cellar
Internet-Draft
Intended status: Standards Track
Expires: 31 August 2026

S. Lhomme

M. Bunkus

D. Rice

27 February 2026

Matroska Media Container Tag Specifications
draft-ietf-cellar-tags-23

Abstract

Matroska is a multimedia container format. It can store timestamped multimedia data but also chapters and tags.

The Tag elements add important metadata to identify and classify the information found in a Matroska Segment.

This document defines the Matroska multimedia container tags, namely the tag names and their respective semantic meaning.

Status of This Memo

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

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

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

This Internet-Draft will expire on 31 August 2026.

Copyright Notice

Table of Contents

1. Introduction

The Tag elements can tag a whole Segment, separate Tracks elements, individual Chapter elements or Attachments elements.

Some details about tagging are already present in Section 24 of [RFC9559]. Readers of this document should be familiar with that section, the different high level parts of Matroska as defined in Section 4.5 of [RFC9559] and EBML Master Elements as defined in Section 7.7 of [RFC8794].

Matroska Tags consist of a name and a value. Unlike EBML or Matroska elements, they are not dependent on the version of the file. All tags defined in this document, and even others, can be used in all versions of Matroska.

While the Matroska tagging framework allows anyone to create their own custom tags, it is important to have a common set of values for interoperability.

2. Notation and Conventions

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

3. Tagging

When a SimpleTag is nested within another SimpleTag, the nested SimpleTag becomes an attribute of its parent SimpleTag. For instance, if one wanted to store the dates that a singer started being the lead performer, then your SimpleTag tree would look something like this:

* Targets
  - TagTrackUID = {track UID of tagged content}.
* ARTIST = "Band Name"
  - LEAD_PERFORMER = "John Doe"
    - DATE_STARTED = "1981-08"

This corresponds to this layout of EBML elements:

```
<Tags>
  <Tag>
    <Targets>
      <TagTrackUID>{track UID of tagged content}</TagTrackUID>
    </Targets>

    <SimpleTag>
      <TagName>ARTIST</TagName>
      <TagString>Band Name</TagString>

      <!-- sub tag(s) about the ARTIST -->
      <SimpleTag>
        <TagName>LEAD_PERFORMER</TagName>
        <TagString>John Doe</TagString>

        <!-- sub tag(s) about the LEAD_PERFORMER -->
        <SimpleTag>
          <TagName>DATE_STARTED</TagName>
          <TagString>1981-08</TagString>
        </SimpleTag>

      </SimpleTag>

    </SimpleTag>
  </Tag>
</Tags>
```

In this way, it becomes possible to store any SimpleTag as an attribute of another SimpleTag.

3.1. Why Assigned Tags Matter

There is a debate between people who think all tags should be free-form and those who think all tags should be limited to a set of names. The recommendations in this document are in between.

An application intended for advanced users might permit the insertion of any tag in a file. While this provides maximum flexibility, custom or exotic tags generally limit interoperable use. Well-known tags improve the ability of others to read and reuse them; most applications will therefore use a small list of useful tags.

So hopefully, when someone wants to put information in one’s file, they will find an assigned one, i.e., found in the IANA Matroska Tags Names registry Section 6.1, that fits their need and hopefully use it. If it is not in the list, this person can try to add a new tag in the registry. This registry is not meant to have every possible information in a file. Matroska files are not meant to 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 ought not be in a file because it doesn't make sense; thus, each request needs to be evaluated to determine if it makes sense to be carried over in a file for storage and/or sharing or if it doesn't belong there.

We also need an assigned list simply for developers 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 desired meaning so that other apps could understand the same meaning.

3.2. Tag Formatting

3.2.1. TagName Formatting

The TagName element can hold any UTF-8 data. However, to distinguish between the IANA assigned names and custom or private ones, a set of rules are defined.

Assigned TagName values MUST consist of latin capital letters, numbers and the underscore character '_'.

Assigned TagName values MUST NOT contain any space.

Assigned TagName values MUST NOT start with the underscore character '_'; see Section 3.1.

It is RECOMMENDED that tag names start with the underscore character '_' for unassigned tags that are not meant to be added to the list of assigned tags.

The syntax of assigned TagName values is defined using this Augmented Backus-Naur Form (ABNF) [RFC5234] notation:

```
TagName               = FirstCharacter [Character]

FirstCharacter        = CapitalLetter / DIGIT
Character             = CapitalLetter / DIGIT / Underscore

CapitalLetter         = %x41-5A  ; "A" to "Z"
Underscore            = %x5F     ; "_"
```

3.2.2. TagString Formatting

Although tags are metadata mostly used for reading, there are cases where the string value could be used for sorting, categorization, etc. For this reason, when possible, strict formatting of the value should be used to improve interoperability.

Multiple items SHOULD NOT be stored as a list in a single TagString. If there is more than one tag value with the same name to be stored, it is RECOMMENDED to use separate SimpleTags with that name for each value.

Preexisting files may have used multiple values in the same TagString but given there are no defined delimiters they cannot be easily split into multiple elements. INSTRUMENTS (Section 4.4) and KEYWORDS (Section 4.6) tags allow using a comma as a separator. However, it is RECOMMENDED to use separate SimpleTags with each containing a single instrument or keyword value, respectively.

Due to the varied nature of tag sources it may also not always possible to know programmatically whether a value is a list that must be split or not.

3.2.2.1. Date Tags Formatting

TagString fields defined in this document with dates MUST have the following format: "YYYY-MM-DD hh:mm:ss.mss" or a reduced version. The format is similar to the [ISO8601] date and time format defined in Appendix A of [RFC3339] without the "T" separator, without the time offset and with the addition of the milliseconds "mss". The date and times represented are in Coordinated Universal Time (UTC).

Date and times are usually not precise to a particular millisecond. To store less accurate dates, parts of the date string are removed starting from the right. For instance, to store only the year, one would use "2004". To store a specific day such as May 1st, 2003, one would use "2003-05-01".

The syntax of this tags-date-time is defined using this Augmented Backus-Naur Form (ABNF) [RFC5234]:

```
time-hour         = 2DIGIT ; 00-24
time-minute       = 2DIGIT ; 00-59
time-second       = 2DIGIT ; 00-58, 00-59, 00-60 based on
                           ; leap-second rules
time-millisecond  = 3DIGIT ; 000-999

timespec-hour     = time-hour
timespec-minute   = timespec-hour ":" time-minute
timespec-second   = timespec-minute ":" time-second
timespec-milli    = timespec-second "." time-millisecond

time              = timespec-hour / timespec-minute
                    / timespec-second / timespec-milli

date-fullyear   = 4DIGIT  ; 0000-9999
date-month      = 2DIGIT  ; 01-12
date-mday       = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31 based on
                          ; month/year

datespec-year     = date-fullyear
datespec-month    = datespec-year "-" date-month
datespec-full     = datespec-month "-" date-mday
datespec-time     = datespec-full " " time

tags-date-time    = datespec-time / datespec-full
                    / datespec-month / datespec-year
```

3.2.2.2. Number Tags Formatting

TagString fields that require a floating-point number MUST use the "." mark instead of the "," mark. Only ASCII numbers "0" to "9" and the "." character MUST be used. The "." separator represents the boundary between the integer value and the decimal parts. If the string doesn't contain the "." separator, the value is an integer value. Digit grouping delimiters MUST NOT be used. The integer value and decimal parts are values in base 10.

