section 91
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 91/TrackTranslateEditionUID
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Specify an edition UID on which this translation applies. When not specified, it means for all editions found in the Segment.

93. TrackTranslateCodec

Element Name	TrackTranslateCodec
Element ID	0x66BF
Element Type	uinteger
Version	1-4
Parent Element	Section 91
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 91/TrackTranslateCodec
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The chapter codec using this ID (0: Matroska Script, 1: DVD-menu).

94. TrackTranslateTrackID

Element Name	TrackTranslateTrackID
Element ID	0x66A5
Element Type	binary
Version	1-4
Parent Element	Section 91
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 91/TrackTranslateTrackID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The binary value used to represent this track in the chapter codec data. The format depends on the ChapProcessCodecID used.

95. Video

Element Name	Video
Element ID	0xE0
Element Type	master
Version	1-4
Parent Element	Section 63
Child Elements	Section 96 Section 97 Section 98 Section 99 Section 100 Section 101 Section 102 Section 103 Section 104 Section 105 Section 106 Section 107 Section 108 Section 109 Section 110 Section 111 Section 112 Section 113 Section 114
Element Context	/Section 8/Section 62/Section 63/Video
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Video settings.

96. FlagInterlaced

Element Name	FlagInterlaced
Element ID	0x9A
Element Type	uinteger
Version	2-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/FlagInterlaced
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A flag to declare is the video is known to be progressive or interlaced and if applicable to declare details about the interlacement. (0: undetermined, 1: interlaced, 2: progressive)

97. FieldOrder

Element Name	FieldOrder
Element ID	0x9D
Element Type	uinteger
Version	4-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/FieldOrder
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Declare the field ordering of the video. If FlagInterlaced is not set to 1, this Element MUST be ignored. (0: Progressive, 1: Interlaced with top field display first and top field stored first, 2: Undetermined field order, 6: Interlaced with bottom field displayed first and bottom field stored first, 9: Interlaced with bottom field displayed first and top field stored first, 14: Interlaced with top field displayed first and bottom field stored first)

98. StereoMode

Element Name	StereoMode
Element ID	0x53B8
Element Type	uinteger
Version	3-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/StereoMode
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Stereo-3D video mode (0: mono, 1: side by side (left eye is first), 2: top-bottom (right eye is first), 3: top-bottom (left eye is first), 4: checkboard (right is first), 5: checkboard (left is first), 6: row interleaved (right is first), 7: row interleaved (left is first), 8: column interleaved (right is first), 9: column interleaved (left is first), 10: anaglyph (cyan/red), 11: side by side (right eye is first), 12: anaglyph (green/magenta), 13 both eyes laced in one Block (left eye is first), 14 both eyes laced in one Block (right eye is first)) . There are some more details on 3D support in the Specification Notes.

99. AlphaMode

Element Name	AlphaMode
Element ID	0x53C0
Element Type	uinteger
Version	3-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/AlphaMode
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Alpha Video Mode. Presence of this Element indicates that the BlockAdditional Element could contain Alpha data.

100. OldStereoMode

Element Name	OldStereoMode
Element ID	0x53B9
Element Type	uinteger
Version	1-0 DEPRECATED
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/OldStereoMode
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DEPRECATED, DO NOT USE. Bogus StereoMode value used in old versions of libmatroska. (0: mono, 1: right eye, 2: left eye, 3: both eyes).

101. PixelWidth

Element Name	PixelWidth
Element ID	0xB0
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/PixelWidth
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Width of the encoded video frames in pixels.

102. PixelHeight

Element Name	PixelHeight
Element ID	0xBA
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/PixelHeight
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Height of the encoded video frames in pixels.

103. PixelCropBottom

Element Name	PixelCropBottom
Element ID	0x54AA
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/PixelCropBottom
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The number of video pixels to remove at the bottom of the image (for HDTV content).

104. PixelCropTop

Element Name	PixelCropTop
Element ID	0x54BB
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/PixelCropTop
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The number of video pixels to remove at the top of the image.

105. PixelCropLeft

Element Name	PixelCropLeft
Element ID	0x54CC
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/PixelCropLeft
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The number of video pixels to remove on the left of the image.

106. PixelCropRight

Element Name	PixelCropRight
Element ID	0x54DD
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/PixelCropRight
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The number of video pixels to remove on the right of the image.

107. DisplayWidth

Element Name	DisplayWidth
Element ID	0x54B0
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/DisplayWidth
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Width of the video frames to display. Applies to the video frame after cropping (PixelCrop* Elements). The default value is only valid when DisplayUnit is 0.

108. DisplayHeight

Element Name	DisplayHeight
Element ID	0x54BA
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/DisplayHeight
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Height of the video frames to display. Applies to the video frame after cropping (PixelCrop* Elements). The default value is only valid when DisplayUnit is 0.

109. DisplayUnit

Element Name	DisplayUnit
Element ID	0x54B2
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/DisplayUnit
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	How DisplayWidth & DisplayHeight are interpreted (0: pixels, 1: centimeters, 2: inches, 3: Display Aspect Ratio).

110. AspectRatioType

Element Name	AspectRatioType
Element ID	0x54B3
Element Type	uinteger
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/AspectRatioType
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Specify the possible modifications to the aspect ratio (0: free resizing, 1: keep aspect ratio, 2: fixed).

111. ColourSpace

Element Name	ColourSpace
Element ID	0x2EB524
Element Type	binary
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/ColourSpace
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Same value as in AVI (32 bits).

112. GammaValue

Element Name	GammaValue
Element ID	0x2FB523
Element Type	float
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/GammaValue
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Gamma Value.

113. FrameRate

Element Name	FrameRate
Element ID	0x2383E3
Element Type	float
Version	1-4
Parent Element	Section 95
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/FrameRate
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Number of frames per second. Informational only.

114. Colour

Element Name	Colour
Element ID	0x55B0
Element Type	master
Version	4-4
Parent Element	Section 95
Child Elements	Section 115 Section 116 Section 117 Section 118 Section 119 Section 120 Section 121 Section 122 Section 123 Section 124 Section 125 Section 126 Section 127 Section 128
Element Context	/Section 8/Section 62/Section 63/Section 95/Colour
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Settings describing the colour format.

115. MatrixCoefficients

Element Name	MatrixCoefficients
Element ID	0x55B1
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/MatrixCoefficients
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The Matrix Coefficients of the video used to derive luma and chroma values from reg, green, and blue color primaries. For clarity, the value and meanings for MatrixCoefficients are adopted from Table 4 of ISO/IEC 23001-8:2013/DCOR1. (0:GBR, 1: BT709, 2: Unspecified, 3: Reserved, 4: FCC, 5: BT470BG, 6: SMPTE 170M, 7: SMPTE 240M, 8: YCOCG, 9: BT2020 Non-constant Luminance, 10: BT2020 Constant Luminance)

116. BitsPerChannel

Element Name	BitsPerChannel
Element ID	0x55B2
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/BitsPerChannel
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Number of decoded bits per channel. A value of 0 indicates that the BitsPerChannel is unspecified.

117. ChromaSubsamplingHorz

Element Name	ChromaSubsamplingHorz
Element ID	0x55B3
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/ChromaSubsamplingHorz
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The amount of pixels to remove in the Cr and Cb channels for every pixel not removed horizontally. Example: For video with 4:2:0 chroma subsampling, the ChromaSubsamplingHorz SHOULD be set to 1.

118. ChromaSubsamplingVert

Element Name	ChromaSubsamplingVert
Element ID	0x55B4
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/ChromaSubsamplingVert
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The amount of pixels to remove in the Cr and Cb channels for every pixel not removed vertically. Example: For video with 4:2:0 chroma subsampling, the ChromaSubsamplingVert SHOULD be set to 1.

119. CbSubsamplingHorz

Element Name	CbSubsamplingHorz
Element ID	0x55B5
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/CbSubsamplingHorz
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The amount of pixels to remove in the Cb channel for every pixel not removed horizontally. This is additive with ChromaSubsamplingHorz. Example: For video with 4:2:1 chroma subsampling, the ChromaSubsamplingHorz SHOULD be set to 1 and CbSubsamplingHorz SHOULD be set to 1.

120. CbSubsamplingVert

Element Name	CbSubsamplingVert
Element ID	0x55B6
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/CbSubsamplingVert
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The amount of pixels to remove in the Cb channel for every pixel not removed vertically. This is additive with ChromaSubsamplingVert.

121. ChromaSitingHorz

Element Name	ChromaSitingHorz
Element ID	0x55B7
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/ChromaSitingHorz
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	How chroma is subsampled horizontally. (0: Unspecified, 1: Left Collocated, 2: Half)

122. ChromaSitingVert

Element Name	ChromaSitingVert
Element ID	0x55B8
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/ChromaSitingVert
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	How chroma is subsampled vertically. (0: Unspecified, 1: Top Collocated, 2: Half)

123. Range

Element Name	Range
Element ID	0x55B9
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Range
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Clipping of the color ranges. (0: Unspecified, 1: Broadcast Range, 2: Full range (no clipping), 3: Defined by MatrixCoefficients/TransferCharacteristics)

124. TransferCharacteristics

Element Name	TransferCharacteristics
Element ID	0x55BA
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/TransferCharacteristics
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The transfer characteristics of the video. For clarity, the value and meanings for TransferCharacteristics 1-15 are adopted from Table 3 of ISO/IEC 23001-8:2013/DCOR1. TransferCharacteristics 16-18 are proposed values. (0: Reserved, 1: ITU-R BT.709, 2: Unspecified, 3: Reserved, 4: Gamma 2.2 curve, 5: Gamma 2.8 curve, 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: SMPTE ST 2084, 17: SMPTE ST 428-1 18: ARIB STD-B67 (HLG))

125. Primaries

Element Name	Primaries
Element ID	0x55BB
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Primaries
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The colour primaries of the video. For clarity, the value and meanings for Primaries are adopted from Table 2 of ISO/IEC 23001-8:2013/DCOR1. (0: Reserved, 1: ITU-R BT.709, 2: Unspecified, 3: Reserved, 4: ITU-R BT.470M, 5: ITU-R BT.470BG, 6: SMPTE 170M, 7: SMPTE 240M, 8: FILM, 9: ITU-R BT.2020, 10: SMPTE ST 428-1, 22: JEDEC P22 phosphors)

126. MaxCLL

Element Name	MaxCLL
Element ID	0x55BC
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/MaxCLL
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Maximum brightness of a single pixel (Maximum Content Light Level) in candelas per square meter (cd/m^2).

127. MaxFALL

Element Name	MaxFALL
Element ID	0x55BD
Element Type	uinteger
Version	4-4
Parent Element	Section 114
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/MaxFALL
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Maximum brightness of a single full frame (Maximum Frame-Average Light Level) in candelas per square meter (cd/m^2).

128. MasteringMetadata

Element Name	MasteringMetadata
Element ID	0x55D0
Element Type	master
Version	4-4
Parent Element	Section 114
Child Elements	Section 129 Section 130 Section 131 Section 132 Section 133 Section 134 Section 135 Section 136 Section 137 Section 138
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/MasteringMetadata
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	SMPTE 2086 mastering data.

129. PrimaryRChromaticityX

Element Name	PrimaryRChromaticityX
Element ID	0x55D1
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/PrimaryRChromaticityX
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Red X chromaticity coordinate as defined by CIE 1931.

130. PrimaryRChromaticityY

Element Name	PrimaryRChromaticityY
Element ID	0x55D2
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/PrimaryRChromaticityY
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Red Y chromaticity coordinate as defined by CIE 1931.

131. PrimaryGChromaticityX

Element Name	PrimaryGChromaticityX
Element ID	0x55D3
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/PrimaryGChromaticityX
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Green X chromaticity coordinate as defined by CIE 1931.

132. PrimaryGChromaticityY

Element Name	PrimaryGChromaticityY
Element ID	0x55D4
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/PrimaryGChromaticityY
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Green Y chromaticity coordinate as defined by CIE 1931.

133. PrimaryBChromaticityX

Element Name	PrimaryBChromaticityX
Element ID	0x55D5
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/PrimaryBChromaticityX
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Blue X chromaticity coordinate as defined by CIE 1931.

134. PrimaryBChromaticityY

Element Name	PrimaryBChromaticityY
Element ID	0x55D6
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/PrimaryBChromaticityY
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Blue Y chromaticity coordinate as defined by CIE 1931.

135. WhitePointChromaticityX

Element Name	WhitePointChromaticityX
Element ID	0x55D7
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/WhitePointChromaticityX
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	White X chromaticity coordinate as defined by CIE 1931.

136. WhitePointChromaticityY

Element Name	WhitePointChromaticityY
Element ID	0x55D8
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/WhitePointChromaticityY
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	White Y chromaticity coordinate as defined by CIE 1931.

137. LuminanceMax

Element Name	LuminanceMax
Element ID	0x55D9
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/LuminanceMax
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Maximum luminance. Represented in candelas per square meter (cd/m^2).

138. LuminanceMin

Element Name	LuminanceMin
Element ID	0x55DA
Element Type	float
Version	4-4
Parent Element	Section 128
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 95/Section 114/Section 128/LuminanceMin
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Mininum luminance. Represented in candelas per square meter (cd/m^2).

139. Audio

Element Name	Audio
Element ID	0xE1
Element Type	master
Version	1-4
Parent Element	Section 63
Child Elements	Section 140 Section 141 Section 142 Section 143 Section 144
Element Context	/Section 8/Section 62/Section 63/Audio
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Audio settings.

140. SamplingFrequency

Element Name	SamplingFrequency
Element ID	0xB5
Element Type	float
Version	1-4
Parent Element	Section 139
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 139/SamplingFrequency
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Sampling frequency in Hz.

141. OutputSamplingFrequency

Element Name	OutputSamplingFrequency
Element ID	0x78B5
Element Type	float
Version	1-4
Parent Element	Section 139
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 139/OutputSamplingFrequency
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Real output sampling frequency in Hz (used for SBR techniques).

142. Channels

Element Name	Channels
Element ID	0x9F
Element Type	uinteger
Version	1-4
Parent Element	Section 139
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 139/Channels
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Numbers of channels in the track.

143. ChannelPositions

Element Name	ChannelPositions
Element ID	0x7D7B
Element Type	binary
Version	1-4
Parent Element	Section 139
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 139/ChannelPositions
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Table of horizontal angles for each successive channel, see appendix.

144. BitDepth

Element Name	BitDepth
Element ID	0x6264
Element Type	uinteger
Version	1-4
Parent Element	Section 139
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 139/BitDepth
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Bits per sample, mostly used for PCM.

145. TrackOperation

Element Name	TrackOperation
Element ID	0xE2
Element Type	master
Version	3-4
Parent Element	Section 63
Child Elements	Section 146 Section 150
Element Context	/Section 8/Section 62/Section 63/TrackOperation
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Operation that needs to be applied on tracks to create this virtual track. For more details look at the Specification Notes on the subject.

146. TrackCombinePlanes

Element Name	TrackCombinePlanes
Element ID	0xE3
Element Type	master
Version	3-4
Parent Element	Section 145
Child Elements	Section 147
Element Context	/Section 8/Section 62/Section 63/Section 145/TrackCombinePlanes
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Contains the list of all video plane tracks that need to be combined to create this 3D track

147. TrackPlane

Element Name	TrackPlane
Element ID	0xE4
Element Type	master
Version	3-4
Parent Element	Section 146
Child Elements	Section 148 Section 149
Element Context	/Section 8/Section 62/Section 63/Section 145/Section 146/TrackPlane
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains a video plane track that need to be combined to create this 3D track

148. TrackPlaneUID

Element Name	TrackPlaneUID
Element ID	0xE5
Element Type	uinteger
Version	3-4
Parent Element	Section 147
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 145/Section 146/Section 147/TrackPlaneUID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The trackUID number of the track representing the plane.

149. TrackPlaneType

Element Name	TrackPlaneType
Element ID	0xE6
Element Type	uinteger
Version	3-4
Parent Element	Section 147
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 145/Section 146/Section 147/TrackPlaneType
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The kind of plane this track corresponds to (0: left eye, 1: right eye, 2: background).

150. TrackJoinBlocks

Element Name	TrackJoinBlocks
Element ID	0xE9
Element Type	master
Version	3-4
Parent Element	Section 145
Child Elements	Section 151
Element Context	/Section 8/Section 62/Section 63/Section 145/TrackJoinBlocks
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Contains the list of all tracks whose Blocks need to be combined to create this virtual track

151. TrackJoinUID

Element Name	TrackJoinUID
Element ID	0xED
Element Type	uinteger
Version	3-4
Parent Element	Section 150
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 145/Section 150/TrackJoinUID
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	The trackUID number of a track whose blocks are used to create this virtual track.

152. TrickTrackUID

Element Name	TrickTrackUID
Element ID	0xC0
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrickTrackUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX trick track extenstions

153. TrickTrackSegmentUID

Element Name	TrickTrackSegmentUID
Element ID	0xC1
Element Type	binary
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrickTrackSegmentUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX trick track extenstions

154. TrickTrackFlag

Element Name	TrickTrackFlag
Element ID	0xC6
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrickTrackFlag
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX trick track extenstions

155. TrickMasterTrackUID

Element Name	TrickMasterTrackUID
Element ID	0xC7
Element Type	uinteger
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrickMasterTrackUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX trick track extenstions

156. TrickMasterTrackSegmentUID

Element Name	TrickMasterTrackSegmentUID
Element ID	0xC4
Element Type	binary
Version	1-4
Parent Element	Section 63
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/TrickMasterTrackSegmentUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX trick track extenstions

157. ContentEncodings

Element Name	ContentEncodings
Element ID	0x6D80
Element Type	master
Version	1-4
Parent Element	Section 63
Child Elements	Section 158
Element Context	/Section 8/Section 62/Section 63/ContentEncodings
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Settings for several content encoding mechanisms like compression or encryption.

158. ContentEncoding

Element Name	ContentEncoding
Element ID	0x6240
Element Type	master
Version	1-4
Parent Element	Section 157
Child Elements	Section 159 Section 160 Section 161 Section 162 Section 165
Element Context	/Section 8/Section 62/Section 63/Section 157/ContentEncoding
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Settings for one content encoding like compression or encryption.

159. ContentEncodingOrder

Element Name	ContentEncodingOrder
Element ID	0x5031
Element Type	uinteger
Version	1-4
Parent Element	Section 158
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/ContentEncodingOrder
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Tells when this modification was used during encoding/muxing starting with 0 and counting upwards. The decoder/demuxer has to start with the highest order number it finds and work its way down. This value has to be unique over all ContentEncodingOrder Elements in the Segment.

160. ContentEncodingScope

Element Name	ContentEncodingScope
Element ID	0x5032
Element Type	uinteger
Version	1-4
Parent Element	Section 158
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/ContentEncodingScope
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A bit field that describes which Elements have been modified in this way. Values (big endian) can be OR’ed. Possible values: 1 - all frame contents, 2 - the track’s private data, 4 - the next ContentEncoding (next ContentEncodingOrder. Either the data inside ContentCompression and/or ContentEncryption)

161. ContentEncodingType

Element Name	ContentEncodingType
Element ID	0x5033
Element Type	uinteger
Version	1-4
Parent Element	Section 158
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/ContentEncodingType
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A value describing what kind of transformation has been done. Possible values: 0 - compression, 1 - encryption

162. ContentCompression

Element Name	ContentCompression
Element ID	0x5034
Element Type	master
Version	1-4
Parent Element	Section 158
Child Elements	Section 163 Section 164
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/ContentCompression
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	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.

163. ContentCompAlgo

Element Name	ContentCompAlgo
Element ID	0x4254
Element Type	uinteger
Version	1-4
Parent Element	Section 162
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/Section 162/ContentCompAlgo
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The compression algorithm used. Algorithms that have been specified so far are: 0 - zlib, 1 - bzlib, 2 - lzo1x 3 - Header Stripping

164. ContentCompSettings

Element Name	ContentCompSettings
Element ID	0x4255
Element Type	binary
Version	1-4
Parent Element	Section 162
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/Section 162/ContentCompSettings
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Settings that might be needed by the decompressor. For Header Stripping (ContentCompAlgo=3), the bytes that were removed from the beggining of each frames of the track.