To display it differently for another locale, it is RECOMMENDED that applications support auto replacement on display. The thousand separator MAY be inserted for display purposes. The decimal separator "." MAY be replaced to match the user locale for display purposes.

In legacy media containers, it is possible that the "," character might have been used as a separator or that digit grouping delimiters might have been used. A Matroska Reader SHOULD use the following character handling to parse such legacy formats:

* if multiple instances of the same non-number character are found, they are ignored,
* if only one "." character is found and no other non-number character is found, the "." is the integer-decimal separator,
* if only one "," character is found and no other non-number character is found, the "," is a digit grouping delimiter,
* any other non-number character is ignored.

3.2.2.3. Country Code Tags Formatting

TagString fields that use a Country Code MUST use the Matroska countries form defined in Section 13 of [RFC9559], i.e., [RFC5646] two-letter region subtags, without the UK exception.

3.3. Target Types

The TargetTypeValue element allows tagging of different parts that are within or outside a given file. For example, in an audio file with one song you could have information about the CD set album it comes from, even if the whole CD set is not found in the file.

For applications to know the kind of information (e.g., "TITLE") relates to a certain level (CD title or track title), we also need a set of official TargetTypeValue values and TargetType names. That also means the same tag name can have different meanings depending on its TargetTypeValue, otherwise we would end up with 7 "TITLE_" tag names.

For human readability, a TargetType string can be added next to the corresponding TargetTypeValue. Audio and video have different TargetType values. The following table summarizes the TargetType values found in Section 5.1.8.1.1.2 of [RFC9559] for audio and video content:

TargetTypeValue	Audio TargetType	Comment
70	COLLECTION	the high hierarchy consisting of many different lower items
60	EDITION / ISSUE / VOLUME / OPUS	a list of lower levels grouped together
50	ALBUM / OPERA / CONCERT	the most common grouping level of music (e.g., an album)
40	PART / SESSION	when an album has different logical parts
30	TRACK / SONG	the common parts of an album
20	SUBTRACK / PART / MOVEMENT	corresponds to parts of a track for audio (e.g., a movement)
10	-	the lowest hierarchy found in music

Table 1: TargetTypeValue Values Audio Semantic Description

TargetTypeValue	Video TargetType	Comment
70	COLLECTION	the high hierarchy consisting of many different lower items
60	SEASON / SEQUEL / VOLUME	a list of lower levels grouped together
50	MOVIE / EPISODE / CONCERT	the most common grouping level of video (e.g., an episode for TV series)
40	PART / SESSION	when an episode has different logical parts
30	CHAPTER	the common parts of a movie or episode
20	SCENE	a sequence of continuous action in a film or video
10	SHOT	the lowest hierarchy found in movies

Table 2: TargetTypeValue Values Video Semantic Description

A tag defined for a given TargetTypeValue also applies to all Targets with numerically smaller TargetTypeValues in the same hierarchy, that is, from higher-level groups to lower-level entities. This means that if a CD has the same artist for all tracks, you just need to set the "ARTIST" tag at TargetTypeValue 50 (ALBUM) and not to each TargetTypeValue 30 (TRACK), but you can also repeat the value for each track. If some tracks of that CD have no known "ARTIST", the value MUST be set to an empty string ("") as detailed in Section 24.2 of [RFC9559], so that the album "ARTIST" doesn't apply.

If a tag with a given TagName is found at a TargetTypeValue, only values of that TagName are valid at that TargetTypeValue level. In other words, the TagName values from upper TargetTypeValue levels don't apply at that level.

Multiple SimpleTags with the same TagName can be used at a given TargetTypeValue level when each SimpleTag contain a TagString. For example this can be useful to find a single "ARTIST" even when they

are found in a collaboration. The concatenation of each TagString represents the value for the TagName at this level. The presentation, for instance with a separator, is up to the application.

3.3.1. Target Types Parts

There are three organizational tags defined in Section 4.2:

* TOTAL_PARTS

* PART_NUMBER

* PART_OFFSET

These tags allow specifying the ordering of some tags within another group of tags.

For example if an album has 10 tracks, tag the second track from it, one would set "TOTAL_PARTS" to "10" at TargetTypeValue 50 (ALBUM). It means the "ALBUM" contains 10 lower parts. The lower part in question is the first lower TargetTypeValue that is specified in the file. TargetTypeValue = 30 (TRACK) would mean the album contains 10 tracks. TargetTypeValue = 20 (MOVEMENT) would mean the album contains 10 movements, etc. And since it’s the second track within the album, the "PART_NUMBER" at TargetTypeValue 30 (TRACK) is set to "2".

If the parts are split into multiple logical entities, you can also use "PART_OFFSET". For example you are tagging the third track of the second CD of a double CD album with a total of 10 tracks the "TOTAL_PARTS" at TargetTypeValue 50 (ALBUM) is "10", the "PART_NUMBER" at TargetTypeValue 30 (TRACK) is "3", and the "PART_OFFSET" at TargetTypeValue 30 (TRACK) is "5", which is the number of tracks on the first CD.

When a TargetTypeValue 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 an audio record with 2 tracks in a single file. The name of the record is also the name of the first track "Main Title". There is one Tag element for the record, and one Tag element per track on the record. Each track is identified by a chapter via a TagChapterUID element which corresponds to the UID of a ChapterUID as defined in Section 5.1.8.1.1.5 of [RFC9559].