165. ContentEncryption

Element Name	ContentEncryption
Element ID	0x5035
Element Type	master
Version	1-4
Parent Element	Section 158
Child Elements	Section 166 Section 167 Section 168 Section 169 Section 170 Section 171
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/ContentEncryption
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Settings describing the encryption used. This Element MUST be present if the value of ContentEncodingType is 1 and absent otherwise.

166. ContentEncAlgo

Element Name	ContentEncAlgo
Element ID	0x47E1
Element Type	uinteger
Version	1-4
Parent Element	Section 165
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/Section 165/ContentEncAlgo
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The encryption algorithm used. The value ’0’ means that the contents have not been encrypted but only signed. Predefined values: 1 - DES, 2 - 3DES, 3 - Twofish, 4 - Blowfish, 5 - AES

167. ContentEncKeyID

Element Name	ContentEncKeyID
Element ID	0x47E2
Element Type	binary
Version	1-4
Parent Element	Section 165
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/Section 165/ContentEncKeyID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	For public key algorithms this is the ID of the public key the the data was encrypted with.

168. ContentSignature

Element Name	ContentSignature
Element ID	0x47E3
Element Type	binary
Version	1-4
Parent Element	Section 165
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/Section 165/ContentSignature
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A cryptographic signature of the contents.

169. ContentSigKeyID

Element Name	ContentSigKeyID
Element ID	0x47E4
Element Type	binary
Version	1-4
Parent Element	Section 165
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/Section 165/ContentSigKeyID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	This is the ID of the private key the data was signed with.

170. ContentSigAlgo

Element Name	ContentSigAlgo
Element ID	0x47E5
Element Type	uinteger
Version	1-4
Parent Element	Section 165
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/Section 165/ContentSigAlgo
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The algorithm used for the signature. A value of ’0’ means that the contents have not been signed but only encrypted. Predefined values: 1 - RSA

171. ContentSigHashAlgo

Element Name	ContentSigHashAlgo
Element ID	0x47E6
Element Type	uinteger
Version	1-4
Parent Element	Section 165
Child Elements	None
Element Context	/Section 8/Section 62/Section 63/Section 157/Section 158/Section 165/ContentSigHashAlgo
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The hash algorithm used for the signature. A value of ’0’ means that the contents have not been signed but only encrypted. Predefined values: 1 - SHA1-160 2 - MD5

172. Cues

Element Name	Cues
Element ID	0x1C53BB6B
Element Type	master
Version	1-4
Parent Element	Section 8
Child Elements	Section 173
Element Context	/Section 8/Cues
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A Top-Level Element to speed seeking access. All entries are local to the Segment. This Element SHOULD be mandatory for non "live" streams.

173. CuePoint

Element Name	CuePoint
Element ID	0xBB
Element Type	master
Version	1-4
Parent Element	Section 172
Child Elements	Section 174 Section 175
Element Context	/Section 8/Section 172/CuePoint
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains all information relative to a seek point in the Segment.

174. CueTime

Element Name	CueTime
Element ID	0xB3
Element Type	uinteger
Version	1-4
Parent Element	Section 173
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/CueTime
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Absolute timestamp according to the Segment time base.

175. CueTrackPositions

Element Name	CueTrackPositions
Element ID	0xB7
Element Type	master
Version	1-4
Parent Element	Section 173
Child Elements	Section 176 Section 177 Section 178 Section 179 Section 180 Section 181 Section 182
Element Context	/Section 8/Section 172/Section 173/CueTrackPositions
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contain positions for different tracks corresponding to the timestamp.

176. CueTrack

Element Name	CueTrack
Element ID	0xF7
Element Type	uinteger
Version	1-4
Parent Element	Section 175
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/CueTrack
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The track for which a position is given.

177. CueClusterPosition

Element Name	CueClusterPosition
Element ID	0xF1
Element Type	uinteger
Version	1-4
Parent Element	Section 175
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/CueClusterPosition
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The position of the Cluster containing the associated Block.

178. CueRelativePosition

Element Name	CueRelativePosition
Element ID	0xF0
Element Type	uinteger
Version	4-4
Parent Element	Section 175
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/CueRelativePosition
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The relative position of the referenced block inside the cluster with 0 being the first possible position for an Element inside that cluster.

179. CueDuration

Element Name	CueDuration
Element ID	0xB2
Element Type	uinteger
Version	4-4
Parent Element	Section 175
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/CueDuration
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The duration of the block according to the Segment time base. If missing the track's DefaultDuration does not apply and no duration information is available in terms of the cues.

180. CueBlockNumber

Element Name	CueBlockNumber
Element ID	0x5378
Element Type	uinteger
Version	1-4
Parent Element	Section 175
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/CueBlockNumber
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Number of the Block in the specified Cluster.

181. CueCodecState

Element Name	CueCodecState
Element ID	0xEA
Element Type	uinteger
Version	2-4
Parent Element	Section 175
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/CueCodecState
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The position of the Codec State corresponding to this Cue Element. 0 means that the data is taken from the initial Track Entry.

182. CueReference

Element Name	CueReference
Element ID	0xDB
Element Type	master
Version	2-4
Parent Element	Section 175
Child Elements	Section 183 Section 184 Section 185 Section 186
Element Context	/Section 8/Section 172/Section 173/Section 175/CueReference
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	The Clusters containing the referenced Blocks.

183. CueRefTime

Element Name	CueRefTime
Element ID	0x96
Element Type	uinteger
Version	2-4
Parent Element	Section 182
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/Section 182/CueRefTime
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Timestamp of the referenced Block.

184. CueRefCluster

Element Name	CueRefCluster
Element ID	0x97
Element Type	uinteger
Version	1-4
Parent Element	Section 182
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/Section 182/CueRefCluster
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The Position of the Cluster containing the referenced Block.

185. CueRefNumber

Element Name	CueRefNumber
Element ID	0x535F
Element Type	uinteger
Version	1-4
Parent Element	Section 182
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/Section 182/CueRefNumber
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Number of the referenced Block of Track X in the specified Cluster.

186. CueRefCodecState

Element Name	CueRefCodecState
Element ID	0xEB
Element Type	uinteger
Version	1-4
Parent Element	Section 182
Child Elements	None
Element Context	/Section 8/Section 172/Section 173/Section 175/Section 182/CueRefCodecState
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The position of the Codec State corresponding to this referenced Element. 0 means that the data is taken from the initial Track Entry.

187. Attachments

Element Name	Attachments
Element ID	0x1941A469
Element Type	master
Version	1-4
Parent Element	Section 8
Child Elements	Section 188
Element Context	/Section 8/Attachments
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Contain attached files.

188. AttachedFile

Element Name	AttachedFile
Element ID	0x61A7
Element Type	master
Version	1-4
Parent Element	Section 187
Child Elements	Section 189 Section 190 Section 191 Section 192 Section 193 Section 194 Section 195 Section 196
Element Context	/Section 8/Section 187/AttachedFile
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	An attached file.

189. FileDescription

Element Name	FileDescription
Element ID	0x467E
Element Type	utf-8
Version	1-4
Parent Element	Section 188
Child Elements	None
Element Context	/Section 8/Section 187/Section 188/FileDescription
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A human-friendly name for the attached file.

190. FileName

Element Name	FileName
Element ID	0x466E
Element Type	utf-8
Version	1-4
Parent Element	Section 188
Child Elements	None
Element Context	/Section 8/Section 187/Section 188/FileName
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Filename of the attached file.

191. FileMimeType

Element Name	FileMimeType
Element ID	0x4660
Element Type	string
Version	1-4
Parent Element	Section 188
Child Elements	None
Element Context	/Section 8/Section 187/Section 188/FileMimeType
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	MIME type of the file.

192. FileData

Element Name	FileData
Element ID	0x465C
Element Type	binary
Version	1-4
Parent Element	Section 188
Child Elements	None
Element Context	/Section 8/Section 187/Section 188/FileData
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The data of the file.

193. FileUID

Element Name	FileUID
Element ID	0x46AE
Element Type	uinteger
Version	1-4
Parent Element	Section 188
Child Elements	None
Element Context	/Section 8/Section 187/Section 188/FileUID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Unique ID representing the file, as random as possible.

194. FileReferral

Element Name	FileReferral
Element ID	0x4675
Element Type	binary
Version	1-4
Parent Element	Section 188
Child Elements	None
Element Context	/Section 8/Section 187/Section 188/FileReferral
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A binary value that a track/codec can refer to when the attachment is needed.

195. FileUsedStartTime

Element Name	FileUsedStartTime
Element ID	0x4661
Element Type	uinteger
Version	1-4
Parent Element	Section 188
Child Elements	None
Element Context	/Section 8/Section 187/Section 188/FileUsedStartTime
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX font extension

196. FileUsedEndTime

Element Name	FileUsedEndTime
Element ID	0x4662
Element Type	uinteger
Version	1-4
Parent Element	Section 188
Child Elements	None
Element Context	/Section 8/Section 187/Section 188/FileUsedEndTime
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	DivX font extension

197. Chapters

Element Name	Chapters
Element ID	0x1043A770
Element Type	master
Version	1-4
Parent Element	Section 8
Child Elements	Section 198
Element Context	/Section 8/Chapters
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A system to define basic menus and partition data. For more detailed information, look at the Chapters Explanation.

198. EditionEntry

Element Name	EditionEntry
Element ID	0x45B9
Element Type	master
Version	1-4
Parent Element	Section 197
Child Elements	Section 199 Section 200 Section 201 Section 202 Section 203
Element Context	/Section 8/Section 197/EditionEntry
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains all information about a Segment edition.

199. EditionUID

Element Name	EditionUID
Element ID	0x45BC
Element Type	uinteger
Version	1-4
Parent Element	Section 198
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/EditionUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A unique ID to identify the edition. It’s useful for tagging an edition.

200. EditionFlagHidden

Element Name	EditionFlagHidden
Element ID	0x45BD
Element Type	uinteger
Version	1-4
Parent Element	Section 198
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/EditionFlagHidden
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	If an edition is hidden (1), it SHOULD NOT be available to the user interface (but still to Control Tracks; see flag notes). (1 bit)

201. EditionFlagDefault

Element Name	EditionFlagDefault
Element ID	0x45DB
Element Type	uinteger
Version	1-4
Parent Element	Section 198
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/EditionFlagDefault
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	If a flag is set (1) the edition SHOULD be used as the default one. (1 bit)

202. EditionFlagOrdered

Element Name	EditionFlagOrdered
Element ID	0x45DD
Element Type	uinteger
Version	1-4
Parent Element	Section 198
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/EditionFlagOrdered
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Specify if the chapters can be defined multiple times and the order to play them is enforced. (1 bit)

203. ChapterAtom

Element Name	ChapterAtom
Element ID	0xB6
Element Type	master
Version	1-4
Parent Element	Section 198
Child Elements	Section 204 Section 205 Section 206 Section 207 Section 208 Section 209 Section 210 Section 211 Section 212 Section 213 Section 215 Section 219
Element Context	/Section 8/Section 197/Section 198/ChapterAtom
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Recursive
Documentation	Contains the atom information to use as the chapter atom (apply to all tracks).

204. ChapterUID

Element Name	ChapterUID
Element ID	0x73C4
Element Type	uinteger
Version	1-4
Parent Element	Section 203
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterUID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A unique ID to identify the Chapter.

205. ChapterStringUID

Element Name	ChapterStringUID
Element ID	0x5654
Element Type	utf-8
Version	3-4
Parent Element	Section 203
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterStringUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A unique string ID to identify the Chapter. Use for WebVTT cue identifier storage.

206. ChapterTimeStart

Element Name	ChapterTimeStart
Element ID	0x91
Element Type	uinteger
Version	1-4
Parent Element	Section 203
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterTimeStart
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Timestamp of the start of Chapter (not scaled).

207. ChapterTimeEnd

Element Name	ChapterTimeEnd
Element ID	0x92
Element Type	uinteger
Version	1-4
Parent Element	Section 203
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterTimeEnd
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Timestamp of the end of Chapter (timestamp excluded, not scaled).

208. ChapterFlagHidden

Element Name	ChapterFlagHidden
Element ID	0x98
Element Type	uinteger
Version	1-4
Parent Element	Section 203
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterFlagHidden
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	If a chapter is hidden (1), it SHOULD NOT be available to the user interface (but still to Control Tracks; see flag notes). (1 bit)

209. ChapterFlagEnabled

Element Name	ChapterFlagEnabled
Element ID	0x4598
Element Type	uinteger
Version	1-4
Parent Element	Section 203
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterFlagEnabled
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Specify wether the chapter is enabled. It can be enabled/disabled by a Control Track. When disabled, the movie SHOULD skip all the content between the TimeStart and TimeEnd of this chapter (see flag notes). (1 bit)

210. ChapterSegmentUID

Element Name	ChapterSegmentUID
Element ID	0x6E67
Element Type	binary
Version	1-4
Parent Element	Section 203
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterSegmentUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The SegmentUID of another Segment to play during this chapter.

211. ChapterSegmentEditionUID

Element Name	ChapterSegmentEditionUID
Element ID	0x6EBC
Element Type	uinteger
Version	1-4
Parent Element	Section 203
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterSegmentEditionUID
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The EditionUID to play from the Segment linked in ChapterSegmentUID. If ChapterSegmentEditionUID is undeclared then no Edition of the linked Segment is used.

212. ChapterPhysicalEquiv

Element Name	ChapterPhysicalEquiv
Element ID	0x63C3
Element Type	uinteger
Version	1-4
Parent Element	Section 203
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterPhysicalEquiv
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Specify the physical equivalent of this ChapterAtom like "DVD" (60) or "SIDE" (50), see complete list of values.

213. ChapterTrack

Element Name	ChapterTrack
Element ID	0x8F
Element Type	master
Version	1-4
Parent Element	Section 203
Child Elements	Section 214
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterTrack
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	List of tracks on which the chapter applies. If this Element is not present, all tracks apply

214. ChapterTrackNumber

Element Name	ChapterTrackNumber
Element ID	0x89
Element Type	uinteger
Version	1-4
Parent Element	Section 213
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/Section 213/ChapterTrackNumber
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	UID of the Track to apply this chapter too. In the absense of a control track, choosing this chapter will select the listed Tracks and deselect unlisted tracks. Absense of this Element indicates that the Chapter SHOULD be applied to any currently used Tracks.

215. ChapterDisplay

Element Name	ChapterDisplay
Element ID	0x80
Element Type	master
Version	1-4
Parent Element	Section 203
Child Elements	Section 216 Section 217 Section 218
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapterDisplay
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains all possible strings to use for the chapter display.

216. ChapString

Element Name	ChapString
Element ID	0x85
Element Type	utf-8
Version	1-4
Parent Element	Section 215
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/Section 215/ChapString
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Contains the string to use as the chapter atom.

217. ChapLanguage

Element Name	ChapLanguage
Element ID	0x437C
Element Type	string
Version	1-4
Parent Element	Section 215
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/Section 215/ChapLanguage
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	The languages corresponding to the string, in the bibliographic ISO-639-2 form.

218. ChapCountry

Element Name	ChapCountry
Element ID	0x437E
Element Type	string
Version	1-4
Parent Element	Section 215
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/Section 215/ChapCountry
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	The countries corresponding to the string, same 2 octets as in Internet domains.

219. ChapProcess

Element Name	ChapProcess
Element ID	0x6944
Element Type	master
Version	1-4
Parent Element	Section 203
Child Elements	Section 220 Section 221 Section 222
Element Context	/Section 8/Section 197/Section 198/Section 203/ChapProcess
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains all the commands associated to the Atom.

220. ChapProcessCodecID

Element Name	ChapProcessCodecID
Element ID	0x6955
Element Type	uinteger
Version	1-4
Parent Element	Section 219
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/Section 219/ChapProcessCodecID
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Contains the type of the codec used for the processing. A value of 0 means native Matroska processing (to be defined), a value of 1 means the DVD command set is used. More codec IDs can be added later.

221. ChapProcessPrivate

Element Name	ChapProcessPrivate
Element ID	0x450D
Element Type	binary
Version	1-4
Parent Element	Section 219
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/Section 219/ChapProcessPrivate
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Some optional data attached to the ChapProcessCodecID information. For ChapProcessCodecID = 1, it is the "DVD level" equivalent.

222. ChapProcessCommand

Element Name	ChapProcessCommand
Element ID	0x6911
Element Type	master
Version	1-4
Parent Element	Section 219
Child Elements	Section 223 Section 224
Element Context	/Section 8/Section 197/Section 198/Section 203/Section 219/ChapProcessCommand
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Contains all the commands associated to the Atom.

223. ChapProcessTime

Element Name	ChapProcessTime
Element ID	0x6922
Element Type	uinteger
Version	1-4
Parent Element	Section 222
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/Section 219/Section 222/ChapProcessTime
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Defines when the process command SHOULD be handled (0: during the whole chapter, 1: before starting playback, 2: after playback of the chapter).

224. ChapProcessData

Element Name	ChapProcessData
Element ID	0x6933
Element Type	binary
Version	1-4
Parent Element	Section 222
Child Elements	None
Element Context	/Section 8/Section 197/Section 198/Section 203/Section 219/Section 222/ChapProcessData
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	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.

225. Tags

Element Name	Tags
Element ID	0x1254C367
Element Type	master
Version	1-4
Parent Element	Section 8
Child Elements	Section 226
Element Context	/Section 8/Tags
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Element containing Elements specific to Tracks/Chapters. A list of valid tags can be found here.

226. Tag

Element Name	Tag
Element ID	0x7373
Element Type	master
Version	1-4
Parent Element	Section 225
Child Elements	Section 227 Section 234
Element Context	/Section 8/Section 225/Tag
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	Element containing Elements specific to Tracks/Chapters.

227. Targets

Element Name	Targets
Element ID	0x63C0
Element Type	master
Version	1-4
Parent Element	Section 226
Child Elements	Section 228 Section 229 Section 230 Section 231 Section 232 Section 233
Element Context	/Section 8/Section 225/Section 226/Targets
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Contain all UIDs where the specified meta data apply. It is empty to describe everything in the Segment.

228. TargetTypeValue

Element Name	TargetTypeValue
Element ID	0x68CA
Element Type	uinteger
Version	1-4
Parent Element	Section 227
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 227/TargetTypeValue
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	A number to indicate the logical level of the target (see TargetType).

229. TargetType

Element Name	TargetType
Element ID	0x63CA
Element Type	string
Version	1-4
Parent Element	Section 227
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 227/TargetType
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	An informational string that can be used to display the logical level of the target like "ALBUM", "TRACK", "MOVIE", "CHAPTER", etc (see TargetType).

230. TagTrackUID

Element Name	TagTrackUID
Element ID	0x63C5
Element Type	uinteger
Version	1-4
Parent Element	Section 227
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 227/TagTrackUID
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	A unique ID to identify the Track(s) the tags belong to. If the value is 0 at this level, the tags apply to all tracks in the Segment.

231. TagEditionUID

Element Name	TagEditionUID
Element ID	0x63C9
Element Type	uinteger
Version	1-4
Parent Element	Section 227
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 227/TagEditionUID
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	A unique ID to identify the EditionEntry(s) the tags belong to. If the value is 0 at this level, the tags apply to all editions in the Segment.

232. TagChapterUID

Element Name	TagChapterUID
Element ID	0x63C4
Element Type	uinteger
Version	1-4
Parent Element	Section 227
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 227/TagChapterUID
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	A unique ID to identify the Chapter(s) the tags belong to. If the value is 0 at this level, the tags apply to all chapters in the Segment.

233. TagAttachmentUID

Element Name	TagAttachmentUID
Element ID	0x63C6
Element Type	uinteger
Version	1-4
Parent Element	Section 227
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 227/TagAttachmentUID
Mandatory	Not Mandatory
Repeatability	Repeatable
Recursive	Not Recursive
Documentation	A unique ID to identify the Attachment(s) the tags belong to. If the value is 0 at this level, the tags apply to all the attachments in the Segment.

234. SimpleTag

Element Name	SimpleTag
Element ID	0x67C8
Element Type	master
Version	1-4
Parent Element	Section 226
Child Elements	Section 235 Section 236 Section 237 Section 238 Section 239
Element Context	/Section 8/Section 225/Section 226/SimpleTag
Mandatory	Mandatory
Repeatability	Repeatable
Recursive	Recursive
Documentation	Contains general information about the target.

235. TagName

Element Name	TagName
Element ID	0x45A3
Element Type	utf-8
Version	1-4
Parent Element	Section 234
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 234/TagName
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The name of the Tag that is going to be stored.

236. TagLanguage

Element Name	TagLanguage
Element ID	0x447A
Element Type	string
Version	1-4
Parent Element	Section 234
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 234/TagLanguage
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Specifies the language of the tag specified, in the Matroska languages form.

237. TagDefault

Element Name	TagDefault
Element ID	0x4484
Element Type	uinteger
Version	1-4
Parent Element	Section 234
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 234/TagDefault
Mandatory	Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	Indication to know if this is the default/original language to use for the given tag. (1 bit)

238. TagString

Element Name	TagString
Element ID	0x4487
Element Type	utf-8
Version	1-4
Parent Element	Section 234
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 234/TagString
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The value of the Tag.

239. TagBinary

Element Name	TagBinary
Element ID	0x4485
Element Type	binary
Version	1-4
Parent Element	Section 234
Child Elements	None
Element Context	/Section 8/Section 225/Section 226/Section 234/TagBinary
Mandatory	Not Mandatory
Repeatability	Not Repeatable
Recursive	Not Recursive
Documentation	The values of the Tag if it is binary. Note that this cannot be used in the same SimpleTag as TagString.

If you intend to implement a Matroska player, make sure you can handle all the files in our test suite [20], or at least the features presented there, not necessarily the same codecs.

240. Beginning of File

An EBML file always starts with 0x1A. The 0x1A makes the DOS command "type" ends display. That way you can include ASCII text before the EBML data and it can be displayed. The EBML parser is safe from false-alarm with these ASCII only codes.

Next the EBML header is stored. This allows the the parser to know what type of EBML file it is parsing.

241. Block Timecodes

The Block’s timecode is signed interger that represents the Raw Timecode relative to the Cluster’s [21] Timecode [22], multiplied by the TimecodeScale (see the TimecodeScale notes [23]).

The Block’s timecode is represented by a 16bit signed interger (sint16). This means that the Block’s timecode has a range of -32768 to +32767 units. When using the default value of TimecodeScale, each integer represents 1ms. So, the maximum time span of Blocks in a Cluster using the default TimecodeScale of 1ms is 65536ms.

If a Cluster's [24] Timecode [25] is set to zero, it is possible to have Blocks with a negative Raw Timecode. Blocks with a negative Raw Timecode are not valid.

242. Default decoded field duration

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

If the video sequence is signaled as interlaced, then the period between two successive fields at the output of the decoding process equals DefaultDecodedFieldDuration.

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

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

Examples:

- Blu-ray movie: 1000000000ns/(48/1.001) = 20854167ns
- PAL broadcast/DVD: 1000000000ns/(50/1.000) = 20000000ns
- N/ATSC broadcast: 1000000000ns/(60/1.001) = 16683333ns
- hard-telecined DVD: 1000000000ns/(60/1.001) = 16683333ns (60 encoded interlaced fields per second)
- soft-telecined DVD: 1000000000ns/(60/1.001) = 16683333ns (48 encoded interlaced fields per second, with "repeat_first_field = 1")

243. Default Values

The default value of an Element is assumed when not present in the data stream. It is assumed only in the scope of its Parent Element (for example "Language" in the scope of the "Track" element). If the "Parent Element" is not present or assumed, then the Element cannot be assumed.

244. DRM

Digital Rights Management. See Encryption [26].

245. EBML Class

A larger EBML Class typically means the Element has a lower probability/importance. A larger Class-ID can be used as a synch word in case the file is damaged. Elements that are used frequently, but do not need to act as a synch word, SHOULD have a small Class-ID. For example, the Cluster has a 4 octet ID and can be used as a synch word if the file is damaged. However, the every common Element in the BlockGroup has a single octet ID to conserve space because of how frequently it is used.

246. Encryption

Encryption in Matroska is designed in a very generic style that allows people to implement whatever form of encryption is best for them. It is easily possible to use the encryption framework in Matroska as a type of DRM.

Because the encryption occurs within the Block, 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 ever decrypting them. This means that the data is more useful, without having to expose it, or go through the intensive process of decrypting.

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

Encryption information is stored in the "ContentEncodings" Master-element under the "ContentEncryption" Element.

247. Image cropping

Thanks to the PixelCropXXX elements, it’s possible to crop the image before being resized. That means the image size follows this path:

PixelXXX (size of the coded image) -> PixelCropXXX (size of the image to keep) -> DisplayXXX (resized cropped image)

248. Matroska version indicators

The EBML Header each Matroska file starts with contains two version number fields that inform a reading application about what to expect. These are "DocTypeVersion" and "DocTypeReadVersion".

"DocTypeVersion" MUST contain the highest Matroska version number of any Element present in the Matroska file. For example, a file using the SimpleBlock Element MUST have a "DocTypeVersion" of at least 2 while a file containing "CueRelativePosition" Elements MUST have a "DocTypeVersion" of at least 4.

The "DocTypeReadVersion" MUST contain the minimum version number a reading application MUST at least suppost properly 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 REQUIRED for standard playback -- it only makes seeking more precise if used.

"DocTypeVersion" MUST always be equal to or greater than "DocTypeReadVersion".

A reading application supporting Matroska version "V" MUST NOT refuse to read an application with "DocReadTypeVersion" equal to or lower than "V" even if "DocTypeVersion" is greater than "V". See also the note about Unknown Elements [27].

249. Mime Types

There is no IETF endorsed MIME type for Matroska files. But you can use the ones we have defined on our web server:

- .mka : Matroska audio "audio/x-matroska"

- .mkv : Matroska video "video/x-matroska"

- .mk3d : Matroska 3D video "video/x-matroska-3d"

250. Octet

An Octet refers to a byte made of 8 bits.

251. 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 is played on the rendering channel instead of the original track.

252. Position References

The position in some Elements refers to the position, in octets, from the beginning of an Element. The reference is the beginning of the first Segment (= its position + the size of its ID and size fields). 0 = first possible position of a level 1 Element in the Segment. When data is spanned over mutiple Segments within a Section 254 (in the same file or in different files), the position represents the accumulated offset of each Segment. For example to reference a position in the third Segment, the position will be: the first segment total size + second segment total size + offset of the Element in the third segment.

253. Raw Timecode

The exact time of an object represented in nanoseconds. To find out a Block's Raw Timecode, you need the Block's timecode, the Cluster's [28] Timecode [29], and the TimecodeScale. For calculation, please see the see the TimecodeScale notes.

254. Linked Segments

Matroska provides several methods to link two or many Segments together to create a Linked Segment. A Linked Segment is a set of multiple Segments related together into a single presentation by using Hard Linking, Soft Linking, or Medium Linking. All Segments within a Linked Segment MUST utilize the same track numbers and timescale. All Segments within a Linked Segment MUST be stored within the same directory. All Segments within a Linked Segment MUST store a "SegmentUID".

254.1. Hard Linking

Hard Linking (also called splitting) is the process of creating a Linked Segment by relating multiple Segments using the "PrevUID" and "NextUID" Elements. Within a Linked Segment the timestamps of each Segment MUST follow consecutively in linking order. With Hard Linking, the chapters of any Segment within the Linked Segment MUST only reference the current Segment. With Hard Linking, the "NextUID" and "PrevUID" MUST reference the respective "SegmentUID" values of the next and previous Segments. The first Segment of a Linked Segment MUST have a "NextUID" Element and MUST NOT have a "PrevUID"

Element. The last Segment of a Linked Segment MUST have a "PrevUID" Element and MUST NOT have a "NextUID" Element. The middle Segments of a Linked Segment MUST have both a "NextUID" Element and a "PrevUID" Element.

As an example four Segments MAY be Hard Linked as a Linked Segment through cross-referencing each other with "SegmentUID", "PrevUID", and "NextUID" as in this table.

file name	SegmentUID	PrevUID	NextUID
"start.mkv"	"71000c23cd31099853fbc94dd984a5dd"	n/a	"a77b3598941cb803eac0fcdafe44fac9"
"middle.mkv"	"a77b3598941cb803eac0fcdafe44fac9"	"71000c23cd31099853fbc94dd984a5dd"	"6c92285fa6d3e827b198d120ea3ac674"
"end.mkv"	"6c92285fa6d3e827b198d120ea3ac674"	"a77b3598941cb803eac0fcdafe44fac9"	n/a

254.2. Soft Linking

Soft Linking is used by codec chapters. They can reference another Segment and jump to that Segment. The way the Segments are described are internal to the chapter codec and unknown to the Matroska level. But there are Elements within the "Info" Element (such as "ChapterTranslate") that can translate a value representing a Segment in the chapter codec and to the current "SegmentUID". All Segments that could be used in a Linked Segment in this way SHOULD be marked as members of the same family via the SegmentFamily Element, so that the player can quickly switch from one to the other.

254.3. Medium Linking

WMedium Linking creates relationships between Segments using Ordered Chapters and the "ChapterSegmentUID" Element. A Segment Edition with Ordered Chapters MAY contain Chapters that reference timestamp ranges from other Segments. The Segment referenced by the Ordered Chapter via the "ChapterSegmentUID" Element SHOULD be played as part of a Linked Segment. The timestamps of Segment content referenced by Ordered Chapters MUST be adjusted according to the cumulative duration of the the previous Ordered Chapters.

As an example a file named "intro.mkv" could have a "SegmentUID" 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 "ChapterSegmentUID", "ChapterSegmentEditionUID", "ChapterTimeStart" and optionally a "ChapterTimeEnd" element. The second chapter references content within the Segment of "program.mkv". A player SHOULD recognize the Linked Segment created by the use of "ChapterSegmentUID" in an enabled Edition and present the reference content of the two Segments together.

255. Timecode Types

- o Absolute Timecode = Block+Cluster
- o Relative Timecode = Block
- o Scaled Timecode = Block+Cluster
- o Raw Timecode = (Block+Cluster)_TimecodeScale_TrackTimecodeScale

256. TimecodeScale

The TimecodeScale [30] is used to calculate the Raw Timecode of a Block. The timecode is obtained by adding the Block's timecode to the Cluster's [31] Timecode [32], and then multiplying that result by the TimecodeScale. The result will be the Block's Raw Timecode in nanoseconds. The formula for this would look like:

```
(a + b) * c

a = [Block's Timecode]({{site.baseurl}}/index.html#block-header)
b = [Cluster's](#cluster) [Timecode](#timecode)
c = [TimeCodeScale]({{site.baseurl}}/index.html#TimeCodeScale)
```

An example of this is, assume a Cluster's [33] Timecode [34] has a value of 564264, the Block has a Timecode of 1233, and the timecodescale is the default of 1000000.

$$(1233 + 564264) * 1000000 = 565497000000$$

So, the Block in this example has a specific time of 565497000000 in nanoseconds. In milliseconds this would be 565497ms.

257. TimecodeScale Rounding

Because the default value of TimecodeScale is 1000000, which makes each integer in the Cluster and Block timecodes equal 1ms, this is the most commonly used. When dealing with audio, this causes innaccuracy with where you are seeking to. When the audio is combined with video, this is not an issue. For most cases the the synch of audio to video does not need to be more than 1ms accurate. This becomes obvious when one considers that sound will take 2-3ms to travel a single meter, so distance from your speakers will have a greater effect on audio/visual synch than this.

However, when dealing with audio only files, seeking accuracy can become critical. For instance, when storing a whole CD in a single track, you want to be able to seek to the exact sample that a song begins at. If you seek a few sample ahead or behind then a 'crack' or 'pop' may result as a few odd samples are rendered. Also, when performing precise editing, it may be very useful to have the audio accuracy down to a single sample.

It is usually true that when storing timecodes for an audio stream, the TimecodeScale MUST have an accuracy of at least that of the audio samplerate, otherwise there are rounding errors that prevent you from knowing the precise location of a sample. Here's how a program has to round each timecode in order to be able to recreate the sample number accurately.

Let's assume that the application has an audio track with a sample rate of 44100. As written above the TimecodeScale MUST have at least the accuracy of the sample rate itself: 1000000000 / 44100 = 22675.7369614512. This value MUST always be truncated. Otherwise the accuracy will not suffice. So in this example the application wil use 22675 for the TimecodeScale. The application could even use some lower value like 22674 which would allow it to be a little bit imprecise about the original timecodes. But more about that in a minute.

Next the application wants to write sample number 52340 and calculates the timecode. This is easy. In order to calculate the Raw Timecode in ns all it has to do is calculate "RawTimecode = round(1000000000 * sample_number / sample_rate)". Rounding at this stage is very important! The application might skip it if it choses a slightly smaller value for the TimecodeScale factor instead of the truncated one like shown above. Otherwise it has to round or the results won't be reversible. For our example we get "RawTimecode = round(1000000000 * 52340 / 44100) = round(1186848072.56236) = 1186848073".

The next step is to calculate the Absolute Timecode - that is the timecode that will be stored in the Matroska file. Here the application has to divide the Raw Timecode from the previous paragraph by the TimecodeScale factor and round the result: "AbsoluteTimecode = round(RawTimecode / TimecodeScale_facotr)" which will result in the following for our example: "AbsoluteTimecode = round(1186848073 / 22675) = round(52341.7011245866) = 52342". This number is the one the application has to write to the file.

Now our file is complete, and we want to play it back with another application. Its task is to find out which sample the first application wrote into the file. So it starts reading the Matroska file and finds the TimecodeScale factor 22675 and the audio sample rate 44100. Later it finds a data block with the Absolute Timecode of 52342. But how does it get the sample number from these numbers?

First it has to calculate the Raw Timecode of the block it has just read. Here's no rounding involved, just an integer multiplication: "RawTimecode = AbsoluteTimecode * TimecodeScale_factor". In our example: "RawTimecode = 52342 * 22675 = 1186854850".

The conversion from the RawTimecode to the sample number again requires rounding: "sample_number = round(RawTimecode * sample_rate / 1000000000)". In our example: "sample_number = round(1186854850 * 44100 / 1000000000) = round(52340.298885) = 52340". This is exactly the sample number that the previous program started with.

Some general notes for a program:

1. Always calculate the timestamps / sample numbers with floating point numbers of at least 64bit precision (called 'double' in most modern programming languages). If you're calculating with integers then make sure they're 64bit long, too.

2. Always round if you divide. Always! If you don't you'll end up with situations in which you have a timecode in the Matroska file that does not correspond to the sample number that it started with. Using a slightly lower timecode scale factor can help here in that it removes the need for proper rounding in the conversion from sample number to Raw Timecode.

If you want some sample code for all these calculations you can have a look at this small C program. For a given sample rate it will iterate over each sample, calculate the AbsoluteTimestamp and then re-calculate the sample number.

258. Track Flags

258.1. Default flag

The "default track" flag is a hint for the playback application and SHOULD always be changeable by the user. If the user wants to see or hear a track of a certain kind (audio, video, subtitles) and she hasn't chosen a specific track then the player SHOULD use the first track of that kind whose "default track" flag is set to "1". If no such track is found then the first track of this kind SHOULD be chosen.

Only one track of a kind MAY have its "default track" flag set in a segment. If a track entry does not contain the "default track" flag element then its default value "1" is to be used.

258.2. Forced flag

The "forced" flag tells the playback application that it MUST display/play this track or another track of the same kind that also has its "forced" flag set. When there are multiple "forced" tracks, the player SHOULD determined based upon the language of the forced flag or use the default flag if no track matches the use languages. Another track of the same kind without the "forced" flag may be use simultaneously with the "forced" track (like DVD subtitles for example).

259. TrackTimecodeScale

The TrackTimecodeScale [35] is used align tracks that would otherwise be played at different speeds. An example of this would be if you have a film that was originally recorded at 24fps video. When playing this back through a PAL broadcasting system, it is standard to speed up the film to 25fps to match the 25fps display speed of the PAL broadcasting standard. However, when broadcasting the video through NTSC, it is typical to leave the film at its original speed. If you wanted to make a single file where there was one video stream, and an audio stream used from the PAL broadcast, as well as an audio stream used from the NTSC broadcast, you would have the problem that the PAL audio stream would be 1/24th faster than the NTSC audio stream, quickly leading to problems. It is possible to stretch out the PAL audio track and reencode it at a slower speed, however when dealing with lossy audio codecs, this often results in a loss of audio quality and/or larger file sizes.

This is the type of problem that TrackTimecodeScale was designed to fix. Using it, the video can be played back at a speed that will synch with either the NTSC or the PAL audio stream, depending on which is being used for playback. To continue the above example:

```
Track 1: Video
Track 2: NTSC Audio
Track 3: PAL Audio
```

Because the NTSC track is at the original speed, it will used as the default value of 1.0 for its TrackTimecodeScale. The video will also be aligned to the NTSC track with the default value of 1.0.

The TrackTimecodeScale value to use for the PAL track would be calculated by determining how much faster the PAL track is than the NTSC track. In this case, because we know the video for the NTSC audio is being played back at 24fps and the video for the PAL audio is being played back at 25fps, the calculation would be:

(25 / 24) = ˜ 1.04166666666666666667

When writing a file that uses a non-default TrackTimecodeScale, the values of the Block's timecode are whatever they would be when normally storing the track with a default value for the TrackTimecodeScale. However, the data is interleaved a little differently. Data SHOULD be interleaved by its Raw Timecode [36] in the order handed back from the encoder. The Raw Timecode of a Block from a track using TrackTimecodeScale is calculated using:

"(Block's Timecode + Cluster's Timecode) * TimecodeScale * TrackTimecodeScale"

So, a Block from the PAL track above that had a Scaled Timecode [37] of 100 seconds would have a Raw Timecode of 104.66666667 seconds, and so would be stored in that part of the file.

When playing back a track using the TrackTimecodeScale, if the track is being played by itself, there is no need to scale it. From the above example, when playing the Video with the NTSC Audio, neither are scaled. However, when playing back the Video with the PAL Audio, the timecodes from the PAL Audio track are scaled using the TrackTimecodeScale, resulting in the video playing back in synch with the audio.

It would be possible for a player to also adjust the audio's samplerate at the same time as adjusting the timecodes if you wanted to play the two audio streams synchronously. It would also be possible to adjust the video to match the audio's speed. However, for playback, the selected track(s) timecodes SHOULD be adjusted if they need to be scaled.

While the above example deals specifically with audio tracks, this element can be used to align video, audio, subtitles, or any other type of track contained in a Matroska file.

260. Unknown elements

Matroska is based upon the principal 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 [38] 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 signalling that a reading application MUST only support a lower version number properly in order to play it back (possibly with a reduced feature set). This implies that 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 which it does not know about if that unknown element fits into the size constraints set by the current parent element.

261. Multi-planar and 3D videos

There are 2 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 [39] 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/Height 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 colours 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 2 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 [40] with TrackCombinePlanes [41] do that. For more details look at how TrackOperation works [42].

The 3D support is still in infancy and may evolve to support more features.

/index.html#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. Readers may support these legacy files by checking Matroska v2 or 0x53B9. The olders values were 0: mono, 1: right eye, 2: left eye, 3: both eyes

262. Track Operation

TrackOperation [43] allows combining multiple tracks to make a virtual one. It uses 2 separate system to combine tracks. One to create a 3D "composition" (left/right/background planes) and one to simplify join 2 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 corresponding to such a virtual track SHOULD be the sum of the Cues elements for each of the tracks it’s composed of (when the Cues are defined per track).

In the case of TrackJoinBlocks, the Blocks (from BlockGroup and SimpleBlock) of all the tracks SHOULD be used as if they were defined for this new virtual Track. When 2 Blocks have overlapping start or end timecodes, it’s up to the underlying system to either drop some of these frames or render them the way they overlap. In the end this situation SHOULD be avoided when creating such tracks as you can never be sure of the end result on different platforms.