The Tag for the record:

* Targets
  - TargetTypeValue = 50
* ARTIST = "Dummy Artist Name"
* TITLE = "Main Title"
* TOTAL_PARTS = "2"

The Tag for the first track:

* Targets
  - TargetTypeValue = 30
  - TagChapterUID = 12345
* TITLE = "Main Title"
* PART_NUMBER = "1"

The Tag for the second track:

* Targets
  - TargetTypeValue = 30
  - TagChapterUID = 67890
* TITLE = "B-side Track Name"
* PART_NUMBER = "2"

This corresponds to this layout of EBML elements:

```
<Tags>
  <!-- description of the whole file/record -->
  <Tag>
    <Targets>
      <TargetTypeValue>50</TargetTypeValue>
    </Targets>

    <SimpleTag>
      <TagName>ARTIST</TagName>
      <TagString>Dummy Artist Name</TagString>
    </SimpleTag>
```

```
      <SimpleTag>
        <TagName>TITLE</TagName>
        <TagString>Main Title</TagString>
      </SimpleTag>

      <SimpleTag>
        <TagName>TOTAL_PARTS</TagName>
        <TagString>2</TagString>
      </SimpleTag>
    </Tag>

    <!-- description of the first track/chapter -->
    <Tag>
      <Targets>
        <TargetTypeValue>30</TargetTypeValue>
        <TagChapterUID>12345</TagChapterUID>
      </Targets>

      <SimpleTag>
        <TagName>TITLE</TagName>
        <TagString>Main Title</TagString>
      </SimpleTag>

      <SimpleTag>
        <TagName>PART_NUMBER</TagName>
        <TagString>1</TagString>
      </SimpleTag>
    </Tag>

    <!-- description of the second track/chapter -->
    <Tag>
      <Targets>
        <TargetTypeValue>30</TargetTypeValue>
        <TagChapterUID>67890</TagChapterUID>
      </Targets>

      <SimpleTag>
        <TagName>TITLE</TagName>
        <TagString>B-side Track Name</TagString>
      </SimpleTag>

      <SimpleTag>
        <TagName>PART_NUMBER</TagName>
        <TagString>2</TagString>
      </SimpleTag>
    </Tag>
  </Tags>
```

Here is an example using the "PART_OFFSET" tag. It corresponds to a file that contains the third track on the second CD of the 2-CD album:

The Tag for the album:

* Targets
  - TargetTypeValue = 50
* ARTIST = "The Album Artist"
  - SORT_WITH = "Album Artist, The"
* TITLE = "Album Title"
* TOTAL_PARTS = "10"

The Tag for the third track of the second CD:

* Targets
  - TargetTypeValue = 30
* TITLE = "Some Track Title"
* PART_NUMBER = "3"
* PART_OFFSET = "5"

This corresponds to this layout of EBML elements:

```
<Tags>
 <!-- description of the whole album -->
 <Tag>
  <Targets>
   <TargetTypeValue>50</TargetTypeValue>
  </Targets>

  <SimpleTag>
   <TagName>ARTIST</TagName>
   <TagString>The Album Artist</TagString>

   <SimpleTag>
    <TagName>SORT_WITH</TagName>
    <TagString>Album Artist, The</TagString>
   </SimpleTag>
  </SimpleTag>
```

```
     <SimpleTag>
      <TagName>TITLE</TagName>
      <TagString>Album Title</TagString>
     </SimpleTag>

     <!-- the number of sub elements in this album (10 tracks) -->
     <SimpleTag>
      <TagName>TOTAL_PARTS</TagName>
      <TagString>10</TagString>
     </SimpleTag>
    </Tag>

    <!-- description of the third track of the second CD -->
    <Tag>
     <Targets>
      <TargetTypeValue>30</TargetTypeValue>
     </Targets>

     <SimpleTag>
      <TagName>TITLE</TagName>
      <TagString>Some Track Title</TagString>
     </SimpleTag>

     <!-- This is the third track of the second CD -->
     <SimpleTag>
      <TagName>PART_NUMBER</TagName>
      <TagString>3</TagString>
     </SimpleTag>

     <!-- The first CD contains 5 tracks -->
     <SimpleTag>
      <TagName>PART_OFFSET</TagName>
      <TagString>5</TagString>
     </SimpleTag>
    </Tag>
   </Tags>
```

3.4. Multiple Targets UID

A Tag element has a single Targets element with a single TargetTypeValue element. However, the Targets element can contain various TagTrackUID, TagEditionUID, TagChapterUID and TagAttachmentUID elements.

When multiple values are found using the same Tag UID element (e.g., TagTrackUID) a logical OR is applied on these elements. In other words, the tags apply to each entity defined by a UID. This is the list of UIDs the tags apply to (e.g., list of TagTrackUID). Such a list may contain a single UID element.

When different lists of Tag UID elements are found (e.g., a list of TagTrackUID and a list of TagChapterUID) a logical AND is applied between those lists. In other words, the tags apply only to the entities matching a UID in each list of Tag UID elements.

These operations allow factorizing tags that would otherwise need to be repeated multiple times.

Here is an example of a Tag applying to 2 chapters, using the same example as in Section 3.3.1:

* Targets
  - TargetTypeValue = 30
  - TagChapterUID = 12345
  - TagChapterUID = 67890
* WRITTEN_BY = "John Doe"
* WRITTEN_BY = "Jane Smith"
* PRODUCER = "John Doe"
* PRODUCER = "Jane Smith"

This corresponds to this layout of EBML elements:

```
<Tags>
  <Tag>
    <Targets>
      <TargetTypeValue>30</TargetTypeValue>

      <!-- chapter with Main Title -->
      <TagChapterUID>12345</TagChapterUID>

      <!-- chapter with B-side Track Name -->
      <TagChapterUID>67890</TagChapterUID>
    </Targets>

    <!-- first writer of Main Title and B-side Track Name -->
    <SimpleTag>
      <TagName>WRITTEN_BY</TagName>
      <TagString>John Doe</TagString>
    </SimpleTag>

    <!-- second writer of Main Title and B-side Track Name -->
    <SimpleTag>
      <TagName>WRITTEN_BY</TagName>
      <TagString>Jane Smith</TagString>
    </SimpleTag>

    <!-- first producer of Main Title and B-side Track Name -->
    <SimpleTag>
      <TagName>PRODUCER</TagName>
      <TagString>John Doe</TagString>
    </SimpleTag>

    <!-- second producer of Main Title and B-side Track Name -->
    <SimpleTag>
      <TagName>PRODUCER</TagName>
      <TagString>Jane Smith</TagString>
    </SimpleTag>
  </Tag>
</Tags>
```

Some combinations of different Tag UID elements are not possible.

A TagChapterUID and TagAttachmentUID can’t be mixed because there is no overlap with a Chapter and an Attachment that would make sense. An attachment applies to the whole segment and can be tied to tracks, via \Segment\Tracks\TrackEntry\AttachmentLink as defined in Section 5.1.4.1.24 of [RFC9559], but not chapters.

Mixing TagEditionUID and TagChapterUID elements is also not useful because each Chapter UID would need to be in one of the Chapter Edition UID. That would be the same as not using the list of TagEditionUID at all.

The following table shows the allowed combinations between lists of Tag UID elements:

UID elements	Track	Edition	Chapter	Attachment
Track	YES	YES	YES	with matching AttachmentLink
Edition	YES	YES	NO	YES
Chapter	YES	NO	YES	NO
Attachment	with matching AttachmentLink	YES	NO	YES

Table 3: Tag UID elements allowed combinations

Here is an example of a Tag applying to a single track and a single chapter. It represents the composer of the music in a part of a movie. The file may contain a second audio track with audio commentary not including that music, so we only tag the track with the music.