263. Matroska Element Ordering Guidelines

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

263.1. Top-Level Elements

A valid Matroska file requires only one Top-Level Element, the "Info" Element; however, to be playable Matroska MUST also contain at least one "Tracks" and "Cluster" Element. The first "Info" Element and the first "Tracks" Element MUST either be stored before the first "Cluster" Element or both be referenced by a "SeekHead" Element which occurs before the first "Cluster" Element.

After a Matroska file has been created it could still be edited. 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 identify 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; therefore, it is RECOMMENDED to use Void Elements as padding in between Top-Level Elements.

263.2. 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. The Matroska specification recommends that "CRC-32" Elements SHOULD NOT be used as an immediate Child Element of the "Segment" Element; however all Top-Level Elements of an EBML Document SHOULD include a CRC-32 Element as a Child Element.

263.3. SeekHead

If used, the first "SeekHead" Element SHOULD be the first non-"CRC-32" Child Element of the "Segment" Element. If a second "SeekHead" Element is used then the first "SeekHead" MUST reference the identity and position of the second "SeekHead", the second "SeekHead" MUST only reference "Cluster" Elements and not any other Top-Level Element, and the second "SeekHead" MAY be stored in any order relative to the other Top-Level Elements. Whether one or two "SeekHead" Elements is used, the "SeekHead" Element(s) MUST reference the identify and position of all Top-Level Elements except for the first "SeekHead".

It is RECOMMENDED that the first "SeekHead" Element be followed by some padding (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.

263.4. Cues (index)

The "Cues" Element is RECOMMENDED to optimize seeking access in Matroska. It is programmatically simpler to add the "Cues" Element after all of the "Cluster" Elements are written because this does not require a prediction of how much space to reserve before writing the "Cluster" Elements. On the other hand, storing the "Cues" Element before the "Clusters" can provide some seeking advantages.

263.5. Info

The first "Info" Element SHOULD occur before the first "Tracks" and first "Cluster" Element.

263.6. Chapters

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

263.7. Attachments

The "Attachments" Element is not meant to use by default when playing the file, but could contain the cover art and/or fonts. Cover art is useful even before the file is played and fonts could be needed before playback starts for initialization of subtitles that could use them. 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.

263.8. Tags

The "Tags" Element is the one that is most subject to changes after the file was originally created. So for easier editing the "Tags" Element SHOULD be placed at the end of the "Segment" Element, even after the "Attachments" Element. On the other hand, it is inconvenient to have to seek in the "Segment" for tags especially for network streams. So it’s better if the "Tags" Element(s) are found early in the stream. When editing the "Tags" Element(s), the original "Tags" Element at the beginning can be voided [44] and a new one written right at the end [45] of the "Segment" Element. The file size will only marginally change.

263.9. Optimum layout from a muxer

- SeekHead
- Info
- Tracks
- Chapters
- Attachments
- Tags
- Clusters
- Cues

263.10. Optimum layout after editing tags

- SeekHead
- Info
- Tracks
- Chapters
- Attachments
- Void
- Clusters
- Cues
- Tags

263.11. Optimum layout with Cues at the front

- SeekHead
- Info
- Tracks
- Chapters

- o Attachments

- o Tags

- o Cues

- o Clusters

263.12. Cluster Timecode

As each "BlockGroup" and "SimpleBlock" of a "Cluster" Element needs the Cluster "Timecode", the "Timecode" Element MUST occur as the first Child Element within the "Cluster" Element.

264. CodecID

As an additional resource to this page Haali has created a list of codec IDs in a PDF [46].

For each TrackEntry inside matroska [47], there has to be a CodecID [48] defined. This ID is represent the codec used to encode data in the Track. The codec works with the coded data in the stream, but also with some codec initialisation. There are 2 different kind of codec "initialisation" :

- o CodecPrivate in the TrackEntry

- o CodecState in the BlockGroup

Each of these elements contain the same kind of data. And these data depend on the codec used.

Important Note:

Please, when reading through this list, always keep in mind that the intention behind it is "NOT" to list all existing audio and video codecs, but more to list those codecs that are "currently supported" in matroska (or will be supported soon), and therfore need a well defined codec ID so that all developers supporting matroska will use the same ID. A list of all the codecs we plan to support in the future can be found on the CoreCodec forum [49] (subject to be changed constantly). If you feel we missed support for a very important codec, please tell us on our development mailing list (matroska-devel at freelists.org).

See

| Codec ID | Name | Description | | Video | | V_MS/VFW/FOURCC | Microsoft (TM) Video Codec Manager (VCM) | V_MS/VFW/FOURCC - Microsoft (TM) Video Codec Manager (VCM) The private data contains the VCM structure BITMAPINFOHEADER including the extra private bytes, as defined by Microsoft [50]. The data are stored in little endian format (like on IA32 machines). Where is the Huffman table stored in HuffYUV, not AVISTREAMINFO ??? And the FourCC, not in AVISTREAMINFO.fccHandler ??? | | V_UNCOMPRESSED | Video, raw uncompressed video frames | The private data is void, all details about the used colour specs and bit depth are to be put/read from the KaxCodecColourSpace elements. | | V_MPEG4/ISO/??? | MPEG4 ISO Profile Video | The stream complies with, and uses the CodecID for, one of the MPEG-4 profiles listed below. |

| V_MPEG4/ISO/SP | MPEG4 ISO simple profile (DivX4) | stream was created via improved codec API (UCI) or even transmuxed from AVI (no b-frames in Simple Profile), frame order is coding order | | V_MPEG4/ISO/ASP | MPEG4 ISO advanced simple profile (DivX5, XviD, FFMPEG) | stream was created via improved codec API (UCI) or transmuxed from MP4, not simply transmuxed from AVI! Note there are differences how b-frames are handled in these native streams, when being compared to a VfW created stream, as here there are "no" dummy frames inserted, the frame order is exactly the same as the coding order, same as in MP4 streams! | | V_MPEG4/ISO/AP | MPEG4 ISO advanced profile | stream was created ... (see above) |

| | V_MPEG4/MS/V3 | Microsoft (TM) MPEG4 V3 | and derivates, means DivX3, Angelpotion, SMR, etc.; stream was created using VfW codec or transmuxed from AVI; note that V1/V2 are covered in VfW compatibility mode | | V_MPEG1 | MPEG 1 | The matroska video stream will contain a demuxed Elementary Stream (ES), where block boundaries are still to be defined. Its RECOMMENDED to use MPEG2MKV.exe for creating those files, and to compare the results with selfmade implementations | | V_MPEG2 | MPEG 2 | The matroska video stream will contain a demuxed Elementary Stream (ES), where block boundaries are still to be defined. Its RECOMMENDED to use MPEG2MKV.exe for creating those files, and to compare the results with selfmade implementations | | V_REAL/???? | Real Video(TM) | The stream is one of the Real Video(TM) video streams listed below. Source for the codec names are from Karl Lillevold on Doom9 [51]. The CodecPrivate element contains a "real_video_props_t" structure in Big Endian byte order as found in librmff [52]. |

| V_REAL/RV10 | RealVideo 1.0 aka RealVideo 5 | Individual slices from the Real container are combined into a single frame. | | V_REAL/RV20 | RealVideo G2 and RealVideo G2+SVT | Individual slices from the Real container are combined into a single frame. | | V_REAL/RV30 | RealVideo 8 | Individual slices from the Real container are combined

into a single frame. | | V_REAL/RV40 | rv40 : RealVideo 9 | Individual slices from the Real container are combined into a single frame. |

| | V_QUICKTIME | Video taken from QuickTime(TM) files | Several codecs as stored in QuickTime, e.g. Sorenson or Cinepak. The CodecPrivate contains all additional data that is stored in the 'stsd' (sample description) atom in the QuickTime file *after* the mandatory video descriptor structure (starting with the size and FourCC fields). For an explanation of the QuickTime file format read Apple's PDF on QuickTime [53]. |

| V_THEORA | Theora | The private data contains the first three Theora packets in order. The lengths of the packets precedes them. The actual layout is:

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

- Bytes 2..n: lengths of the first '"#p"' packets, coded in Xiph-style lacing [54]. The length of the last packet is the length of the CodecPrivate block minus the lengths coded in these bytes minus one.

- Bytes n+1..: The Theora identification header, followed by the commend header followed by the codec setup header. Those are described in the Theora specs [55].

| | V_PRORES | Apple ProRes | The private data contains the fourcc as found in MP4 movies:

- apch: ProRes 422 High Quality

- apcn: ProRes 422 Standard Definition

- apcs: ProRes 422 LT

- apco: ProRes 422 Proxy

- ap4h: ProRes 4444

this page for more technical details on ProRes [56]

| | Audio | | A_MPEG/L3 | MPEG Audio 1, 2, 2.5 Layer III |

The private data is void. The data contain everything needed for playback in the MPEG Audio header of each frame.

Corresponding ACM wFormatTag : 0x0055

| | A_MPEG/L2 | MPEG Audio 1, 2 Layer II |

The private data is void. The data contain everything needed for playback in the MPEG Audio header of each frame.

Corresponding ACM wFormatTag : 0x0050

| | A_MPEG/L1 | MPEG Audio 1, 2 Layer I |

The private data is void. The data contain everything needed for playback in the MPEG Audio header of each frame.

Corresponding ACM wFormatTag : 0x0050

| | A_PCM/INT/BIG | PCM Integer Big Endian |

The private data is void. The bitdepth has to be read and set from KaxAudioBitDepth element

Corresponding ACM wFormatTag : ???

| | A_PCM/INT/LIT | PCM Integer Little Endian |

The private data is void. The bitdepth has to be read and set from KaxAudioBitDepth element

Corresponding ACM wFormatTag : 0x0001

| | A_PCM/FLOAT/IEEE | Floating Point, IEEE compatible |

The private data is void. The bitdepth has to be read and set from KaxAudioBitDepth element (32 bit in most cases). The float are stored in little endian order (most common float format).

Corresponding ACM wFormatTag : 0x0003

| | A_MPC | MPC (musepack) SV8 | The main developer for musepack has requested that we wait until the SV8 framing has been fully defined for musepack before defining how to store it in Matroska. | |

A_AC3

A_AC3/BSID9

A_AC3/BSID10

| (Dolby[TM]) AC3 |

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

AC3/BSID9 and AC3/BSID10 (DolbyNet) : The ac3 frame header has, similar to the mpeg-audio header a version field. Normal ac3 is defiened as bitstream id 8 (5 Bits, numbers are 0-15). Everything below 8 is still compatible with all decoders that handle 8 correctly. Everything higher are additions that break decoder compatibility. For the samplerates 24kHz (00); 22,05kHz (01) and 16kHz (10) the BSID is 9 For the samplerates 12kHz (00); 11,025kHz (01) and 8kHz (10) the BSID is 10

| | A_ALAC | ALAC (Apple Lossless Audio Codec) | The private data contains ALAC's magic cookie (both the codec specific configuration as well as the optional channel layout information). Its format is described in ALAC's official source code [57]. |

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

| A_DTS/EXPRESS | Digital Theatre System Express | DTS Express (a.k.a. LBR) audio streams. The private data is void. Corresponding ACM wFormatTag : 0x2001 |

| A_DTS/LOSSLESS | Digital Theatre System Lossless | DTS Lossless audio that does not have a core substream. The private data is void. Corresponding ACM wFormatTag : 0x2001 |

| A_VORBIS | Vorbis | The private data contains the first three Vorbis packet in order. The lengths of the packets precedes them. The actual layout is: Byte 1: number of distinct packets '"#p"' minus one inside the CodecPrivate block. This MUST be '2' for current (as of 2016-07-08) Vorbis headers. Bytes 2..n: lengths of the first '"#p"' packets, coded in Xiph-style lacing [58]. The length of the last packet is the length of the CodecPrivate block minus the lengths coded in these bytes minus one. Bytes n+1..: The Vorbis identification header [59], followed by the Vorbis comment header [60] followed by the codec setup header [61]. |

| A_FLAC | FLAC (Free Lossless Audio Codec) [62] | The private data contains all the header/metadata packets before the first data packet. These include the first header packet containing only the word "fLaC" as well as all metadata packets. | | A_REAL/???? | Realmedia Audio codecs | The stream contains one of the following

audio codecs. In each case the CodecPrivate element contains either the "real_audio_v4_props_t" or the "real_audio_v5_props_t" structure (differentiated by their "version" field; Big Endian byte order) as found in librmff [63]. |

| A_REAL/14_4 | Real Audio 1 | | A_REAL/28_8 | Real Audio 2 | | A_REAL/COOK | Real Audio Cook Codec (codename: Gecko) | | A_REAL/ SIPR | Sipro Voice Codec | | A_REAL/RALF | Real Audio Lossless Format | | A_REAL/ATRC | Sony Atrac3 Codec |

| | A_MS/ACM | Microsoft(TM) Audio Codec Manager (ACM) | The private data contains the ACM structure WAVEFORMATEX including the extra private bytes, as defined by Microsoft [64]. The data are stored in little endian format (like on IA32 machines). | | A_AAC/?????/??? | AAC Profile Audio | The stream complies with, and uses the CodecID for, one of the AAC profiles listed below. AAC audio always uses wFormatTag 0xFF |

| A_AAC/MPEG2/MAIN | MPEG2 Main Profile | The private data is void. Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal matroska frame based muxing scheme is applied. | | A_AAC/MPEG2/LC | Low Complexity | The private data is void. Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal matroska frame based muxing scheme is applied. | | A_AAC/MPEG2/LC/SBR | Low Complexity with Spectral Band Replication | The private data is void. Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal matroska frame based muxing scheme is applied. | | A_AAC/MPEG2/SSR | Scalable Sampling Rate | The private data is void. Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal matroska frame based muxing scheme is applied. | | A_AAC/MPEG4/MAIN | MPEG4 Main Profile | The private data is void. Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal matroska frame based muxing scheme is applied. | | A_AAC/MPEG4/LC | Low Complexity | The private data is void. Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal matroska frame based muxing scheme is applied. | | A_AAC/MPEG4/LC/SBR | Low Complexity with Spectral Band Replication | The private data is void. Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal matroska frame based muxing scheme is applied. | | A_AAC/MPEG4/SSR | Scalable Sampling Rate | The private data is void. Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS

headers and normal matroska frame based muxing scheme is applied. | | A_AAC/MPEG4/LTP | Long Term Prediction | The private data is void. Channel number and sample rate have to be read from the corresponding audio element. Audio stream is stripped from ADTS headers and normal matroska frame based muxing scheme is applied. |

| | A_QUICKTIME | Audio taken from QuickTime(TM) files | Several codecs as stored in QuickTime, e.g. QDesign Music v1 or v2. The CodecPrivate contains all additional data that is stored in the 'stsd' (sample description) atom in the QuickTime file *after* the mandatory sound descriptor structure (starting with the size and FourCC fields). For an explanation of the QuickTime file format read Apple's PDF on QuickTime [65]. |

| A_QUICKTIME/???? | QuickTime audio codecs | This CodecID is deprecated in favor of A_QUICKTIME (without a trailing codec name). Otherwise the storage is identical; see A_QUICKTIME for details. |

A_QUICKTIME/QDMC | QDesign Music |

| A_QUICKTIME/QDM2 | QDesign Music v2 |

| A_TTA1 | The True Audio [66] lossles audio compressor | TTA format description [67] Each frame is kept intact, including the CRC32. The header and seektable are dropped. The private data is void. SamplingFrequency, Channels and BitDepth are used in the TrackEntry. wFormatTag = 0x77A1 |

| A_WAVPACK4 | WavPack [68] lossles audio compressor | The Wavpack packets consist of a stripped header followed by the frame data. For multi-track (> 2 tracks) a frame consists of many packets. For hybrid files (lossy part + correction part), the correction part is stored in an additional block (level 1). For more details, check the WavPack muxing description [69]. | | Subtitle | | S_TEXT/UTF8 | UTF-8 Plain Text | Basic text subtitles. For more information, please look at the Subtitle specifications [70]. | | S_TEXT/SSA | Subtitles Format | The [Script Info] and [V4 Styles] sections are stored in the codecprivate. Each event is stored in its own Block. For more information, please read the specs for SSA/ASS [71]. | | S_TEXT/ASS | Advanced Subtitles Format | The [Script Info] and [V4 Styles] sections are stored in the codecprivate. Each event is stored in its own Block. For more information, please read the specs for SSA/ASS [72]. | | S_TEXT/USF | Universal Subtitle Format | This is mostly defined, but not typed out yet. It will first be available on the USF specs page [73]. | | S_TEXT/WEBVTT | Web Video Text Tracks Format (WebVTT) | Advanced text subtitles. For more information about the storage please look at the WebVTT in Matroska specifications [74]. | | S_IMAGE/BMP | Bitmap | Basic image based subtitle format;

The subtitles are stored as images, like in the DVD. The timestamp in the block header of matroska indicates the start display time, the duration is set with the Duration element. The full data for the subtitle bitmap is stored in the Block's data section. | | S_VOBSUB | VobSub subtitles | The same subtitle format used on DVDs. Supoprted is only format version 7 and newer. VobSubs consist of two files, the .idx containing information, and the .sub, containing the actual data. The .idx file is stripped of all empty lines, of all comments and of lines beginning with "alt:" or "langidx:". The line beginning with "id:" SHOULD be transformed into the appropriate Matroska track language element and is discarded. All remaining lines but the ones containing timestamps and file positions are put into the "CodecPrivate" element. For each line containing the timestamp and file position data is read from the appropriate position in the .sub file. This data consists of a MPEG program stream which in turn contains SPU packets. The MPEG program stream data is discarded, and each SPU packet is put into one Matroska frame. | | S_KATE | Karaoke And Text Encapsulation | A subtitle format developped for ogg. The mapping for Matroska is described on the Xiph wiki [75]. As for Theora and Vorbis, Kate headers are stored in the private data as xiph-laced packets. | | Buttons | | B_VOBBTN | VobBtn Buttons | Based on MPEG/VOB PCI packets [76]. The file contains a header consisting of the string "butonDVD" followed by the width and height in pixels (16 bits integer each) and 4 reserved bytes. The rest is full PCI packets [77]. |

To be supported later :

'V_MSWMV'; Video, Microsoft Video

'V_INDEO5'; Video, Indeo 5; transmuxed from AVI or created using VfW codec

'V_MJPEG'; Video, MJpeg codec (lossy mode, general)

'V_MJPEG2000'; Video, MJpeg 2000

'V_MJPEG2000LL'; Video, MJpeg Lossless

'V_DV'; Video, DV Video, type 1 (audio and video mixed)

'V_TARKIN'; Video, Ogg Tarkin

'V_ON2VP4'; Video, ON2, VP4

'V_ON2VP5'; Video, ON2, VP5

'V_3IVX'; Video, 3ivX (is D4 decoder downwards compatible?)

'V_HUFFYUV'; Video, HuffYuv, lossless; auch als VfW moeglich

'V_COREYUV'; Video, CoreYuv, lossless; auch als VfW moeglich

'V_RUDUDU'; Nicola's Rududu Wavelet codec

...... to be continued--- #Chapters

264.1. 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.

- o 00000ms - 05000ms : Intro
- o 05000ms - 25000ms : Before the crime
- o 25000ms - 27500ms : The crime
- o 27500ms - 38000ms : The killer arrested
- o 38000ms - 43000ms : Credits

This would translate in the following matroska form :

| Chapters | EditionEntry | ChapterAtom | ChapterUID | 0x123456 | ChapterTimeStart | 0 ns | ChapterTimeEnd | 5,000,000 ns | ChapterDisplay | ChapterString | Intro | ChapterLanguage | eng | ChapterAtom | ChapterUID | 0x234567 | ChapterTimeStart | 5,000,000 ns | ChapterTimeEnd | 25,000,000 ns | ChapterDisplay | ChapterString | Before the crime | ChapterLanguage | eng | ChapterDisplay | ChapterString | Avant le crime | ChapterLanguage | fra | ChapterAtom | ChapterUID | 0x345678 | ChapterTimeStart | 25,000,000 ns | ChapterTimeEnd | 27,500,000 ns | ChapterDisplay | ChapterString | The crime | ChapterLanguage | eng | ChapterDisplay | ChapterString | Le crime | ChapterLanguage | fra | ChapterAtom | ChapterUID | 0x456789 | ChapterTimeStart | 27,500,000 ns | ChapterTimeEnd | 38,000,000 ns | ChapterDisplay | ChapterString | After the crime | ChapterLanguage | eng | ChapterDisplay | ChapterString | Apres le crime | ChapterLanguage | fra | ChapterAtom | ChapterUID | 0x456789 | ChapterTimeStart | 38,000,000 ns | ChapterTimeEnd | 43,000,000 ns | ChapterDisplay | ChapterString | Credits | ChapterLanguage | eng | ChapterDisplay | ChapterString | Generique | ChapterLanguage | fra |

264.2. Example 2 : nested chapters

In this example an (existing) album is split into different chapters, and one of them contain another splitting.

264.2.1. The Micronauts "Bleep To Bleep"

- 00:00 - 12:28 : Baby Wants To Bleep/Rock
  - 00:00 - 04:38 : Baby wants to bleep (pt.1)
  - 04:38 - 07:12 : Baby wants to rock
  - 07:12 - 10:33 : Baby wants to bleep (pt.2)
  - 10:33 - 12:28 : Baby wants to bleep (pt.3)
- 12:30 - 19:38 : Bleeper_O+2
- 19:40 - 22:20 : Baby wants to bleep (pt.4)
- 22:22 - 25:18 : Bleep to bleep
- 25:20 - 33:35 : Baby wants to bleep (k)
- 33:37 - 44:28 : Bleeper

| Chapters | EditionEntry | ChapterAtom | ChapterUID | 0x654321 | ChapterTimeStart | 0 ns | ChapterTimeEnd | 748,000,000 ns | ChapterDisplay | ChapterString | Baby wants to bleep/rock | ChapterAtom | ChapterUID | 0x123456 | ChapterTimeStart | 0 ns | ChapterTimeEnd | 278,000,000 ns | ChapterDisplay | ChapterString | Baby wants to bleep (pt.1) | ChapterAtom | ChapterUID | 0x234567 | ChapterTimeStart | 278,000,000 ns | ChapterTimeEnd | 432,000,000 ns | ChapterDisplay | ChapterString | Baby wants to rock | ChapterAtom | ChapterUID | 0x345678 | ChapterTimeStart | 432,000,000 ns | ChapterTimeEnd | 633,000,000 ns | ChapterDisplay | ChapterString | Baby wants to bleep (pt.2) | ChapterAtom | ChapterUID | 0x456789 | ChapterTimeStart | 633,000,000 ns | ChapterTimeEnd | 748,000,000 ns | ChapterDisplay | ChapterString | Baby wants to bleep (pt.3) | ChapterAtom | ChapterUID | 0x567890 | ChapterTimeStart | 750,000,000 ns | ChapterTimeEnd | 1,178,500,000 ns | ChapterDisplay | ChapterString | Bleeper_O+2 | ChapterAtom | ChapterUID | 0x678901 | ChapterTimeStart | 1,180,500,000 ns | ChapterTimeEnd | 1,340,000,000 ns | ChapterDisplay | ChapterString | Baby wants to bleep (pt.4) | ChapterAtom | ChapterUID | 0x789012 | ChapterTimeStart | 1,342,000,000 ns | ChapterTimeEnd | 1,518,000,000 ns | ChapterDisplay | ChapterString | Bleep to bleep | ChapterAtom |

ChapterUID | 0x890123 | ChapterTimeStart | 1,520,000,000 ns | ChapterTimeEnd | 2,015,000,000 ns | ChapterDisplay | ChapterString | Baby wants to bleep (k) | ChapterAtom | ChapterUID | 0x901234 | ChapterTimeStart | 2,017,000,000 ns | ChapterTimeEnd | 2,668,000,000 ns | ChapterDisplay | ChapterString | Bleeper |

264.3. Edition and chapter flags

264.3.1. Chapter flags

There are two important flags that apply to chapter atoms: _enabled_ and _hidden_. The effect of those flags always applies to child atoms of an atom affected by that flag.

For example: Let's assume a parent atom with flag _hidden_ set to _true_; that parent contains two child atom, the first with _hidden_ set to _true_ as well and the second child with the flag either set to _false_ or not present at all (in which case the default value applies, and that again is _false_).

As the parent is hidden all of its children are initially hidden as well. However, when a control track toggles the parent's _hidden_ flag to _false_ then only the the parent and its second child will be visible. The first child's explicitely set flag retains its value until its value is toggled to _false_ by a control track.

Corresponding behavior applies to the _enabled_ flag.

264.3.2. Edition flags

The edition's _hidden_ flag behaves much the same as the chapter's _hidden_ flag: if an edition is hidden then none of its children SHALL be visible, no matter their own _hidden_ flags. If the edition is toggled to being visible then the chapter atom's _hidden_ flags decide whether or not the chapter is visible.

264.4. 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. The private data depend on the type of menu system (stored in ChapProcessPrivate), idem for the data in the chapters (stored in ChapProcessData).

264.4.1. Matroska Script (0)

This is the case when ChapProcessCodecID [78] = 0. This is a script language build for Matroska purposes. The inspiration comes from ActionScript, javascript and other similar scripting languages. The commands are stored as text commands, in UTF-8. The syntax is C like, with commands spanned on many lines, each terminating with a ";". You can also include comments at the end of lines with "//" or comment many lines using "/* */". The scripts are stored in ChapProcessData. For the moment ChapProcessPrivate is not used.

The one and only command existing for the moment is "GotoAndPlay(ChapterUID);". As the same suggests, it means that when this command is encountered, the playback SHOULD jump to the Chapter specified by the UID and play it.

264.4.2. DVD menu (1)

This is the case when ChapProcessCodecID [79] = 1. Each level of a chapter corresponds to a logical level in the DVD system that is stored in the first octet of the ChapProcessPrivate. This DVD hierarchy is as follows:

| ChapProcessPrivate | DVD Name | Hierarchy | Commands Possible | Comment | | 0x30 | SS | DVD domain | - | First Play, Video Manager, Video Title | | 0x2A | LU | Language Unit | - | Contains only PGCs | | 0x28 | TT | Title | - | Contains only PGCs | | 0x20 | PGC | Program Group Chain (PGC) | * | | 0x18 | PG | Program 1 | Program 2 | Program 3 | - | | 0x10 | PTT | Part Of Title 1 | Part Of Title 2 | - | Equivalent to the chapters on the sleeve. | | 0x08 | CN | Cell 1 | Cell 2 | Cell 3 | Cell 4 | Cell 5 | Cell 6 | - |

You can also recover wether a Segment is a Video Manager (VMG), Video Title Set (VTS) or Video Title Set Menu (VTSM) from the ChapterTranslateID [80] element found in the Segment Info. This field uses 2 octets as follows:

1. Domain Type: 0 for VMG, the domain number for VTS and VTSM

2. Domain Value: 0 for VMG and VTSM, 1 for the VTS source.

For instance, the menu part from VTS_01_0.VOB would be coded [1,0] and the content part from VTS_02_3.VOB would be [2,1]. The VMG is always [0,0]

The following octets of ChapProcessPrivate are as follows:

| Octet 1 | DVD Name | Following Octets | | 0x30 | SS | Domain name code (1: 0x00= First play, 0xC0= VMG, 0x40= VTSM, 0x80= VTS) + VTS(M) number (2) | | 0x2A | LU | Language code (2) + Language extension (1) | | 0x28 | TT | global Title number (2) + corresponding TTN of the VTS (1) | | 0x20 | PGC | PGC number (2) + Playback Type (1) + Disabled User Operations (4) | | 0x18 | PG | Program number (2) | | 0x10 | PTT | PTT-chapter number (1) | | 0x08 | CN | Cell number [VOB ID(2)][Cell ID(1)][Angle Num(1)] |

If the level specified in ChapProcessPrivate is a PGC (0x20), there is an octet called the Playback Type, specifying the kind of PGC defined:

o 0x00: entry only/basic PGC

o 0x82: Title+Entry Menu (only found in the Video Manager domain)

o 0x83: Root Menu (only found in the VTSM domain)

o 0x84: Subpicture Menu (only found in the VTSM domain)

o 0x85: Audio Menu (only found in the VTSM domain)

o 0x86: Angle Menu (only found in the VTSM domain)

o 0x87: Chapter Menu (only found in the VTSM domain)

The next 4 following octets correspond to the User Operation flags [81] in the standard PGC. When a bit is set, the command SHOULD be disabled.

ChapProcessData contains the pre/post/cell commands in binary format as there are stored on a DVD. There is just an octet preceeding these data to specify the number of commands in the element. As follows: [# of commands(1)][command 1 (8)][command 2 (8)][command 3 (8)].

More information on the DVD commands and format on DVD-replica [82], where we got most of the info about it. You can also get information on DVD from the DVDinfo project [83].---

265. Subtitles

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

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

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

- Any Matroska file containing only subtitles SHOULD use the extension ".mks".

- As a general rule of thumb for all codecs, information that is global to an entire stream SHOULD be stored in the CodecPrivate element.

- Start and stop timecodes that are used in a timecodes native storage format SHOULD be removed when being placed in Matroska as they could interfere if the file is edited afterwards. Instead, the Blocks timecode and Duration SHOULD be used to say when the timecode is displayed.

- Because a "subtitle" stream is actually just an overlay stream, anything with a transparency layer could be use, including video.

266. Images Subtitles

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

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

The .IFO file will not be used at all.

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

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

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

Here is an example .IDX file:

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

# Settings

# Original frame size
size: 720x480

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

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

# Alpha blending
alpha: 100%

# Smoothing for very blocky images (use OLD for no filtering)
```

```
smooth: OFF

# In millisecs
fadein/out: 50, 50

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

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

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

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

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

# Language index in use
langidx: 0

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

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

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

Finally, the "timestamp" will be used to set the Block's timecode. Once it is set there, there is no need for it to be stored here.

Also, as it may interfere if the file is edited, it SHOULD NOT be stored here.

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

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

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

267. SRT Subtitles

SRT is perhaps the most basic of all subtitle formats.

It consists of four parts, all in text..

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

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

Here is an example SRT file:

|

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

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

|

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

The same is repeated for the next subtitle.

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

268. SSA/ASS Subtitles

SSA stands for Sub Station Alpha. It's the file format used by the popular subtitle editor, SubStation Alpha [84]. This format is widely used by fansubbers.

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

For detailed information on SSA/ASS, see the SSA specs [85]. It includes an SSA specs description and the avanced features added by ASS format (standing for Advanced SSA). Because SSA and ASS are so similar, they are treated the same here.

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

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

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

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

For example this :

|

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

|

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

|

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

|

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

Now, how are they stored in Matroska ?

- o All text is converted to UTF-8* All the headers are stored in CodecPrivate (Script Info and the Styles list)* Start & End field are used to set TimeStamp and the BlockDuration element. the data stored is :* Events are stored in the Block in this order: ReadOrder, Layer, Style, Name, MarginL, MarginR, MarginV, Effect, Text (Layer comes from ASS specs ... it’s empty for SSA.) "ReadOrder field is needed for the decoder to be able to reorder the streamed samples as they were placed originally in the file."

Here is an example of an SSA file.

| [Script Info] ; This is a Sub Station Alpha v4 script. ; For Sub Station Alpha info and downloads, ; go to http://www.eswat.demon.co.uk/ ; or email kotus@eswat.demon.co.uk [87] Title: Wolf’s rain 2 Original Script: Anime-spirit Ishin-francais Original Translation: Coolman Original Editing: Spikewolfwood

Original Timing: Lord_alucard Original Script Checking: Spikewolfwood ScriptType: v4.00 Collisions: Normal PlayResY: 1024 PlayDepth: 0 Wav: 0, 128697,D:\Alex\Anime- Fansub -- TAFF -\Wolf's Rain\WR_-_02_Wav.wav Wav: 0, 120692,H:\team truc\WR_-_02.wav Wav: 0, 116504,E:\sub\wolf's_rain\WOLF'S RAIN 02.wav LastWav: 3 Timer: 100,0000

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

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

Here is what would be placed into the CodecPrivate element.

| [Script Info] ; This is a Sub Station Alpha v4 script. ; For Sub Station Alpha info and downloads, ; go to http://www.eswat.demon.co.uk/ ; or email kotus@eswat.demon.co.uk [89] Title: Wolf's rain 2 Original Script: Anime-spirit Ishin-francais Original Translation: Coolman Original Editing: Spikewolfwood Original Timing: Lord_alucard Original Script Checking: Spikewolfwood ScriptType: v4.00 Collisions: Normal PlayResY: 1024 PlayDepth: 0 Wav: 0, 128697,D:\Alex\Anime- Fansub -- TAFF -\Wolf's Rain\WR_-_02_Wav.wav Wav: 0, 120692,H:\team truc\WR_-_02.wav Wav: 0, 116504,E:\sub\wolf's_rain\WOLF'S RAIN 02.wav LastWav: 3 Timer: 100,0000

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

And here are the two blocks that would be generated.

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

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

Block’s timecode: 00:02:42.420 BlockDuration: 00:00:01.730

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

269. USF Subtitles

Under construction

270. WebVTT

The "Web Video Text Tracks Format" (short: WebVTT) is developed by the World Wide Web Consortium (W3C) [90]. Its specifications are freely available [91].

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

- o Consistency: store data in a similar way to other subtitle codecs

- o Simplicity: making decoding and remuxing as easy as possible for existing infrastructures

- o Completeness: keeping as much data as possible from the original WebVTT file

270.1. Storage of WebVTT in Matroska

270.1.1. CodecID: codec identification

The CodecID to use is S_TEXT/WEBVTT.

270.1.2. CodecPrivate: storage of gloal WebVTT blocks

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

270.1.3. Storage of non-global WebVTT blocks

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

270.1.4. Storage of Cues in Matroska blocks

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

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

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

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

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

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

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

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

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

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

270.2. Examples of transformation

Here's an example how a WebVTT is transformed.

270.2.1. Example WebVTT file

Let's take the following example file:

270.2.2. CodecPrivate

The resulting CodecPrivate element will look like this:

270.2.3. Storage of Cue 1

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

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

270.2.4. Storage of Cue 2

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

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

270.2.5. Storage of Cue 3

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

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

270.2.6. Storage of Cue 4

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

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

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

270.3. Storage of WebVTT in Matroska vs. WebM

Note: the storage of WebVTT in Matroska is not the same as the design document for storage of WebVTT in WebM. There are several reasons for this including but not limited to: the WebM document is old (from February 2012) and was based on an earlier draft of WebVTT and ignores several parts that were added to WebVTT later; WebM does still not support subtitles at all [92]; the proposal suggests splitting the information across multiple tracks making demuxer’s and remuxer’s life very difficult.---

271. Tagging

When a Tag is nested within another Tag, the nested Tag becomes an attribute of the base tag. For instance, if you wanted to store the dates that a singer used certain addresses for, that singer being the lead singer for a track that included multiple bands simultaneously, then your tag tree would look something like this: Targets - TrackUID BAND - LEADPERFORMER -- ADDRESS --- DATE --- DATEEND -- ADDRESS --- DATE In this way, it becomes possible to store any Tag as attributes of another tag. Multiple items SHOULD never be stored as a list in a single TagString. If there is more than one tag of a certain type to be stored, then more than one SimpleTag SHOULD be used. For authoring Tags outside of EBML, the following XML syntax is proposed [93] used in mkvmerge [94]. Binary data SHOULD be stored using BASE64 encoding if it is being stored at authoring time.

271.1. Why official tags matter

There is a debate between people who think all tags SHOULD be free and those who think all tags SHOULD be strict. If you look at this page you will realise we are in between.

Advanced-users application might let you put any tag in your file. But for the rest of the applications, they usually give you a basic list of tags you can use. Both have their needs. But it’s usually a bad idea to use custom/exotic tags because you will probably be the only person to use this information even though everyone else could benefit from it. So hopefully when someone wants to put information in one’s file, they will find an official one that fit them and hopefully use it ! If it’s not in the list, this person can contact us any time for addition of such a missing tag. But it doesn’t mean it will be accepted... Matroska files are not meant the become a whole database of people who made costumes for a film. A website would be better for that... It’s hard to define what SHOULD be in and what doesn’t make sense in a file. So we’ll treat each request carefully.

We also need an official list simply for developpers to be able to display relevant information in their own design (if they choose to support a list of meta-information they SHOULD know which tag has the wanted meaning so that other apps could understand the same meaning).

271.2. Tag translations

To be able to save tags from other systems to Matroska we need to translate them to our system. There is a translation table on our site [95].

271.3. Tag Formatting

- The TagName SHOULD always be written in all capital letters and contain no space.

- The fields with dates SHOULD have the following format: YYYY-MM-DD HH:MM:SS.MSS YYYY = Year, -MM = Month, -DD = Days, HH = Hours, :MM = Minutes, :SS = Seconds, :MSS = Milliseconds. To store less accuracy, you remove items starting from the right. To store only the year, you would use, "2004". To store a specific day such as May 1st, 2003, you would use "2003-05-01".

- Fields that require a Float SHOULD use the "." mark instead of the "," mark. To display it differently for another local, applications SHOULD support auto replacement on display. Also, a thousandths separator SHOULD NOT be used.

- For currency amounts, there SHOULD only be a numeric value in the Tag. Only numbers, no letters or symbols other than ".". For instance, you would store "15.59" instead of "$15.59USD".

271.4. Target types

The TargetType element allows tagging of different parts that are inside or outside a given file. For example in an audio file with one song you could have information about the album it comes from and even the CD set even if it’s not found in the file.

For application to know what kind of information (like TITLE) relates to a certain level (CD title or track title), we also need a set of official TargetType names. For now audio and video will have different values & names. That also means the same tag name can have different meanings depending on where it is (otherwise we would end up with 15 TITLE_ tags).

An upper level value tag applies to the lower level. That means if a CD has the same artist for all tracks, you just need to set the ARTIST tag at level 50 (ALBUM) and not to each TRACK (but you can). That also means that if some parts of the CD have no known ARTIST the value MUST be set to nothing (a void string "").

When a level doesn't exist it MUST NOT be specified in the files, so that the TOTAL_PARTS and PART_NUMBER elements match the same levels.

Here is an example of how these "organizational" tags work: If you set 10 TOTAL_PARTS to the ALBUM level (40) it means the album contains 10 lower parts. The lower part in question is the first lower level that is specified in the file. So if it's TRACK (30) then that means it contains 10 tracks. If it's MOVEMENT (20) that means it's 10 movements, etc.

271.5. Official tags

The following is a complete list of the supported Matroska Tags. While it is possible to use Tag names that are not listed below, this is not reccommended as compatability will be compromised. If you find that there is a Tag missing that you would like to use, then please contact the Matroska team for its inclusion in the specifications before the format reaches 1.0.

271.6. Notes

- In the Target list, a logicial OR is applied on all tracks, a logicial OR is applied on all chapters. Then a logical AND is applied between the Tracks list and the Chapters list to know if an element belongs to this Target.

With the rise of Media Centers and even programs to manage large amounts of audio files, it's becoming necessary to visualize your files easily, not just browse by names. It is also a lot nicer to browse for the user. Matroska supports attachments and they can be used for cover arts. This document defines a set of guidelines for coders and file creators to add cover arts correctly in Matroska files. These guidelines will ensure the user experience is good and consistent wherever you put your files.

The pictures SHOULD only use the JPEG and PNG picture formats.

There can be 2 different cover for a movie/album. A portrait one (like a DVD case) and a landscape one (like a banner ad for example, looking better on a wide screen).

There can be 2 versions of the same cover, the normal one and the small one. The dimension of the normal one SHOULD be 600 on the smallest side (eg 960x600 for landscape and 600x800 for portrait, 600x600 for square). The dimension of the small one SHOULD be 120 (192x120 or 120x160).