* Targets
  - TargetTypeValue = 30
  - TagTrackUID = 123
  - TagChapterUID = 987654321
* COMPOSER = "Jane Smith"

This corresponds to this layout of EBML elements:

```
<Tags>
  <Tag>
    <Targets>
      <TargetTypeValue>30</TargetTypeValue>

      <!-- chapter with the music -->
      <TagTrackUID>123</TagTrackUID>

      <!-- track with the music -->
      <TagChapterUID>67890</TagChapterUID>
    </Targets>

    <!-- composer of the music in that chapter for that track -->
    <SimpleTag>
      <TagName>COMPOSER</TagName>
      <TagString>Jane Smith</TagString>
    </SimpleTag>

  </Tag>
</Tags>
```

4. Assigned Tags

The following is the initial list of assigned Matroska Tags. As stated in Section 3.1 it is better to use assigned tags as much as possible, otherwise compatibility will be compromised. If you find that there is a Tag missing that you would like to use, then please contact the persons mentioned in the IANA Matroska Tags Registry for its inclusion; see Section 6.1.

4.1. Nesting Information

Nesting Information tags are tags that usually contain any other tags.

Tag Name	Type	Description
ORIGINAL	nested	A special tag that is meant to have other tags inside (using nested tags) to describe the original work of art that this item is based on.
SAMPLE	nested	A tag that contains other tags to describe a sample used in the targeted item originally found in another work of art.
COUNTRY	UTF-8	The name of the country that is meant to have other tags inside (using nested tags) with country specific information about the item, using the Country Code format defined in Section 3.2.2.3.

Table 4: Nesting Information tags

4.2. Organization Information

All tags in this section express hierarchy defined in Section 3.3.1.

Tag Name	Type	Description
TOTAL_PARTS	UTF-8	Total number of parts defined at the first lower level. (e.g., if TargetTypeValue is "50" (TargetType = "ALBUM"), the total number of tracks of an audio CD).
PART_NUMBER	UTF-8	Index of the current part relative to parts of the same level, starting at 1. (e.g., if TargetTypeValue is "30" (TargetType = "TRACK"), the track number of an audio CD).
PART_OFFSET	UTF-8	A number to add to "PART_NUMBER", when the parts at that level don’t start at 1 (e.g., if TargetTypeValue is "30" (TargetType = "TRACK"), the track number of the second audio CD).

Table 5: Organization Information tags

4.3. Titles

Tag Name	Type	Description
TITLE	UTF-8	The title of this item. For example, for music you might label this "Canon in D", or for video's audio track you might use "English 5.1" This is akin to the "TIT2" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
SUBTITLE	UTF-8	Sub Title of the entity. This is akin to the "TIT3" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).

Table 6: Titles tags

4.4. Nested Information

Nested Information tags are tags providing information about their parent tags.

Tag Name	Type	Description
URL	UTF-8	URL corresponding to the tag it is included in, using the format defined in [RFC3986].
SORT_WITH	UTF-8	A child SimpleTag element to indicate what alternative value the parent SimpleTag element can have to be sorted -- for example, "Band" instead of "The Band". Or "Doe John" and "Doe John H." (no comma needed).
INSTRUMENTS	UTF-8	The instruments that are being used/ played, separated by a comma. It MUST be a child of another tag, including the following: "ARTIST", "LEAD_PERFORMER", or "ACCOMPANIMENT".
EMAIL	UTF-8	Email corresponding to the tag it is included in, using the "Addr-Spec" format defined in Section 3.4.1 of [RFC5322].
ADDRESS	UTF-8	The physical address of the entity. The address MAY include a country code. If a country code is included, it is RECOMMENDED to use the Country Code format defined in Section 3.2.2.3. It can be useful for a recording label.
FAX	UTF-8	The fax number corresponding to the tag it is included in. It can be useful for a recording label.
PHONE	UTF-8	The phone number corresponding to the tag it is included in. It can be useful for a recording label.

Table 7: Nested Information tags

4.5. Entities

Tag Name	Type	Description
ARTIST	UTF-8	A person or band/collective generally considered responsible for the work. This is akin to the "TPE1" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
LEAD_PERFORMER	UTF-8	Lead Performer/Soloist(s). This can sometimes be the same as "ARTIST". This is akin to the "TPE1" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
ACCOMPANIMENT	UTF-8	Band/orchestra/accompaniment/ musician. This is akin to the "TPE2" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
COMPOSER	UTF-8	The name of one composer of this item. This is akin to the "TCOM" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
ARRANGER	UTF-8	The name of a person who arranged the piece (e.g., Ravel).
LYRICS	UTF-8	The lyrics corresponding to a song, in case audio synchronization is not known or as a duplicate of a subtitle track. Editing this value, when it is a duplicate of a subtitle track, SHOULD also result in editing the subtitle track for more consistency, and vice versa.

LYRICIST	UTF-8	The name of a person who wrote the lyrics for a musical item. This is akin to the "TEXT" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
CONDUCTOR	UTF-8	Conductor/performer refinement. This is akin to the "TPE3" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
DIRECTOR	UTF-8	The name of a director of a movie. This is akin to the "IART" tag [RIFF.tags].
ASSISTANT_DIRECTOR	UTF-8	The name of the assistant director.
DIRECTOR_OF_PHOTOGRAPHY	UTF-8	The name of the director of photography, also known as cinematographer. This is akin to the "ICNM" tag in [RIFF.tags].
SOUND_ENGINEER	UTF-8	The name of the sound engineer or sound recordist.
ART_DIRECTOR	UTF-8	The person who oversees the artists and craftspeople who build the sets.
PRODUCTION_DESIGNER	UTF-8	Artist responsible for designing the overall visual appearance of a movie.
CHOREGRAPHER	UTF-8	The name of the choreographer.
COSTUME_DESIGNER	UTF-8	The name of the costume designer.
ACTOR	UTF-8	An actor or actress playing a role in this movie. This is the person’s real name, not the character’s name the

		person is playing.
CHARACTER	UTF-8	The name of the character an actor or actress plays in this movie. This SHOULD be a sub-tag of an ACTOR tag in order to not cause ambiguities.
WRITTEN_BY	UTF-8	The author of the story or script (used for movies and TV shows).
SCREENPLAY_BY	UTF-8	The author of the screenplay or scenario (used for movies and TV shows).
EDITED_BY	UTF-8	The name of a film editor for a movie. This is akin to the "IEDT" tag in [RIFF.tags].
PRODUCER	UTF-8	The name of a producer for a song/movie. This is akin to the "IPRO" tag in [RIFF.tags].
COPRODUCER	UTF-8	The name of a co-producer.
EXECUTIVE_PRODUCER	UTF-8	The name of an executive producer.
DISTRIBUTED_BY	UTF-8	The name of a company distributing the content. This is akin to the "IDST" tag in [RIFF.tags].
MASTERED_BY	UTF-8	The engineer who mastered the content for a physical medium or for digital distribution.
ENCODED_BY	UTF-8	This is akin to the "TENC" tag in [ID3v2.3].
MIXED_BY	UTF-8	DJ mix by the artist specified.
REMIXED_BY	UTF-8	Interpreted, remixed, or otherwise modified by. This

		is akin to the "TPE4" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
PRODUCTION_STUDIO	UTF-8	The name of a physical studio where the content was recorded. This is akin to the "ISTD" tag in [RIFF.tags].
THANKS_TO	UTF-8	A very general tag for everyone else that wants to be listed.
PUBLISHER	UTF-8	This is akin to the "TPUB" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
LABEL	UTF-8	The record label or imprint on the disc.