The way to differentiate between all these versions is by the filename. The default filename is cover.(png/jpg) for backward compatibility reasons. That is the "big" version of the file (600) in square or portrait mode. It SHOULD also be the first file in the attachments. The smaller resolution SHOULD be prefixed with "small_", ie small_cover.(jpg/png). The landscape variant SHOULD be suffixed with "_land", ie cover_land.jpg. The filenames are case sensitive and SHOULD all be lower case.

- cover.jpg (portrait/square 600)

- small_cover.png (portrait/square 120)

- cover_land.png (landscape 600)

- small_cover_land.jpg (landscape 120)

There is a sample file [96] available to test player compatibility or to demonstrate the use of cover art in Matroska files.---

272. Matroska Streaming

There exist multiple ways to stream a content. The term streaming itself is very vague. It means reading a file stored on a server. But the server could be very distant or very close. The transport system and the protocol used for streaming makes the whole difference.

In the case of Matroska, there are mostly 2 different kinds of stream: file access and live streaming.

273. File Access

File access can simply be reading a file located on your computer, but also accessing it from an HTTP (web) server or CIFS (windows share) server. All these protocols are usually safe from reading errors and seeking in the stream is possible. On other hand when the file is stored far away or on a slow server, seeking can be an expensive operation and SHOULD be avoided. That’s why we set a few guidelines [97] that, when followed, help reduce the number of seeking for regular playback and also have the playback start quickly without a lot of data needed to read first (like the Cues (index), Attachments or Meta Seek of all the Clusters).

Matroska having a small overhead, it is well suited for storing music/videos on file servers without having a big impact on the bandwidth used. It doesn’t require to load the index before playing (the index can be loaded only when seeking is requested the first time), so playback can start very quickly too.

274. Live Streaming

Live streaming is the equivalent of TV broadcasting on the internet. There are 2 families of servers for that. The RTP/RTSP ones and the HTTP servers. Matroska is not meant to be used over RTP. RTP already has timing and channel mechanisms that would wasted if doubled in Matroska. On the other hand live streaming of Matroska over HTTP (or any other plain protocol based on TCP) is very possible.

A live Matroska stream is different than a file, because it may have no known end (only when the client disconnects). For that the Segment MUST use the "unknown" size (all 1s in the size). The other option would be to concatenate Segments with known sizes one after the other. This solution allows a change of codec/resolution between each segment which can be useful in some cases (switch between 4:3 and 16:9 in some TV programs for example).

The Segment(s) being continuous, certain elements like Meta Seek, Cues, Chapters, Attachments MUST NOT be used in this context.

On the player side, it is possible to detect that a stream is not seekable. If the stream does not have a Meta Seek list or a Cues list at the beginning of the stream, it SHOULD be considered as non seekable. Even though it's still theoretically possible to seek blindly forward in the stream, if the server supports it.

In the context of a live radio or even web TV it is possible to "Tag" the content that is currently playing. The Tags level 1 element can be placed between Clusters each time necessary. In that case, the new Tags found MUST reset the previously encountered tags and use the new values instead (be they empty).

275. Menu Specifications

276. Introduction

This document is a _draft of the Menu system_ that will be the default one in Matroska. As it will just be composed of a Control Track, it will be seen as a "codec" and could be replaced later by something else if needed.

A menu is like what you see on DVDs, when you have some screens to select the audio format, subtitles or scene selection.

277. Requirements

What we'll try to have is a system that can do almost everything done on a DVD, or more, or better, or drop the unused features if necessary.

As the name suggests, a Control Track is a track that can control the playback of the file and/or all the playback features. To make it as simple as possible for players, the Control Track will just give orders to the player and get the actions associated with the highlights/hotspots.

277.1. Highlights/Hotspots

A hightlight is basically a rectangle/key associated with an action UID. When that rectangle/key is activated, the player send the UID of the action to the Control Track handler (codec). The fact that it can also be a key means that even for audio only files, a keyboard shortcut or button panel could be used for menues. But in that case, the hotspot will have to be associated with a name to display.

So this hightlight is sent from the Control Track to the Player. Then the player has to handle that highlight until it's disactivated (see Playback features [98])

The hightlight contains a UID of the action, a displayable name (UTF-8), an associated key (list of keys to be defined, probably up/down/left/right/select), a screen position/range and an image to display. The image will be displayed either when the user place the mouse over the rectangle (or any other shape), or when an option of the screen is selected (not activated). There could be a second image used when the option is activated. And there could be a third image that can serve as background. This way you could have a still image (like in some DVDs) for the menu and behind that image blank video (small bitrate).

When a highlight is activated by the user, the player has to send the UID of the action to the Control Track. Then the Control Track codec will handle the action and possibly give new orders to the player.

The format used for storing images SHOULD be extensible. For the moment we'll use PNG and BMP, both with alpha channel.

277.2. Playback features

All the following features will be sent from the Control Track to the Player :

- Jump to chapter (UID, prev, next, number)
- Disable all tracks of a kind (audio, video, subtitle)
- Enable track UID (the kind doesn’t matter)
- Define/Disable a highlight
- Enable/Disable jumping
- Enable/Disable track selection of a kind
- Select Edition ID (see chapters)
- Pause playback
- Stop playback
- Enable/Disable a Chapter UID
- Hide/Unhide a Chapter UID

All the actions will be writen in a normal Matroska track, with a timecode. A "Menu Frame" SHOULD be able to contain more that one action/highlight for a given timecode. (to be determined, EBML format structure)

277.3. Player requirements

Some players might not support the control track. That mean they will play the active/looped parts as part of the data. So I suggest putting the active/looped parts of a movie at the end of a movie. When a Menu-aware player encouter the default Control Track of a Matroska file, the first order SHOULD be to jump at the start of the active/looped part of the movie.

278. Working Graph

279. Ideas

280. Data Structure

As a Matroska side project, the obvious choice for storing binary data is EBML.---

281. Overhead

One of the way to compare containers is to analyze the overhead produced for the same raw data. The overhead is the amount of data added to these raw data, due to the internal structure of the format.

This document is only giving the overhead introduced by matroska. It is intended for the matroska team to better tune the format for efficiency. And also specify where matroska is best suited.

281.1. Example 1 - low bitrate audio

Low bitrate audio (like speex or vorbis) is usually using very small packets of data, all independent ones (no reference to previous or future frames). The bitrate can be as low as 8kbps (to 64kbps). For 8kbps you have 1kBps. If you cut that into 20ms parts, that makes approximately 100 bytes (to 1000 bytes) per part (packet). Low bitrate codec are usually tuned for real-time conferencing on limited bandwidth lines, so the VBR aspect is usually limited. That's why we will consider that bitrate to be constant, ie the size of packets.

bitrate	pkt size	No lacing	2 in lace	3 in lace	4 in lace	5 in lace	6 in lace	7 in lace
8kbps	100	1+1+4+0+100 (6%)	1+2+4+2+200 (4.5%)	1+2+4+3+300 (3.3%)	1+2+4+4+400 (2.8%)	1+2+4+5+500 (2.4%)	1+2+4+6+600 (2.2%)	1+2+4+7+700 (2%)
16kbps	200	1+2+4+0+200 (3.5%)	1+2+4+2+400 (2.3%)	1+2+4+3+600 (1.7%)	1+2+4+4+800 (1.4%)	1+2+4+5+1000 (1.2%)	1+2+4+6+1200 (1.1%)	1+2+4+7+1400 (1%)
24kbps	300	1+2+4+0+300 (2%)	1+2+4+3+600 (1.8%)	1+2+4+5+900 (1.3%)	1+2+4+7+1200 (1.2%)	1+2+4+9+1500 (1.1%)	1+2+4+11+1800 (1%)	1+2+4+13+2100 (0.95%)
32kbps	400	1+2+4+0+400 (1.8%)	1+2+4+3+800 (1.3%)	1+2+4+5+1200 (1%)	1+2+4+7+1600 (0.88%)	1+2+4+9+2000 (0.8%)	1+2+4+11+2400 (0.67%)	1+2+4+13+2800 (0.71%)
48kbps	600	1+2+4+0+600 (1%)	1+2+4+4+1200 (0.92%)	1+2+4+7+1800 (0.78%)	1+2+4+10+2400 (0.71%)	1+2+4+13+3000 (0.75%)	1+2+4+16+3600 (0.64%)	1+2+4+19+4200 (0.62%)
64kbps	800	1+2+4+0+800 (0.88%)	1+2+4+5+1600 (0.75%)	1+2+4+9+2400 (0.67%)	1+2+4+13+3200 (0.63%)	1+2+4+17+4000 (0.60%)	1+2+4+21+4800 (0.58%)	1+2+4+26+5600 (0.59%)

explanations : This is the overhead introduced by the Block level. There is more overhead in a matroska file, ie the Cluster head (that contains many Blocks) and the file header (containing various meta information).

the first number is the number is the size of the total Block+Data, ie element ID (1), size (1 to 8), Block head (4), lacing (0 to infinite), data

the second number is the pourcentage of overhead for each packet

In matroska, synchronisation and error recovery is done with a Cluster. A cluster contains 1 to many Blocks. So let’s see the effect of having 1 Block per Cluster, 2 Blocks per Cluster and 3 Blocks per Cluster. These are the worst cases where synchronisation is necessary often, ie every 20, 40 and 60ms (with 1 block only). A value of 1s is a pretty agressive case and is the minimum supported here (just to save me from a few calculus).

```
 +---------------------+
 | 1 Block per Cluster |
 +---------------------+
 |       bitrate       |
 |         ---         |
 |        8kbps        |
 |        16kbps       |
 |        24kbps       |
 |        32kbps       |
 |        48kbps       |
 |        64kbps       |
 +---------------------+

+----------------------+
| 2 Blocks per Cluster |
+----------------------+
|       bitrate        |
|         ---          |
|        8kbps         |
|        16kbps        |
|        24kbps        |
|        32kbps        |
|        48kbps        |
|        64kbps        |
+----------------------+

+----------------------+
| 3 Blocks per Cluster |
+----------------------+
|       bitrate        |
|         ---          |
|        8kbps         |
|        16kbps        |
|        24kbps        |
|        32kbps        |
|        48kbps        |
|        64kbps        |
+----------------------+
```

explanations : the third number is the max time between 2 error recoveries (re-sync) in milliseconds. This is based on the example of all frames have a 20ms granularity.

281.1.1. Conclusions

- o For the same error-recovery (1.2s, 2.4s) we clearly see that it’s better to put as much data in one Block and use lacing. In this case there will be 1 Block per Cluster. One of the drawback is that only the first packet of data has a timecode (one timecode every second).

- o If we agree that 2% overhead is OK for low bitrates (the same as for MP3), the minimum bitrate acceptable is between 8kbps and 16kbps (probably closer to 16 kbps). Actually this value depend on the block size and not really the bitrate.

- o For bitrates higher than 48kbps (or a fixed packet rate of 600 bytes) the overhead is very good even with one frame per packet.

281.2. Example 2 - medium bitrate audio

281.3. Example 3 - high bitrate audio

281.4. Example 4 - low bitrate video

Low bitrate video is, like low bitrate audio, the worst case for overhead. The range we will use here is 64 kbps to 256kbps. I think these are reasonable values for what can be considered as low video bitrate.

One of the major difference in video is that the number of frame per second is fixed (10 to 30 for low bitrates).

Another difference is that the frames usually depend on others to save some compression bits. One of the effect is that all frames don’t have the same size. You have key frames (I) that don’t depend on other frames, (P) frames that depend on an older frame and (B) frames that rely on an older and a future frame.

It is mandatory that all I frames have an actual timecode coming from the container, because the P and B frames will rely on that timecode for reference.

281.4.1. First example - all I frames (CBR)

With all I frames, the average data is easy to compute. For 64kbps=8kBps we have a frame=1/20s. That means each frame is 400 bytes big. So the results are similar to 32kbps audio.

Lacing SHOULD NOT be used for better seeking in the file. But it is still possible for use in this particular case.

```
 +-----------------------------------+
 | 1 I frame/Block per Cluster (5ms) |
 +-----------------------------------+
 |              bitrate              |
 |                ---                |
 |               64kbps              |
 |              128kbps              |
 |              256kbps              |
 +-----------------------------------+

+------------------------------------+
| 2 I frame/Block per Cluster (10ms) |
+------------------------------------+
|              bitrate               |
|                ---                 |
|               64kbps               |
|              128kbps               |
|              256kbps               |
+------------------------------------+

+------------------------------------+
| 4 I frame/Block per Cluster (20ms) |
+------------------------------------+
|              bitrate               |
|                ---                 |
|               64kbps               |
|              128kbps               |
|              256kbps               |
+------------------------------------+

+------------------------------------+
| 8 I frame/Block per Cluster (40ms) |
+------------------------------------+
|              bitrate               |
|                ---                 |
|               64kbps               |
|              128kbps               |
|              256kbps               |
+------------------------------------+
```

We clearly see that there is another more interresting table (lacing use is dropped) :

bitrate	pkt size	1 Block/C (50ms)	2 Blocks/C (100ms)	4 Blocks/C (200ms)	8 Blocks/C (400ms)	16 Blocks/C (800ms)	20 Blocks/C (1s)
64kbps	400	4+2+1*(1+2+4+400) (3.3%)	4+2+2*(1+2+4+400) (2.5%)	4+2+4*(1+2+4+400) (2.1%)	4+2+8*(1+2+4+400) (1.9%)	4+2+16*(1+2+4+400) (1.8%)	4+2+20*(1+2+4+400) (1.8%)
128kbps	800	4+2+1*(1+2+4+800) (1.63%)	4+2+2*(1+2+4+800) (1.3%)	4+2+4*(1+2+4+800) (1.06%)	4+2+8*(1+2+4+800) (0.97%)	4+2+16*(1+2+4+800) (0.92%)	4+2+20*(1+2+4+800) (0.91%)
256kbps	1200	4+2+1*(1+2+4+1200) (1.08%)	4+2+2*(1+2+4+1200) (0.83%)	4+2+4*(1+2+4+1200) (1.06%)	4+2+8*(1+2+4+1200) (0.65%)	4+2+16*(1+2+4+1200) (0.61%)	4+2+20*(1+2+4+1200) (0.61%)

Conslusions :

o In most cases, the best result is when 8 Blocks are packed in a Cluster. Bigger values don't improve the overhead much. It also seems to be a good value to stop using lacing in the low audio bitrate example. *So both lacing and clustering will be limited to 8 elements in libmatroska* on writing.

o In an agressive case like a 64kbps CBR video codec at 20 frames/sec can still achieve an overhead of less than 2% which is quite good.

o Upper 128kbps an overhead of less than 1% can always be achieved.

o Having 2 frames in a lace can improve substantially the overhead. But as I frames SHOULD always have a proper timecode, it is not possible to use this solution.

281.4.2. Second example - 1 I frame for 1 B frame

In this case, each frame has to be separated in a Block (no lacing possible). The reference timecode in matroska (of the previous frame) is written in a separate element ([BlockAddition][TimecodeReference]). So more overhead is introduced.

We will establish that the P frame is 3x smaller than the I frame (I hope this is a realistic case). That means that, at 20 frames per second, for 64 kbps the I frame is 600 bytes and the B frame is 200 bytes (2 frames are 2*400 bytes big).

bitrate	pkt size	2 Blocks/C (100ms)	4 Blocks/C (200ms)	8 Blocks/C (400ms)	16 Blocks/C (800ms)	20 Blocks/C (1s)
64kbps	600/200	4+2+(1+2+4+600)+(1+2+4+200+1+1+2) (3%)	4+2+2*((1+2+4+600)+(1+2+4+200+1+1+2)) (2.6%)	4+2+4*((1+2+4+600)+(1+2+4+200+1+1+2)) (2.4%)	4+2+8*((1+2+4+600)+(1+2+4+200+1+1+2)) (2.3%)	4+2+10*((1+2+4+600)+(1+2+4+200+1+1+2)) (2.3%)
128kbps	1200/400	4+2+(1+2+4+1200)+(1+2+4+400+1+1+2) (1.5%)	4+2+2*((1+2+4+1200)+(1+2+4+400+1+1+2)) (1.3%)	4+2+4*((1+2+4+1200)+(1+2+4+400+1+1+2)) (1.2%)	4+2+8*((1+2+4+1200)+(1+2+4+400+1+1+2)) (1.1%)	4+2+10*((1+2+4+1200)+(1+2+4+400+1+1+2)) (1.1%)
256kbps	1800/600	4+2+(1+2+4+1800)+(1+2+4+600+1+1+2) (1.01%)	4+2+2*((1+2+4+1800)+(1+2+4+600+1+1+2)) (0.88%)	4+2+4*((1+2+4+1800)+(1+2+4+600+1+1+2)) (0.81%)	4+2+8*((1+2+4+1800)+(1+2+4+600+1+1+2)) (0.78%)	4+2+10*((1+2+4+1800)+(1+2+4+600+1+1+2)) (0.78%)

Conclusions :

- o As for the previous case, 8 Blocks per Cluster seem to be the optimum value.

- o The use of B frame degrades the overhead by approximately 0.5%. It is due to the additional backward reference.

- A bitrate of over 128 kbps still have a good overhead. But the 2.4% for 64kbps is still acceptable.