Table 8: Entities tags

4.6. Search and Classification

Tag Name	Type	Description
GENRE	UTF-8	The main genre (classical, ambient-house, synthpop, sci-fi, drama, etc.). The format follows the "TCON" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
MOOD	UTF-8	Intended to reflect the mood of the item with a few keywords (e.g., "Romantic", "Sad" or "Uplifting"). The format follows that of the "TMOO" tag in [ID3v2.4] when the TargetTypeValue is 30 (TRACK).
ORIGINAL_MEDIA_TYPE	UTF-8	Describes the original type of the media, such as, "DVD",

		"CD", "computer image," "drawing," "lithograph," and so forth. This is akin to the "TMED" tag in [ID3v2.4].
CONTENT_TYPE	UTF-8	The type of the item (e.g., Documentary, Feature Film, Cartoon, Music Video, Music, Sound FX).
SUBJECT	UTF-8	Describes the topic of the file, such as "Aerial view of Seattle."
DESCRIPTION	UTF-8	A short description of the content, such as "Two birds flying."
KEYWORDS	UTF-8	Keywords to the item separated by a comma, used for searching.
SUMMARY	UTF-8	A plot outline or a summary of the story.
SYNOPSIS	UTF-8	A description of the story line of the item.
INITIAL_KEY	UTF-8	The initial key that a musical track starts in. The format is identical to "TKEY" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
PERIOD	UTF-8	Describes the period that the piece is from or about. For example, "Renaissance".
LAW_RATING	UTF-8	Depending on the "COUNTRY" it is the format of the rating of a movie (P, R, X in the USA, an age in other countries or a URI defining a logo).

Table 9: Search and Classification tags

4.7. Temporal Information

All tags in this section use the Date format defined in Section 3.2.2.1.

Tag Name	Type	Description
DATE_RELEASED	UTF-8	The time that the item was originally released. This is akin to the "TDRL" tag in [ID3v2.4] when the TargetTypeValue is 30 (TRACK).
DATE_RECORDED	UTF-8	The time that the recording began. This is akin to the "TDRC" tag in [ID3v2.4] when the TargetTypeValue is 30 (TRACK).
DATE_ENCODED	UTF-8	The time that the encoding of this item was completed began. This is akin to the "TDEN" tag in [ID3v2.4] when the TargetTypeValue is 30 (TRACK).
DATE_TAGGED	UTF-8	The time that the tags were done for this item. This is akin to the "TDTG" tag in [ID3v2.4] when the TargetTypeValue is 30 (TRACK).
DATE_DIGITIZED	UTF-8	The time that the item was transferred to a digital medium. This is akin to the "IDIT" tag in [RIFF.tags].
DATE_WRITTEN	UTF-8	The time that the writing of the music/script began.
DATE_PURCHASED	UTF-8	Information on when the file was purchased; see also Section 4.12 on purchase tags.
DATE_STARTED	UTF-8	When the information of the parent SimpleTag element starts being valid. The information of the parent SimpleTag element is only valid between this date and the "DATE_ENDED" date of the same level. The "DATE_ENDED" is OPTIONAL. If empty or omitted the end date is unknown.

DATE_ENDED	UTF-8	When the information is not valid anymore. The information of the parent SimpleTag element is only valid between the "DATE_STARTED" date of the same level and this date. The "DATE_STARTED" is OPTIONAL. If empty or omitted the start date is unknown.

Table 10: Temporal Information tags

4.8. Spatial Information

Tag Name	Type	Description
RECORDING_LOCATION	UTF-8	The location where the item was recorded, using the Country Code format defined in Section 3.2.2.3. This code is followed by a comma, then more detailed information such as state/province, another comma, and then city. For example, "US, Texas, Austin". This will allow for easy sorting. It is okay to only store the country, or the country and the state/province. More detailed information can be added after the city through the use of additional commas. In cases where the province/state is unknown, but you want to store the city, simply leave a space between the two commas. For example, "US, , Austin".
COMPOSITION_LOCATION	UTF-8	Location that the item was originally designed/written, using the Country Code format defined in Section 3.2.2.3. This code is followed by a comma, then more detailed information such as state/province, another comma, and then city. For example, "US, Texas, Austin". This will allow for easy sorting. It is okay to

		only store the country, or the country and the state/province. More detailed information can be added after the city through the use of additional commas. In cases where the province/state is unknown, but you want to store the city, simply leave a space between the two commas. For example, "US, , Austin".
COMPOSER_NATIONALITY	UTF-8	Nationality of the main composer of the item, mostly for classical music, using the Country Code format defined in Section 3.2.2.3.

Table 11: Spatial Information tags

4.9. User Information

All tags in this section are personal to the user of these files.

Tag Name	Type	Description
COMMENT	UTF-8	Any comment related to the content.
PLAY_COUNTER	UTF-8	The number of times the item has been played.
RATING	UTF-8	A numeric value defining how much a person likes the song/movie. The number is between 0 and 5, stored using the Float number defined in Section 3.2.2.2 (e.g., 2.7), 5(.0) being the highest possible rating. Other rating systems with different ranges will have to be scaled.

Table 12: User Information tags

4.10. Technical Information

These tags represent values that could be parsed to handle the playback better. For historical reasons they are usually stored as strings but a stricter binary format is RECOMMENDED for new tags.

Tag Name	Type	Description
ENCODER	UTF-8	The software or hardware used to encode this item. ("LAME" or "XviD")
ENCODER_SETTINGS	UTF-8	A list of the settings used for encoding this item. No specific format.
BPS	UTF-8	The average bits per second of the specified item stored using the Float number defined in Section 3.2.2.2. This is only the data in the Block(s), and excludes headers and any container overhead.
FPS	UTF-8	The average frames per second of the specified item. This is typically the average number of Blocks per second stored using the Float number defined in Section 3.2.2.2. In the event that lacing is used, each laced chunk is to be counted as a separate frame.
BPM	UTF-8	Average number of beats per minute in