281.5. Example 5 - medium bitrate video

281.6. Example 6 - high bitrate video

281.7. Example 7 - low bitrate video + low bitrate audio

281.8. Example 8 - medium bitrate video + medium bitrate audio

281.9. Example 9 - high bitrate video + high bitrate audio

282. References

282.1. URIs

[1] http://mukoli.free.fr/mcf/mcf.html

[2] https://github.com/Matroska-Org/ebml-specification/blob/master/specification.markdown

[3] https://datatracker.ietf.org/wg/cellar/charter/

[4] https://matroska.org/files/matroska.pdf

[5] {{site.baseurl}}/diagram.html

[6] https://github.com/Matroska-Org/ebml-specification/blob/master/specification.markdown

[7] https://github.com/Matroska-Org/foundation-source/blob/master/spectool/specdata.xml

[8] https://tools.ietf.org/html/rfc2119

[9] https://github.com/Matroska-Org/ebml-specification/blob/master/specification.markdown

[10] https://github.com/Matroska-Org/ebml-specification/blob/master/specification.markdown#ebml-element-types

[11] https://github.com/Matroska-Org/ebml-specification/blob/master/specification.markdown#ebml-schema

[12] https://github.com/Matroska-Org/ebml-specification/blob/master/specification.markdown#structure

[13] http://www.loc.gov/standards/iso639-2/php/English_list.php

[14] http://www.iana.org/cctld/cctld-whois.htm

[15] {{site.baseurl}}/index.html#block

[16] http://www.w3.org/TR/1999/REC-xml-names-19990114/#ns-decl

[17] https://www.w3.org/TR/xmlschema-0/#ref6

[18] https://www.w3.org/TR/xmlschema-0/#ref6

[19] http://www.webmproject.org/docs/container/

[20] http://www.matroska.org/downloads/test_w1.html

[21] {{site.baseurl}}/index.html#Cluster

[22] {{site.baseurl}}/index.html#Timecode

[23] {{site.baseurl}}/notes.html#TimecodeScale

[24] {{site.baseurl}}/index.html#Cluster

[25] {{site.baseurl}}/index.html#Timecode

[26] {{site.baseurl}}/notes.html#Encryption

[27] {{site.baseurl}}/notes.html#unknown-elements

[28] {{site.baseurl}}/index.html#Cluster

[29] {{site.baseurl}}/index.html#Timecode

[30] {{site.baseurl}}/index.html#TimecodeScale

[31] {{site.baseurl}}/index.html#Cluster

[32] {{site.baseurl}}/index.html#Timecode

[33] {{site.baseurl}}/index.html#Cluster

[34] {{site.baseurl}}/index.html#Timecode

[35] {{site.baseurl}}/index.html#TrackTimeCodeScale

[36] {{site.baseurl}}/notes.html#raw-timecode

[37] {{site.baseurl}}/notes.html#timecode-types

[38] {{site.baseurl}}/notes.html#matroska-version-indicators-doctypeversion-and-doctypereadversion

[39] {{site.baseurl}}/index.html#StereoMode

[40] {{site.baseurl}}/index.html#TrackOperation

[41] {{site.baseurl}}/index.html#TrackCombinePlanes

[42] {{site.baseurl}}/notes.html#track-operation

[43] {{site.baseurl}}/index.html#TrackOperation

[44] {{site.baseurl}}/index.html#Void

[45] {{site.baseurl}}/order_guidelines.html#tags-end

[46] http://haali.su/mkv/codecs.pdf

[47] {{site.baseurl}}/index.html

[48] {{site.baseurl}}/index.html#CodecID

[49] http://www.corecodec.com/modules.php?op=modload&name=PNphpBB2&file=viewtopic&t=227

[50] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/bitmaps_1rw2.asp

[51] http://forum.doom9.org/showthread.php?s=&threadid=55773&perpage=20&pagenumber=2#post331855

[52] https://github.com/mbunkus/mkvtoolnix/blob/master/lib/librmff/librmff.h

[53] http://developer.apple.com/documentation/quicktime/QTFF/qtff.pdf

[54] {{site.baseurl}}/index.html#lacing

[55] http://www.theora.org/doc/Theora_I_spec.pdf

[56] http://wiki.multimedia.cx/index.php?title=Apple_ProRes#Frame_layout

[57] http://alac.macosforge.org/trac/browser/trunk/ALACMagicCookieDescription.txt

[58] {{site.baseurl}}/index.html#lacing

[59] http://www.xiph.org/ogg/vorbis/doc/vorbis-spec-ref.html

[60] http://www.xiph.org/ogg/vorbis/doc/v-comment.html

[61] http://www.xiph.org/ogg/vorbis/doc/vorbis-spec-ref.html

[62] http://flac.sourceforge.net/

[63] https://github.com/mbunkus/mkvtoolnix/blob/master/lib/librmff/librmff.h

[64] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/multimed/mmstr_625u.asp

[65] http://developer.apple.com/documentation/quicktime/QTFF/qtff.pdf

[66] http://tausoft.org/

[67] http://tausoft.org/wiki/True_Audio_Codec_Format

[68] http://www.wavpack.com/

[69] wavpack.html

[70] {{site.baseurl}}/subtitles.html

[71] ../subtitles/ssa.html

[72] {{site.baseurl}}/subtitles.html

[73] {{site.baseurl}}/subtitles.html

[74] {{site.baseurl}}/subtitles.html

[75] http://wiki.xiph.org/index.php/OggKate#Matroska_mapping

[76] http://dvd.sourceforge.net/dvdinfo/pci_pkt.html

[77] http://dvd.sourceforge.net/dvdinfo/pci_pkt.html

[78] {{site.baseurl}}/index.html#ChapProcessCodecID

[79] {{site.baseurl}}/index.html#ChapProcessCodecID

[80] {{site.baseurl}}/index.html#ChapterTranslateID

[81] http://dvd.sourceforge.net/dvdinfo/uops.html

[82] http://www.dvd-replica.com/DVD/

[83] http://dvd.sourceforge.net/dvdinfo/

[84] http://wiki.multimedia.cx/index.php?title=SubStation_Alpha

[85] http://moodub.free.fr/video/ass-specs.doc

[87] mailto:kotus@eswat.demon.co.uk

[89] mailto:kotus@eswat.demon.co.uk

[90] https://www.w3.org/

[91] https://w3c.github.io/webvtt/

[92] http://www.webmproject.org/docs/container/

[93] http://www.matroska.org/files/tags/matroskatags.dtd

[94] http://www.bunkus.org/videotools/mkvtoolnix/doc/mkvmerge.html#mkvmerge.tags

[95] othertagsystems/comparetable.html

[96] https://sourceforge.net/projects/matroska/files/test_files/cover_art.mkv/download

[97] {{site.baseurl}}/index.html

[98] {{site.baseurl}}/chapters/menu.html#playback-features

Authors' Addresses

Steve Lhomme

Moritz Bunkus

Dave Rice

FF Video Codec 1
draft-niedermayer-cellar-ffv1-00

Abstract

This document defines FFV1, a lossless intra-frame video encoding format. FFV1 is designed to efficiently compress video data in a variety of pixel formats. Compared to uncompressed video, FFV1 offers storage compression, frame fixity, and self-description, which makes FFV1 useful as a preservation or intermediate video format.

Table of Contents

1. Introduction 3
2. Notation and Conventions 3
  2.1. Definitions 3
3. Conventions 3
  3.1. Arithmetic operators 4
  3.2. Assignment operators 4
  3.3. Comparison operators 4
  3.4. Mathematical functions 5
  3.5. Order of operation precedence 5
  3.6. Range 5
  3.7. Bitstream functions 5
4. General Description 6
  4.1. Border 6
  4.2. Median predictor 6
  4.3. Context 7
  4.4. Quantization 7
  4.5. Colorspace 7
    4.5.1. JPEG2000-RCT 7
  4.6. Coding of the sample difference 8
    4.6.1. Range coding mode 8
    4.6.2. Huffman coding mode 11
5. Bitstream 13
  5.1. Configuration Record 14
    5.1.1. In AVI File Format 15
    5.1.2. In ISO/IEC 14496-12 (MP4 File Format) 15
    5.1.3. In NUT File Format 15
  5.2. Frame 15
  5.3. Slice 16
  5.4. Slice Header 16
  5.5. Slice Content 17
  5.6. Line 18
  5.7. Slice Footer 19
  5.8. Parameters 19
  5.9. Quantization Tables 24
    5.9.1. Restrictions 25
6. Appendixes 25
  6.1. Decoder implementation suggestions 26
    6.1.1. Multi-threading support and independence of slices . 26
7. Changelog 26
8. ToDo 26
9. Bibliography 27
  9.1. References 27
10. Copyright 27
  11.1. URIs 27
Author’s Address 28

1. Introduction

The FFV1 video codec is a simple and efficient lossless intra-frame only codec.

The latest version of this document is available at <https://raw.github.com/FFmpeg/FFV1/master/ffv1.md>

This document assumes familiarity with mathematical and coding concepts such as Range coding and YCbCr colorspaces.

2. Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

2.1. Definitions

ESC An ESCape symbol to indicate that the symbol to be stored is too large for normal storage and that an alternate storage method.

MSB Most Significant Bit, the bit that can cause the largest change in magnitude of the symbol.

RCT Reversible Color Transform, a near linear, exactly reversible integer transform that converts between RGB and YCbCr representations of a sample.

VLC Variable Length Code.

RGB A reference to the method of storing the value of a sample by using three numeric values that represent Red, Green, and Blue.

YCbCr A reference to the method of storing the value of a sample by using three numeric values that represent the luminance of the sample (Y) and the chrominance of the sample (Cb and Cr).

TBA To Be Announced. Used in reference to the development of future iterations of the FFV1 specification.

3. Conventions

Note: the operators and the order of precedence are the same as used in the C programming language Section 9.1.

3.1. Arithmetic operators

"a + b" means a plus b.

"a - b" means a minus b.

"-a" means negation of a.

"a * b" means a multiplied by b.

"a / b" means a divided by b.

"a & b" means bit-wise "and" of a and b.

"a | b" means bit-wise "or" of a and b.

"a >> b" means arithmetic right shift of two’s complement integer representation of a by b binary digits.

"a << b" means arithmetic left shift of two’s complement integer representation of a by b binary digits.

3.2. Assignment operators

"a = b" means a is assigned b.

"a++" is equivalent to a = a + 1.

"a-" is equivalent to a = a - 1.

"a += b" is equivalent to a = a + b.

"a -= b" is equivalent to a = a - b.

3.3. Comparison operators

"a > b" means a is greater than b.

"a >= b" means a is greater than or equal to b.

"a < b" means a is less than b.

"a <= b" means a is less than or equal b.

"a == b" means a is equal to b.

"a != b" means a is not equalto b.

"a && b" means boolean logical "and" of a and b.

"a || b" means boolean logical "or" of a and b.

"!a" means boolean logical "not".

"a ? b : c" if a is true, then b, otherwise c.

3.4. Mathematical functions

$\lfloor a \rfloor$ the largest integer less than or equal to a

$\lceil a \rceil$ the smallest integer greater than or equal to a

3.5. Order of operation precedence

When order of precedence is not indicated explicitly by use of parentheses, operations are evaluated in the following order (from top to bottom, operations of same precedence being evaluated from left to right). This order of operations is based on the order of operations used in Standard C.

```
a++, a-
!a, -a
a * b, a / b, a % b
a + b, a - b
a << b, a >> b
a < b, a <= b, a > b, a >= b
a == b, a != b
a & b
a | b
a && b
a || b
a ? b : c
a = b, a += b, a -= b
```

3.6. Range

"a...b" means any value starting from a to b, inclusive.

3.7. Bitstream functions

"remaining_bits_in_bitstream( )" means the count of remaining bits after the current position in the bitstream. It is computed from the NumBytes value multiplied by 8 minus the count of bits already read by the bitstream parser.

"byte_aligned( )" means "remaining_bits_in_bitstream( )" is a multiple of 8.

\pagebreak

4. General Description

Each frame is split in 1 to 4 planes (Y, Cb, Cr, Alpha). In the case of the normal YCbCr colorspace the Y plane is coded first followed by the Cb and Cr planes, if an Alpha/transparency plane exists, it is coded last. In the case of the JPEG2000-RCT colorspace the lines are interleaved to improve caching efficiency since it is most likely that the RCT will immediately be converted to RGB during decoding; the interleaved coding order is also Y, Cb, Cr, Alpha.

Samples within a plane are coded in raster scan order (left->right, top->bottom). Each sample is predicted by the median predictor from samples in the same plane and the difference is stored see Section 4.6.

4.1. Border

For the purpose of the predictior and context, samples above the coded slice are assumed to be 0; samples to the right of the coded slice are identical to the closest left sample; samples to the left of the coded slice are identical to the top right sample (if there is one), otherwise 0.

```
+---+---+---+---+---+---+---+---+
| 0 | 0 |   | 0 | 0 | 0 |   | 0 |
| 0 | 0 |   | 0 | 0 | 0 |   | 0 |
|   |   |   |   |   |   |   |   |
| 0 | 0 |   | a | b | c |   | c |
| 0 | a |   | d |   | e |   | e |
| 0 | d |   | f | g | h |   | h |
+---+---+---+---+---+---+---+---+
```

4.2. Median predictor

median(left, top, left + top - diag)

left, top, diag are the left, top and left-top samples

Note, this is also used in Section 9.1.

4.3. Context

```
+---+----+---+----+
|   |    | T |    |
|   | tl | t | tr |
| L | l  | X |    |
+---+----+---+----+
```

The quantized sample differences L-l, l-tl, tl-t, t-T, t-tr are used as context:

$context=Q_{0}[l-tl]+\left|Q_{0}\right|(Q_{1}[tl-t]+\left|Q_{1}\right|(Q_{2}[t-tr]+\left|Q_{2}\right|(Q_{3}[L-l]+\left|Q_{3}\right|Q_{4}[T-t])))$

If the context is smaller than 0 then -context is used and the difference between the sample and its predicted value is encoded with a flipped sign.

4.4. Quantization

There are 5 quantization tables for the 5 sample differences, both the number of quantization steps and their distribution are stored in the bitstream. Each quantization table has exactly 256 entries, and the 8 least significant bits of the sample difference are used as index:

$Q_{i}[a-b]=Table_{i}[(a-b)\&255]$

4.5. Colorspace

4.5.1. JPEG2000-RCT

$Cb=b-g$

$Cr=r-g$

$Y=g+(Cb+Cr)>>2$

$g=Y-(Cb+Cr)>>2$

$r=Cr+g$

$b=Cb+g$

Section 9.1

4.6. Coding of the sample difference

Instead of coding the n+1 bits of the sample difference with Huffman-, or Range coding (or n+2 bits, in the case of RCT), only the n (or n+1) least significant bits are used, since this is sufficient to recover the original sample. In the equation below, the term "bits" represents bits_per_raw_sample+1 for RCT or bits_per_raw_sample otherwise:

$coder_input=\left[\left(sample_difference+2^{bits-1}\right)&\left(2^{bits}-1\right)\right]-2^{bits-1}$

4.6.1. Range coding mode

Early experimental versions of FFV1 used the CABAC Arithmetic coder from Section 9.1 but due to the uncertain patent/royality situation, as well as its slightly worse performance, CABAC was replaced by a Range coder based on an algorithm defined by _G. Nigel N. Martin_ in 1979 Section 9.1.

4.6.1.1. Range binary values

To encode binary digits efficiently a Range coder is used. C_{i} is the i-th Context. B_{i} is the i-th byte of the bytestream. b_{i} is the i-th Range coded binary value, $S_{0,i}$ is the i-th initial state, which is 128. The length of the bytestream encoding n binary symbols is j_{n} bytes.

$r_{i}=\left\lfloor \frac{R_{i}S_{i,C_{i}}}{2^{8}}\right\rfloor$

$\begin{array}{ccccccccc} S_{i+1,C_{i}}=zero_state_{S_{i,C_{i}}} & \wedge & l_{i}=L_{i} & \wedge & t_{i}=R_{i}-r_{i} & \Longleftarrow & b_{i}=0 & \Longleftrightarrow & L_{i}

=one_state_{S_{i,C_{i}}} & \wedge & l_{i}=L_{i}-R_{i}+r_{i} & \wedge & t_{i}=r_{i} & \Longleftarrow & b_{i}=1 & \Longleftrightarrow & L_{i}\geq R_{i}-r_{i} \end{array}$

$\begin{array}{ccc} S_{i+1,k}=S_{i,k} & \Longleftarrow & C_{i}\neq k \end{array}$

$\begin{array}{ccccccc} R_{i+1}=2^{8}t_{i} & \wedge & L_{i+1}=2^{8}l_{i}+B_{j_{i}} & \wedge & j_{i+1}=j_{i}+1 & \Longleftarrow & t_{i}<2^{8}
R_{i+1}=t_{i} & \wedge & L_{i+1}=l_{i} & \wedge & j_{i+1}=j_{i} & \Longleftarrow & t_{i}\geq2^{8} \end{array}$

$R_{0}=65280$

$L_{0}=2^{8}B_{0}+B_{1}$

$j_{0}=2$

4.6.1.2. Range non binary values

To encode scalar integers it would be possible to encode each bit separately and use the past bits as context. However that would mean 255 contexts per 8-bit symbol which is not only a waste of memory but also requires more past data to reach a reasonably good estimate of the probabilities. Alternatively assuming a Laplacian distribution and only dealing with its variance and mean (as in Huffman coding) would also be possible, however, for maximum flexibility and simplicity, the chosen method uses a single symbol to encode if a number is 0 and if not encodes the number using its exponent, mantissa and sign. The exact contexts used are best described by the following code, followed by some comments.

```
void put_symbol(RangeCoder *c, uint8_t *state, int v, int is_signed) {
    int i;
    put_rac(c, state+0, !v);
    if (v) {
        int a= ABS(v);
        int e= log2(a);

        for (i=0; i<e; i++)
            put_rac(c, state+1+MIN(i,9), 1);  //1..10

        put_rac(c, state+1+MIN(i,9), 0);
        for (i=e-1; i>=0; i--)
            put_rac(c, state+22+MIN(i,9), (a>>i)&1); //22..31

        if (is_signed)
            put_rac(c, state+11 + MIN(e, 10), v < 0); //11..21
    }
}
```

4.6.1.3. Initial values for the context model

At keyframes all Range coder state variables are set to their initial state.

4.6.1.4. State transition table

$one\_state_{i}=default\_state\_transition_{i}+state\_transition\_delta_{i}$

$zero\_state_{i}=256-one\_state_{256-i}$

4.6.1.5. default_state_transition

```
  0,  0,  0,  0,  0,  0,  0,  0, 20, 21, 22, 23, 24, 25, 26, 27,

 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 37, 38, 39, 40, 41, 42,

 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 56, 57,

 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,

 74, 75, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,

 89, 90, 91, 92, 93, 94, 94, 95, 96, 97, 98, 99,100,101,102,103,

104,105,106,107,108,109,110,111,112,113,114,114,115,116,117,118,

119,120,121,122,123,124,125,126,127,128,129,130,131,132,133,133,

134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,

150,151,152,152,153,154,155,156,157,158,159,160,161,162,163,164,

165,166,167,168,169,170,171,171,172,173,174,175,176,177,178,179,

180,181,182,183,184,185,186,187,188,189,190,190,191,192,194,194,

195,196,197,198,199,200,201,202,202,204,205,206,207,208,209,209,

210,211,212,213,215,215,216,217,218,219,220,220,222,223,224,225,

226,227,227,229,229,230,231,232,234,234,235,236,237,238,239,240,

241,242,243,244,245,246,247,248,248,  0,  0,  0,  0,  0,  0,  0,
```

4.6.1.6. alternative state transition table

The alternative state transition table has been build using iterative minimization of frame sizes and generally performs better than the default. To use it, the coder_type has to be set to 2 and the difference to the default has to be stored in the parameters. The reference implemenation of FFV1 in FFmpeg uses this table by default at the time of this writing when Range coding is used.

```
  0, 10, 10, 10, 10, 16, 16, 16, 28, 16, 16, 29, 42, 49, 20, 49,

 59, 25, 26, 26, 27, 31, 33, 33, 33, 34, 34, 37, 67, 38, 39, 39,

 40, 40, 41, 79, 43, 44, 45, 45, 48, 48, 64, 50, 51, 52, 88, 52,

 53, 74, 55, 57, 58, 58, 74, 60,101, 61, 62, 84, 66, 66, 68, 69,

 87, 82, 71, 97, 73, 73, 82, 75,111, 77, 94, 78, 87, 81, 83, 97,

 85, 83, 94, 86, 99, 89, 90, 99,111, 92, 93,134, 95, 98,105, 98,

105,110,102,108,102,118,103,106,106,113,109,112,114,112,116,125,

115,116,117,117,126,119,125,121,121,123,145,124,126,131,127,129,

165,130,132,138,133,135,145,136,137,139,146,141,143,142,144,148,

147,155,151,149,151,150,152,157,153,154,156,168,158,162,161,160,

172,163,169,164,166,184,167,170,177,174,171,173,182,176,180,178,

175,189,179,181,186,183,192,185,200,187,191,188,190,197,193,196,

197,194,195,196,198,202,199,201,210,203,207,204,205,206,208,214,

209,211,221,212,213,215,224,216,217,218,219,220,222,228,223,225,

226,224,227,229,240,230,231,232,233,234,235,236,238,239,237,242,

241,243,242,244,245,246,247,248,249,250,251,252,252,253,254,255,
```

4.6.2. Huffman coding mode

This coding mode uses golomb rice codes. The VLC code is split into 2 parts, the prefix stores the most significant bits, the suffix stores the k least significant bits or stores the whole number in the ESC case. The end of the bitstream (of the frame) is filled with 0-bits so that the bitstream contains a multiple of 8 bits.

4.6.2.1. Prefix

bits	value
1	0
01	1
...	...
0000 0000 0001	11
0000 0000 0000	ESC

4.6.2.2. Suffix

non ESC	the k least significant bits MSB first
ESC	the value - 11, in MSB first order, ESC may only be used if the value cannot be coded as non ESC

4.6.2.3. Examples

k	bits	value
0	"1"	0
0	"001"	2
2	"1 00"	0
2	"1 10"	2
2	"01 01"	5
any	"000000000000 10000000"	139

4.6.2.4. Run mode

Run mode is entered when the context is 0, and left as soon as a non-0 difference is found, the level is identical to the predicted one, the run and the first different level is coded.