		the complete target (e.g., a chapter) stored using the Float number defined in Section 3.2.2.2.
MEASURE	UTF-8	In music, a measure is a unit of time in Western music like "4/4". It represents a regular grouping of beats, a meter, as indicated in musical notation by the time signature. The majority of the contemporary rock and pop music you hear on the radio these days is written in the 4/4 time signature.
TUNING	UTF-8	It is saved as a frequency in hertz to allow near-perfect tuning of instruments to the same tone as the musical piece (e.g., "441.34" in Hertz). The value is stored using the Float number defined in Section 3.2.2.2.
REPLAYGAIN_GAIN	UTF-8	The gain to apply to reach 89 dB SPL (Sound Pressure Level in decibels) on playback. The value is computed according to the [ReplayGain] standard. The value in decibels (dB) is stored as a string (e.g., "-0.42 dB"), using the Float number defined in Section 3.2.2.2. The

		decibel unit is OPTIONAL. There MAY be a space between the number and the decibel unit. Note that ReplayGain information can be found at all TargetType levels (track, album, etc.).
REPLAYGAIN_PEAK	UTF-8	The maximum absolute peak amplitude of the item. The value is computed according to the [ReplayGain] standard. The value is a normalized absolute sample value of the target audio, using the Float number defined in Section 3.2.2.2 (e.g., "1.0129"). Note that ReplayGain information can be found at all TargetType levels (track, album, etc.).
EBU_R128_LOUDNESS	binary	EBU R 128 Loudness. The value is the Loudness relative to nominal full scale in LUFS (Loudness Units Full Scale) normalized to a Target Level of -23.0 LUFS as defined in [EBU-R.128]. This value is stored as a floating-point number in the 32-bit and 64-bit binary interchange format, as defined in [IEEE.754]. It is similar to a EBML

			floating number value Section 7.3 of [RFC8794].
EBU_R128_MAX_TRUE_PEAK	binary	EBU R 128 Maximum True Peak Level. This corresponds to the maximum value of the audio signal waveform of a programme in the continuous time domain, measured in dB True Peak (dBTP), as defined in [EBU-R.128]. This value is stored as a floating-point number in the 32-bit and 64-bit binary interchange format, as defined in [IEEE.754]. It is similar to a EBML floating number value Section 7.3 of [RFC8794].
EBU_R128_LOUDNESS_RANGE	binary	EBU R 128 Loudness Range, measures the variation in a time-varying loudness measurement, in LU (Loudness Units) as defined in [EBU-TECH.3342]. This value is stored as a floating-point number in the 32-bit and 64-bit binary interchange format, as defined in [IEEE.754]. It is similar to a EBML floating number value Section 7.3 of [RFC8794].

EBU_R128_MAX_MOMENTARY_LOUDNESS	binary	EBU R 128 Maximum Momentary Loudness, measures the variation of loudness on a 0.4 s sliding rectangular window, in LUFS (Loudness Units Full Scale) as defined in [EBU-TECH.3341]. This value is stored as a floating-point number in the 32-bit and 64-bit binary interchange format, as defined in [IEEE.754]. It is similar to a EBML floating number value Section 7.3 of [RFC8794].
EBU_R128_MAX_SHORT_LOUDNESS	binary	EBU R 128 Maximum Short-Term Loudness, measures the variation of loudness on a 3 s sliding rectangular window, in LUFS (Loudness Units Full Scale) as defined in [EBU-TECH.3341]. This value is stored as a floating-point number in the 32-bit and 64-bit binary interchange format, as defined in [IEEE.754]. It is similar to a EBML floating number value Section 7.3 of [RFC8794].

Table 13: Technical Information tags

4.11. External Identifiers

Tag Name	Type	Description
ISRC	UTF-8	The International Standard Recording Code [ISRC], excluding the "ISRC" prefix and including hyphens.
MCDI	binary	This is a binary dump of the TOC of the CD-ROM that this item was taken from. This holds the same information as the "MCDI" in [ID3v2.3] when the TargetTypeValue is 50 (ALBUM).
ISBN	UTF-8	International Standard Book Number [ISBN].
BARCODE	UTF-8	European Article Numbering EAN-13 barcode defined in [GS1] General Specifications.
CATALOG_NUMBER	UTF-8	A label-specific string used to identify the release -- for example, TIC 01.
LABEL_CODE	UTF-8	A 4-digit or 5-digit number to identify the record label, typically printed as (LC) xxxx or (LC) 0xxxx on CDs medias or covers (only the number is stored).
LCCN	UTF-8	United States of America Library of Congress Control Number [LCCN].
IMDB	UTF-8	Internet Movie Database [IMDb] title identifier. "tt" followed by at least 7 digits for Movies, TV Shows, and Episodes.
TMDB	UTF-8	The Movie DB "movie_id" or "tv_id" identifier for movies/TV shows [MovieDB]. The variable length digits string MUST be prefixed with either "movie/" or "tv/".

TVDB	UTF-8	The TV Database "Series ID" or "Episode ID" identifier for TV shows [TheTVDB]. Variable length all-digits string identifying a TV Show to use with the "series/{id}" API.
TVDB2	UTF-8	The TV Database [TheTVDB] tag which can include movies. The variable length digits string representing a "Series ID", "Episode ID" or "Movie ID" identifier MUST be prefixed with "series/", "episodes/", or "movies/", respectively.

Table 14: External Identifiers tags

4.12. Commercial

Tag Name	Type	Description
PURCHASE_ITEM	UTF-8	URL to purchase this file using the URL format defined in [RFC3986]. This is akin to the "WPAY" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
PURCHASE_INFO	UTF-8	Information on where to purchase this album using the URL format defined in [RFC3986]. This is akin to the "WCOM" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
PURCHASE_OWNER	UTF-8	Information on the person who purchased the file. This is akin to the "TOWN" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
PURCHASE_PRICE	UTF-8	The amount paid for entity, using the Float number defined in Section 3.2.2.2. The currency is not included. For instance, you would store "15.59" instead of "$15.59USD".
PURCHASE_CURRENCY	UTF-8	The currency type used to pay for

			the entity. Use [ISO4217] for the 3 letter alphabetic code.

Table 15: Commercial tags

4.13. Legal

Tag Name	Type	Description
COPYRIGHT	UTF-8	The copyright information as per the copyright holder. This is akin to the "TCOP" tag in [ID3v2.3] when the TargetTypeValue is 30 (TRACK).
PRODUCTION_COPYRIGHT	UTF-8	The copyright information as per the production copyright holder. This is akin to the "TPRO" tag in [ID3v2.4] when the TargetTypeValue is 30 (TRACK).
LICENSE	UTF-8	The license applied to the content (e.g., Creative Commons variants).
TERMS_OF_USE	UTF-8	The terms of use for this item. This is akin to the "USER" tag in [ID3v2.3].

Table 16: Legal tags

5. Security Considerations

This document inherits security considerations from the EBML [RFC8794] and Matroska [RFC9559] documents.

Tag values can be either TagString or TagBinary blobs. In both cases issues can happen if the parsing of the data fails.

Most of the time strings are kept as-is and don’t pose a security issue, apart from invalid UTF-8 values. Implementations MUST validate TagString inputs for UTF-8 correctness and reasonable length before use, in accordance with the security considerations in Section 10 of [RFC3629] and Section 7 of [RFC9839].