4.6.2.5. Run length coding

The run value is encoded in 2 parts, the prefix part stores the more significant part of the run as well as adjusting the run_index which determines the number of bits in the less significant part of the run. The 2nd part of the value stores the less significant part of the run as it is. The run_index is reset for each plane and slice to 0.

```
log2_run[41]={
 0, 0, 0, 0, 1, 1, 1, 1,
 2, 2, 2, 2, 3, 3, 3, 3,
 4, 4, 5, 5, 6, 6, 7, 7,
 8, 9,10,11,12,13,14,15,
16,17,18,19,20,21,22,23,
24,
};

if (run_count == 0 && run_mode == 1) {
    if (get_bits1()) {
        run_count = 1 << log2_run[run_index];
        if (x + run_count <= w)
            run_index++;
    } else {
        if (log2_run[run_index])
            run_count = get_bits(log2_run[run_index]);
        else
            run_count = 0;
        if (run_index)
            run_index--;
        run_mode = 2;
    }
}
```

The log2_run function is also used within Section 9.1.

4.6.2.6. Level coding

Level coding is identical to the normal difference coding with the exception that the 0 value is removed as it cannot occur:

```
if(diff>0) diff--;
encode(diff);
```

Note, this is different from JPEG-LS, which doesn’t use prediction in run mode and uses a different encoding and context model for the last difference On a small set of test samples the use of prediction slightly improved the compression rate.

5. Bitstream

Symbol	Defintion
u(n)	unsigned big endian integer using n bits
sg	Golomb Rice coded signed scalar symbol coded with the method described in Section 4.6.2
br	Range coded boolean (1-bit) symbol with the method described in Section 4.6.1.1
ur	Range coded unsigned scalar symbol coded with the method described in Section 4.6.1.2
sr	Range coded signed scalar symbol coded with the method described in Section 4.6.1.2

The same context which is initialized to 128 is used for all fields in the header.

The following MUST be provided by external means during initialization of the decoder:

"frame_pixel_width" is defined as frame width in pixels.

"frame_pixel_height" is defined as frame height in pixels.

Default values at the decoder initialization phase:

"ConfigurationRecordIsPresent" is set to 0.

5.1. Configuration Record

In the case of a bitstream with version >= 3, a configuration record is stored in the underlying container, at the track header level. It contains the parameters used for all frames. The size of the configuration record, NumBytes, is supplied by the underlying container.

"c ConfigurationRecord(NumBytes) { ConfigurationRecordIsPresent = 1 Parameters( ) while(remaining_bits_in_bitstream( ) > 32) reserved_for_future_use // u(1) configuration_record_crc_parity // u(32)"‘

"reserved_for_future_use" has semantics that are reserved for future use. Encoders conforming to this version of this specification SHALL NOT write this value. Decoders conforming to this version of this specification SHALL ignore its value.

"configuration_record_crc_parity" 32 bits that are choosen so that the configuration record as a whole has a crc remainder of 0. This is equivalent to storing the crc remainder in the 32-bit parity. The CRC generator polynom used is the standard IEEE CRC polynom (0x104C11DB7) with initial value 0.

This configuration record can be placed in any file format supporting configuration records, fitting as much as possible with how the file format uses to store configuration records. The configuration record storage place and NumBytes are currently defined and supported by this version of this specification for the following container formats:

5.1.1. In AVI File Format

The Configuration Record extends the stream format chunk ("AVI ", "hdlr", "strl", "strf") with the ConfigurationRecord bistream. See Section 9.1 for more information about chunks.

"NumBytes" is defined as the size, in bytes, of the strf chunk indicated in the chunk header minus the size of the stream format structure.

5.1.2. In ISO/IEC 14496-12 (MP4 File Format)

The Configuration Record extends the sample description box ("moov", "trak", "mdia", "minf", "stbl", "stsd") with a "glbl" box which contains the ConfigurationRecord bitstream. See Section 9.1 for more information about boxes.

"NumBytes" is defined as the size, in bytes, of the "glbl" box indicated in the box header minus the size of the box header.

5.1.3. In NUT File Format

The codec_specific_data element (in "stream_header" packet) contains the ConfigurationRecord bitstream. See Section 9.1 for more information about elements.

"NumBytes" is defined as the size, in bytes, of the codec_specific_data element as indicated in the "length" field of codec_specific_data

5.2. Frame

A frame consists of the keyframe field, parameters (if version <=1), and a sequence of independent slices.

| | |--|---:| |Frame ( ) { |type| | keyframe | br| | if(keyframe &&

!ConfigurationRecordIsPresent)| | | Parameters(
) | | | while (remaining_bits_in_bitstream()
) | | | Slice( ) | | |} | |

5.3. Slice

| | |---|:--
----| |Slice( ) { | type | | if(version >= 3
) | | | SliceHeader( ) | | | SliceContent( ) | | | if (
coder_type == 0) | | | while (!byte_aligned()) | | |
padding | u(1) | | if(version >= 3) | | | SliceFooter(
) | | |} | |

"padding" specifies a bit without any significance and used only for byte alignment. MUST be 0.

5.4. Slice Header

SliceHeader( ) {	type
slice_x	ur
slice_y	ur
slice_width - 1	ur
slice_height - 1	ur
for(i = 0; i < quant_table_index_count; i++)	
quant_table_index [i]	ur
picture_structure	ur
sar_num	ur
sar_den	ur
if(version >= 4) {	
reset_contexts	br
slice_coding_mode	ur
}	
}	

"slice_x" indicates the x position on the slice raster formed by num_h_slices. Inferred to be 0 if not present.

"slice_y" indicates the y position on the slice raster formed by num_v_slices. Inferred to be 0 if not present.

"slice_width" indicates the width on the slice raster formed by num_h_slices. Inferred to be 1 if not present.

"slice_height" indicates the height on the slice raster formed by num_v_slices. Inferred to be 1 if not present.

"quant_table_index_count" is defined as 1 + ((chroma_planes || version <= 3) ? 1 : 0) + (alpha_plane ? 1 : 0).

"quant_table_index" indicates the index to select the quantization table set and the initial states for the slice. Inferred to be 0 if not present.

"picture_structure" specifies the picture structure. Inferred to be 0 if not present.

value	picure structure used
0	unknown
1	top field first
2	bottom field first
3	progressive
Other	reserved for future use

"sar_num" specifies the sample aspect ratio numerator. Inferred to be 0 if not present. MUST be 0 if sample aspect ratio is unknown.

"sar_den" specifies the sample aspect ratio numerator. Inferred to be 0 if not present. MUST be 0 if sample aspect ratio is unknown.

"reset_contexts" indicates if slice contexts must be reset. Inferred to be 0 if not present.

"slice_coding_mode" indicates the slice coding mode. Inferred to be 0 if not present.

value	slice coding mode
0	normal Range Coding or VLC
1	raw PCM
Other	reserved for future use

5.5. Slice Content

```
| | |----------------------------------------------------------|:
------| |SliceContent( ) { | type | |    if( colorspace_type == 0)
{ | | |        for( p = 0; p < primary_color_count; p++ )
{ | | |            for( y = 0; y < plane_pixel_height[ p ]; y++
) | | |                Line( p, y ) | | |    } else if(
colorspace_type == 1 ) { | | |        for( y = 0; y <
```

slice_pixel_height; y++) | | | for(p = 0; p <
primary_color_count; p++) { | | | Line(p, y
) | | | } | | |} | |

"primary_color_count" is defined as 1 + (chroma_planes ? 2 : 0) + (alpha_plane ? 1 : 0).

"plane_pixel_height[p]" is the height in pixels of plane p of the slice.
plane_pixel_height[0] and plane_pixel_height[1 + (chroma_planes ? 2 : 0)] value is slice_pixel_height
if chroma_planes is set to 1, plane_pixel_height[1] and plane_pixel_height[2] value is $\lceil slice_pixel_height / v_chroma_subsample \rceil$

"slice_pixel_height" is the height in pixels of the slice.
Its value is $\lfloor (slice_y + slice_height) * slice_pixel_height / num_v_slices \rfloor - slice_pixel_y$

"slice_pixel_y" is the slice vertical position in pixels.
Its value is $\lfloor slice_y * frame_pixel_height / num_v_slices \rfloor$

5.6. Line

| | |--|:
------| |Line(p, y) { | type | | if(colorspace_type == 0)
{ | | | for(x = 0; x < plane_pixel_width[p]; x++
) | | | Pixel(p, y, x) | | | } else if(
colorspace_type == 1) { | | | for(x = 0; x <
slice_pixel_width; x++) | | | Pixel(p, y, x
) | | | } | | |} | |

"plane_pixel_width[p]" is the width in pixels of plane p of the slice.
plane_pixel_width[0] and plane_pixel_width[1 + (chroma_planes ? 2 : 0)] value is slice_pixel_width
if chroma_planes is set to 1, plane_pixel_width[1] and plane_pixel_width[2] value is $\lceil slice_pixel_width / v_chroma_subsample \rceil$

"slice_pixel_width" is the width in pixels of the slice.
Its value is $\lfloor (slice_x + slice_width) * slice_pixel_width / num_h_slices \rfloor - slice_pixel_x$

"slice_pixel_x" is the slice horizontal position in pixels.
Its value is $\lfloor slice_x * frame_pixel_width / num_h_slices \rfloor$

5.7. Slice Footer

Note: slice footer is always byte aligned.

| | |---|:------| |SliceFooter( ) { | type | | slice_size | u(24) | | if(ec) { | | | | error_status | u(8) | | slice_crc_parity | u(32) | | } | | |} | |

"slice_size" indicates the size of the slice in bytes. Note: this allows finding the start of slices before previous slices have been fully decoded. And allows this way parallel decoding as well as error resilience.

"error_status" specifies the error status.

value	error status
0	no error
1	slice contains a correctable error
2	slice contains a uncorrectable error
Other	reserved for future use

"slice_crc_parity" 32 bits that are choosen so that the slice as a whole has a crc remainder of 0. This is equivalent to storing the crc remainder in the 32-bit parity. The CRC generator polynom used is the standard IEEE CRC polynom (0x104C11DB7) with initial value 0.

5.8. Parameters

```
+------------------------------------------+------+
|              Parameters( ) {             | type |
|                 version                  |   ur |
|             if( version >= 3 )           |      |
|               micro_version              |   ur |
|                coder_type                |   ur |
|            if( coder_type > 1 )          |      |
|        for( i = 1; i < 256; i++ )        |      |
|       state_transition_delta[ i ]        |   sr |
|             colorspace_type              |   ur |
|             if( version >= 1 )           |      |
|           bits_per_raw_sample            |   ur |
|              chroma_planes               |   br |
|        log2( h_chroma_subsample )        |   ur |
|        log2( v_chroma_subsample )        |   ur |
|               alpha_plane                |   br |
|            if( version >= 3 ) {          |      |
|             num_h_slices - 1             |   ur |
|             num_v_slices - 1             |   ur |
|            quant_table_count             |   ur |
|                    }                     |      |
|  for( i = 0; i < quant_table_count; i++ ) |      |
|          QuantizationTable( i )          |      |
|            if( version >= 3 ) {          |      |
| for( i = 0; i < quant_table_count; i++ ) { |      |
|               states_coded               |   br |
|             if( states_coded )           |      |
| for( j = 0; j < context_count[ i ]; j++ ) |      |
|    for( k = 0; k < CONTEXT_SIZE; k++ )   |      |
|     initial_state_delta[ i ][ j ][ k ]   |   sr |
|                    }                     |      |
|                    ec                    |   ur |
|                  intra                   |   ur |
|                    }                     |      |
|                    }                     |      |
+------------------------------------------+------+
```

"version" specifies the version of the bitstream. Each version is incompatible with others versions: decoders SHOULD reject a file due to unknown version. Decoders SHOULD reject a file with version =< 1 && ConfigurationRecordIsPresent == 1. Decoders SHOULD reject a file with version >= 3 && ConfigurationRecordIsPresent == 0.

value	version
0	FFV1 version 0
1	FFV1 version 1
2	reserved*
3	FFV1 version 3
Other	reserved for future use

* Version 2 was never enabled in the encoder thus version 2 files SHOULD NOT exist, and this document does not describe them to keep the text simpler.

"micro_version" specifies the micro-version of the bitstream. After a version is considered stable (a micro-version value is assigned to be the first stable variant of a specific version), each new micro-version after this first stable variant is compatible with the previous micro-version: decoders SHOULD NOT reject a file due to an unknown micro-version equal or above the micro-version considered as stable.

Meaning of micro_version for version 3:

value	micro_version
0...3	reserved*
4	first stable variant
Other	reserved for future use

* were development versions which may be incompatible with the stable variants.

Meaning of micro_version for version 4 (note: at the time of writting of this specification, version 4 is not considered stable so the first stable version value is to be annonced in the future):

value	micro_version
0...TBA	reserved*
TBA	first stable variant
Other	reserved for future use

* were development versions which may be incompatible with the stable variants.

"coder_type" specifies the coder used

value	coder used
0	Golomb Rice
1	Range Coder with default state transition table
2	Range Coder with custom state transition table
Other	reserved for future use

"state_transition_delta" specifies the Range coder custom state transition table. If state_transition_delta is not present in the bitstream, all Range coder custom state transition table elements are assumed to be 0.

"colorspace_type" specifies the color space.

value	color space used
0	YCbCr
1	JPEG 2000 RCT
Other	reserved for future use

"chroma_planes" indicates if chroma (color) planes are present.

value	color space used
0	chroma planes are not present
1	chroma planes are present

"bits_per_raw_sample" indicates the number of bits for each luma and chroma sample. Inferred to be 8 if not present.

value	bits for each luma and chroma sample
0	reserved*
Other	the actual bits for each luma and chroma sample

* Encoders MUST NOT store bits_per_raw_sample = 0 Decoders SHOULD accept and interpret bits_per_raw_sample = 0 as 8.

"h_chroma_subsample" indicates the subsample factor between luma and chroma width ($chroma_width=2^{-log2_h_chroma_subsample}luma_width$)

"v_chroma_subsample" indicates the subsample factor between luma and chroma height ($chroma_height=2^{-log2_v_chroma_subsample}luma_height$)

alpha_plane
indicates if a transparency plane is present.

value	color space used
0	transparency plane is not present
1	transparency plane is present

"num_h_slices" indicates the number of horizontal elements of the slice raster. Inferred to be 1 if not present.

"num_v_slices" indicates the number of vertical elements of the slice raster. Inferred to be 1 if not present.

"quant_table_count" indicates the number of quantization table sets. Inferred to be 1 if not present.

"states_coded" indicates if the respective quantization table set has the initial states coded. Inferred to be 0 if not present.

value	initial states
0	initial states are not present and are assumed to be all 128
1	initial states are present

"initial_state_delta" [i][j][k] indicates the initial Range coder state, it is encoded using k as context index and pred = j ? initial_states[i][j - 1][k] : 128 initial_state[i][j][k] = (pred + initial_state_delta[i][j][k]) & 255

"ec" indicates the error detection/correction type.

value	error detection/correction type
0	32bit CRC on the global header
1	32bit CRC per slice and the global header
Other	reserved for future use

"intra" indicates the relationship between frames. Inferred to be 0 if not present.

value	relationship
0	frames are independent or dependent (key and non key frames)
1	frames are independent (key frames only)
Other	reserved for future use

5.9. Quantization Tables

The quantization tables are stored by storing the number of equal entries -1 of the first half of the table using the method described in Section 4.6.1.2. The second half doesn’t need to be stored as it is identical to the first with flipped sign.

example:

Table: 0 0 1 1 1 1 2 2-2-2-2-1-1-1-1 0

Stored values: 1, 3, 1

```
QuantizationTable( i ) {
    scale = 1
    for( j = 0; j < MAX_CONTEXT_INPUTS; j++ ) {
        QuantizationTablePerContext( i, j, scale )
        scale *= 2 * len_count[ i ][ j ] - 1
    }
    context_count[ i ] = ( scale + 1 ) / 2
```

MAX_CONTEXT_INPUTS is 5.

```
+------------------------------------------------------------+------+
|         QuantizationTablePerContext(i, j, scale) {         | type |
|                           v = 0                            |      |
|                  for( k = 0; k < 128; ) {                  |      |
|                          len - 1                           |  sr  |
|                for( a = 0; a < len; a++ ) {                |      |
|          quant_tables[ i ][ j ][ k ] = scale* v            |      |
|                            k++                             |      |
|                             }                              |      |
|                            v++                             |      |
|                             }                              |      |
|                for( k = 1; k < 128; k++ ) {                |      |
| quant_tables[ i ][ j ][ 256 - k ] = -quant_tables[ i ][ j  |      |
|                           ][ k ]                           |      |
|                             }                              |      |
|  quant_tables[ i ][ j ][ 128 ] = -quant_tables[ i ][ j ][  |      |
|                           127 ]                            |      |
|                  len_count[ i ][ j ] = v                   |      |
|                             }                              |      |
+------------------------------------------------------------+------+
```

"quant_tables" indicates the quantification table values.

"context_count" indicates the count of contexts.

5.9.1. Restrictions

To ensure that fast multithreaded decoding is possible, starting version 3 and if frame_pixel_width * frame_pixel_height is more than 101376, slice_width * slice_height MUST be less or equal to num_h_slices * num_v_slices / 4. Note: 101376 is the frame size in pixels of a 352x288 frame also known as CIF ("Common Intermediate Format") frame size format.

For each frame, each position in the slice raster MUST be filled by one and only one slice of the frame (no missing slice position, no slice overlapping).

For each Frame with keyframe value of 0, each slice MUST have the same value of slice_x, slice_y, slice_width, slice_height as a slice in the previous frame, except if reset_contexts is 1.

6. Appendixes

6.1. Decoder implementation suggestions

6.1.1. Multi-threading support and independence of slices

The bitstream is parsable in two ways: in sequential order as described in this document or with the pre-analysis of the footer of each slice. Each slice footer contains a slice_size field so the boundary of each slice is computable without having to parse the slice content. That allows multi-threading as well as independence of slice content (a bitstream error in a slice header or slice content has no impact on the decoding of the other slices).

After having checked keyframe field, a decoder SHOULD parse slice_size fields, from slice_size of the last slice at the end of the frame up to slice_size of the first slice at the beginning of the frame, before parsing slices, in order to have slices boundaries. A decoder MAY fallback on sequential order e.g. in case of corrupted frame (frame size unknown, slice_size of slices not coherant...) or if there is no possibility of seek into the stream.

Architecture overwiew of slices in a frame:

```
+-----------------------------------------------------------------+
| first slice header                                              |
| first slice content                                             |
| first slice footer                                              |
| --------------------------------------------------------------- |
| second slice header                                             |
| second slice content                                            |
| second slice footer                                             |
| --------------------------------------------------------------- |
| ...                                                             |
| --------------------------------------------------------------- |
| last slice header                                               |
| last slice content                                              |
| last slice footer                                               |
+-----------------------------------------------------------------+
```

7. Changelog

See <https://github.com/FFmpeg/FFV1/commits/master>

8. ToDo

- o mean,k estimation for the golomb rice codes

9. Bibliography

9.1. References

RFC 2119 - Key words for use in RFCs to Indicate Requirement Levels <https://www.ietf.org/rfc/rfc2119.txt>

ISO/IEC 9899 - Programming languages - C <http://www.open-std.org/JTC1/SC22/WG14/www/standards>

JPEG-LS FCD 14495 <http://www.jpeg.org/public/fcd14495p.pdf>

H.264 Draft <http://bs.hhi.de/~wiegand/JVT-G050.pdf>

HuffYuv <http://cultact-server.novi.dk/kpo/huffyuv/huffyuv.html>

FFmpeg <http://ffmpeg.org>

JPEG2000 <http://www.jpeg.org/jpeg2000/>

Range encoding: an algorithm for removing redundancy from a digitised message. Presented by G. Nigel N. Martin at the Video & Data Recording Conference, IBM UK Scientific Center held in Southampton July 24-27 1979.

AVI RIFF File Format <https://msdn.microsoft.com/en-us/library/windows/desktop/dd318189%28v=vs.85%29.aspx>

Information technology Coding of audio-visual objects Part 12: ISO base media file format <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=61988>

NUT Open Container Format <http://www.ffmpeg.org/~michael/nut.txt>

10. Copyright

11. References

11.1. URIs

[1] https://tools.ietf.org/html/rfc2119

Author's Address

Michael Niedermayer