String tags that are parsed (such as "REPLAYGAIN_GAIN" or "REPLAYGAIN_PEAK" defined in Section 4.10), string tags following the TagString formatting rules Section 3.2.2, or string tags following other strict formats like URLs, may cause issues when the string is bogus or in an unexpected format.

Binary tags that need to be parsed (such as "MCDI" defined in Section 4.11) may cause issues when the data is bogus or incomplete.

Some tags like "URL" (Section 4.4) and "PURCHASE_URL" (Section 4.12) contain a URL. Bogus or altered URLs may direct the user to unwanted places.

Due to the nature of nested SimpleTag, it is possible to exhaust the memory of the host app by using very deep nesting. A host app MAY add some limits to the amount of nesting possible to avoid such issues.

Some elements found in Section 4.4 and Section 4.9 may contain physical addresses, email, etc. about a person. Care should be taken to ensure not to provide such files to people that ought not have this information when it’s not public knowledge. This can be achieved by either removing personal information or by controlling the diffusion of files containing these pieces of information.

6. IANA Considerations

6.1. Matroska Tags Names Registry

IANA has created a new registry called the "Matroska Tag Names" registry.

To register a new Tag Name in this registry, one needs a Name, a Type, a Change Controller, and an optional Reference to a document describing the Element ID.

The Name corresponds to the value stored in the TagName element. The Name is written in all latin capital letters, numbers and the underscore character ’_’ as defined in Section 3.2. The Name must not start with the underscore character ’_’.

The Type corresponds to which element will be stored the tag value. There can be 3 values for the Type:

* UTF-8: the value of the Tag is stored in TagString,

Matroska Tag Names for UTF-8 data are to be allocated according to the "First Come First Served" policy [RFC8126].

* binary: the value of the Tag is stored in TagBinary,

Matroska Tag Names for binary data are to be allocated according to the "Specification Required" policy [RFC8126].

* nested: the tag doesn’t contain a value, i.e., neither a TagBinary nor a TagString child element, only SimpleTag child elements inside.

Matroska Tag Names for nested tags are to be allocated according to the "Specification Required" policy [RFC8126].

Matroska Tag Names Values found in this document are assigned as initial values as follows:

Tag Name	Tag Type	Reference
ORIGINAL	nested	This document, Section 4.1
SAMPLE	nested	This document, Section 4.1
COUNTRY	UTF-8	This document, Section 4.1
TOTAL_PARTS	UTF-8	This document, Section 4.2
PART_NUMBER	UTF-8	This document, Section 4.2
PART_OFFSET	UTF-8	This document, Section 4.2
TITLE	UTF-8	This document, Section 4.3
SUBTITLE	UTF-8	This document, Section 4.3
URL	UTF-8	This document, Section 4.4
SORT_WITH	UTF-8	This document, Section 4.4

	INSTRUMENTS	UTF-8	This document, Section 4.4
	EMAIL	UTF-8	This document, Section 4.4
	ADDRESS	UTF-8	This document, Section 4.4
	FAX	UTF-8	This document, Section 4.4
	PHONE	UTF-8	This document, Section 4.4
	ARTIST	UTF-8	This document, Section 4.5
	LEAD_PERFORMER	UTF-8	This document, Section 4.5
	ACCOMPANIMENT	UTF-8	This document, Section 4.5
	COMPOSER	UTF-8	This document, Section 4.5
	ARRANGER	UTF-8	This document, Section 4.5
	LYRICS	UTF-8	This document, Section 4.5
	LYRICIST	UTF-8	This document, Section 4.5
	CONDUCTOR	UTF-8	This document, Section 4.5
	DIRECTOR	UTF-8	This document, Section 4.5
	ASSISTANT_DIRECTOR	UTF-8	This document, Section 4.5
	DIRECTOR_OF_PHOTOGRAPHY	UTF-8	This document, Section 4.5

SOUND_ENGINEER	UTF-8	This document, Section 4.5
ART_DIRECTOR	UTF-8	This document, Section 4.5
PRODUCTION_DESIGNER	UTF-8	This document, Section 4.5
CHOREGRAPHER	UTF-8	This document, Section 4.5
COSTUME_DESIGNER	UTF-8	This document, Section 4.5
ACTOR	UTF-8	This document, Section 4.5
CHARACTER	UTF-8	This document, Section 4.5
WRITTEN_BY	UTF-8	This document, Section 4.5
SCREENPLAY_BY	UTF-8	This document, Section 4.5
EDITED_BY	UTF-8	This document, Section 4.5
PRODUCER	UTF-8	This document, Section 4.5
COPRODUCER	UTF-8	This document, Section 4.5
EXECUTIVE_PRODUCER	UTF-8	This document, Section 4.5
DISTRIBUTED_BY	UTF-8	This document, Section 4.5
MASTERED_BY	UTF-8	This document, Section 4.5
ENCODED_BY	UTF-8	This document, Section 4.5

MIXED_BY	UTF-8	This document, Section 4.5
REMIXED_BY	UTF-8	This document, Section 4.5
PRODUCTION_STUDIO	UTF-8	This document, Section 4.5
THANKS_TO	UTF-8	This document, Section 4.5
PUBLISHER	UTF-8	This document, Section 4.5
LABEL	UTF-8	This document, Section 4.5
GENRE	UTF-8	This document, Section 4.6
MOOD	UTF-8	This document, Section 4.6
ORIGINAL_MEDIA_TYPE	UTF-8	This document, Section 4.6
CONTENT_TYPE	UTF-8	This document, Section 4.6
SUBJECT	UTF-8	This document, Section 4.6
DESCRIPTION	UTF-8	This document, Section 4.6
KEYWORDS	UTF-8	This document, Section 4.6
SUMMARY	UTF-8	This document, Section 4.6
SYNOPSIS	UTF-8	This document, Section 4.6
INITIAL_KEY	UTF-8	This document, Section 4.6

PERIOD	UTF-8	This document, Section 4.6
LAW_RATING	UTF-8	This document, Section 4.6
DATE_RELEASED	UTF-8	This document, Section 4.7
DATE_RECORDED	UTF-8	This document, Section 4.7
DATE_ENCODED	UTF-8	This document, Section 4.7
DATE_TAGGED	UTF-8	This document, Section 4.7
DATE_DIGITIZED	UTF-8	This document, Section 4.7
DATE_WRITTEN	UTF-8	This document, Section 4.7
DATE_PURCHASED	UTF-8	This document, Section 4.7
DATE_STARTED	UTF-8	This document, Section 4.7
DATE_ENDED	UTF-8	This document, Section 4.7
RECORDING_LOCATION	UTF-8	This document, Section 4.8
COMPOSITION_LOCATION	UTF-8	This document, Section 4.8
COMPOSER_NATIONALITY	UTF-8	This document, Section 4.8
COMMENT	UTF-8	This document, Section 4.9
PLAY_COUNTER	UTF-8	This document, Section 4.9

	RATING	UTF-8	This document, Section 4.9
	ENCODER	UTF-8	This document, Section 4.10
	ENCODER_SETTINGS	UTF-8	This document, Section 4.10
	BPS	UTF-8	This document, Section 4.10
	FPS	UTF-8	This document, Section 4.10
	BPM	UTF-8	This document, Section 4.10
	MEASURE	UTF-8	This document, Section 4.10
	TUNING	UTF-8	This document, Section 4.10
	REPLAYGAIN_GAIN	UTF-8	This document, Section 4.10
	REPLAYGAIN_PEAK	UTF-8	This document, Section 4.10
	EBU_R128_LOUDNESS	binary	This document, Section 4.10
	EBU_R128_MAX_TRUE_PEAK	binary	This document, Section 4.10
	EBU_R128_LOUDNESS_RANGE	binary	This document, Section 4.10
	EBU_R128_MAX_MOMENTARY_LOUDNESS	binary	This document, Section 4.10
	EBU_R128_MAX_SHORT_LOUDNESS	binary	This document, Section 4.10
	ISRC	UTF-8	This document, Section 4.11

	MCDI	binary	This document, Section 4.11
	ISBN	UTF-8	This document, Section 4.11
	BARCODE	UTF-8	This document, Section 4.11
	CATALOG_NUMBER	UTF-8	This document, Section 4.11
	LABEL_CODE	UTF-8	This document, Section 4.11
	LCCN	UTF-8	This document, Section 4.11
	IMDB	UTF-8	This document, Section 4.11
	TMDB	UTF-8	This document, Section 4.11
	TVDB	UTF-8	This document, Section 4.11
	TVDB2	UTF-8	This document, Section 4.11
	PURCHASE_ITEM	UTF-8	This document, Section 4.12
	PURCHASE_INFO	UTF-8	This document, Section 4.12
	PURCHASE_OWNER	UTF-8	This document, Section 4.12
	PURCHASE_PRICE	UTF-8	This document, Section 4.12
	PURCHASE_CURRENCY	UTF-8	This document, Section 4.12
	COPYRIGHT	UTF-8	This document, Section 4.13

	PRODUCTION_COPYRIGHT	UTF-8	This document, Section 4.13
	LICENSE	UTF-8	This document, Section 4.13
	TERMS_OF_USE	UTF-8	This document, Section 4.13

Table 17: Initial Contents of "Matroska Tag Names" Registry

6.2. Guidelines for the Designated Experts

Criteria that should be applied by the designated experts include determining whether the proposed registration duplicates existing entries and whether the registration description is clear and fits the purpose of this registry.

Moreover, criteria for binary tags include ensuring the data in the TagBinary element are defined in a specification. When possible, i.e., the binary format is not already in use elsewhere, it is RECOMMENDED that the data not start with the size of the data to follow, as this size is already part of the TagBinary element. The content of the binary data MUST NOT be a single UTF-8 string, in which case the tag should be TagString.

Criteria for nested tags include ensuring that the tag consists of one or more child SimpleTag elements to describe the metadata corresponding to that tag.

7. References

7.1. Normative References

[EBU-R.128]
"LOUDNESS NORMALISATION AND PERMITTED MAXIMUM LEVEL OF AUDIO SIGNALS", November 2023, <https://tech.ebu.ch/publications/r128/>.

[EBU-TECH.3341]
"LOUDNESS METERING: 'EBU MODE' METERING TO SUPPLEMENT EBU R 128 LOUDNESS NORMALIZATION", November 2023, <https://tech.ebu.ch/publications/tech3341>.

[EBU-TECH.3342]
"LOUDNESS RANGE: A MEASURE TO SUPPLEMENT EBU R 128 LOUDNESS NORMALIZATION", November 2023, <https://tech.ebu.ch/publications/tech3342>.

[GS1] "GS1 General Specifications", GS1 20.0, January 2020, <https://www.gs1.org/standards/barcodes-epcrfid-id-keys/gs1-general-specifications>.

[ID3v2.3] Nilsson, M., Mahoney, D., Ed., and J. Sundstrom, Ed., "ID3 tag version 2.3.0", 3 February 1999, <https://id3.org/id3v2.3.0>.

[ID3v2.4] Nilsson, M., "ID3 tag version 2.4.0 - Native Frames", 1 November 2000, <https://id3.org/id3v2.4.0-frames>.

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

[IMDb] Internet Movie Database, "IMDb data key concepts", <https://developer.imdb.com/documentation/key-concepts/>.

[ISBN] International ISBN Agency, "ISBN Users' Manual", December 2017, <https://www.isbn-international.org/content/isbn-users-manual/29>.

[ISO4217] International Organization for Standardization, "ISO 4217 Currency codes", ISO 4217:2015, August 2015, <https://www.iso.org/iso-4217-currency-codes.html>.

[ISRC] International ISRC Registration Authority, "International Standard Recording Code (ISRC) Handbook", IFPI 4th Edition, 2021, <https://www.ifpi.org/isrc_handbook/>.

[LCCN] United States Library Of Congress, "Library Of Congress Control Number", October 1999, <https://www.loc.gov/marc/lccn.html>.

[MovieDB] The Movie Database, "The Movie Database API", <https://developers.themoviedb.org/3/movies/get-movie-details>.

[ReplayGain] Robinson, D., "ReplayGain 1.0 specification", 10 July 2001, <https://wiki.hydrogenaud.io/index.php?title=Replay_Gain_specification>.

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

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

[RFC8794] Lhomme, S., Rice, D., and M. Bunkus, "Extensible Binary Meta Language", RFC 8794, DOI 10.17487/RFC8794, July 2020, <https://www.rfc-editor.org/info/rfc8794>.

[RFC9559] Lhomme, S., Bunkus, M., and D. Rice, "Matroska Media Container Format Specification", RFC 9559, DOI 10.17487/RFC9559, October 2024, <https://www.rfc-editor.org/info/rfc9559>.

[RFC9839] Bray, T. and P. Hoffman, "Unicode Character Repertoire Subsets", RFC 9839, DOI 10.17487/RFC9839, August 2025, <https://www.rfc-editor.org/info/rfc9839>.

[TheTVDB] The TVDB, "TVDB API V4", <https://thetvdb.github.io/v4-api/>.

7.2. Informative References

[ISO8601] International Organization for Standardization, "Date and time - Representations for information interchange - Part 1: Basic rules", ISO 8601-1:2019, February 2019, <https://www.iso.org/standard/70907.html>.

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

[RIFF.tags] Exiftool, "RIFF Tags", <https://exiftool.org/TagNames/RIFF.html>.

Authors' Addresses

Steve Lhomme
Email: slhomme@matroska.org

Moritz Bunkus
Email: moritz@bunkus.org

Dave Rice
Email: dave@dericed.com