
enzyme Documentation

Release 0.4.2

Antoine Bertin

Sep 27, 2017

Contents

1	Usage	3
2	License	5
3	API Documentation	7
3.1	MKV	7
3.2	Parsers	9
	Python Module Index	15

Release v0.4.2

Enzyme is a Python module to parse video metadata.

CHAPTER 1

Usage

Parse a MKV file:

```
>>> with open('How.I.Met.Your.Mother.S08E21.720p.HDTV.X264-DIMENSION.mkv', 'rb') as f:
...     mkv = enzyme.MKV(f)
...
>>> mkv.info
<Info [title=None, duration=0:20:56.005000, date=2013-04-15 14:06:50]>
>>> mkv.video_tracks
[<VideoTrack [1, 1280x720, V_MPEG4/ISO/AVC, name=None, language=eng]>]
>>> mkv.audio_tracks
[<AudioTrack [2, 6 channel(s), 48000Hz, A_AC3, name=None, language=und]>]
```


CHAPTER 2

License

Apache2

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

MKV

Matroska Video files use the *EBML* structure.

Track types

`enzyme.mkv.VIDEO_TRACK`
Video track type

`enzyme.mkv.AUDIO_TRACK`
Audio track type

`enzyme.mkv.SUBTITLE_TRACK`
Subtitle track type

Main interface

class `enzyme.mkv.MKV` (*stream*, *recurse_seek_head=False*)
Matroska Video file

Parameters *stream* – seekable file-like object

class `enzyme.mkv.Info` (*title=None*, *duration=None*, *date_utc=None*, *timecode_scale=None*, *muxing_app=None*, *writing_app=None*)
Object for the Info EBML element

classmethod `fromelement` (*element*)
Load the *Info* from an *Element*

Parameters *element* (*Element*) – the Info element

class enzyme.mkv.**Track** (*type=None, number=None, name=None, language=None, enabled=None, default=None, forced=None, lacing=None, codec_id=None, codec_name=None*)
Base object for the Tracks EBML element

classmethod **fromelement** (*element*)
Load the *Track* from an *Element*

Parameters *element* (*Element*) – the Track element

class enzyme.mkv.**VideoTrack** (*width=0, height=0, interlaced=False, stereo_mode=None, crop=None, display_width=None, display_height=None, display_unit=None, aspect_ratio_type=None, **kwargs*)
Object for the Tracks EBML element with *VIDEO_TRACK* TrackType

classmethod **fromelement** (*element*)
Load the *VideoTrack* from an *Element*

Parameters *element* (*Element*) – the Track element with *VIDEO_TRACK* TrackType

class enzyme.mkv.**AudioTrack** (*sampling_frequency=None, channels=None, output_sampling_frequency=None, bit_depth=None, **kwargs*)
Object for the Tracks EBML element with *AUDIO_TRACK* TrackType

classmethod **fromelement** (*element*)
Load the *AudioTrack* from an *Element*

Parameters *element* (*Element*) – the Track element with *AUDIO_TRACK* TrackType

class enzyme.mkv.**SubtitleTrack** (*type=None, number=None, name=None, language=None, enabled=None, default=None, forced=None, lacing=None, codec_id=None, codec_name=None*)
Object for the Tracks EBML element with *SUBTITLE_TRACK* TrackType

class enzyme.mkv.**Tag** (*targets=None, simletags=None*)
Object for the Tag EBML element

classmethod **fromelement** (*element*)
Load the *Tag* from an *Element*

Parameters *element* (*Element*) – the Tag element

class enzyme.mkv.**SimpleTag** (*name, language='und', default=True, string=None, binary=None*)
Object for the SimpleTag EBML element

classmethod **fromelement** (*element*)
Load the *SimpleTag* from an *Element*

Parameters *element* (*Element*) – the SimpleTag element

class enzyme.mkv.**Chapter** (*start, hidden=False, enabled=False, end=None, string=None, language=None*)
Object for the ChapterAtom and ChapterDisplay EBML element

Note: For the sake of simplicity, it is assumed that the ChapterAtom element has no more than 1 ChapterDisplay child element and informations it contains are merged into the *Chapter*

classmethod **fromelement** (*element*)
Load the *Chapter* from an *Element*

Parameters *element* (*Element*) – the ChapterAtom element

Parsers

A parser extract structured information as a tree from a container as a file-like object. It does the type conversion when explicit but does not interpret anything else. Parsers can raise a `ParserError`.

EBML

EBML (Extensible Binary Meta Language) is used by Matroska and WebM.

Element types

`enzyme.parsers.ebml.INTEGER`
Signed integer element type

`enzyme.parsers.ebml.UINTEGER`
Unsigned integer element type

`enzyme.parsers.ebml.FLOAT`
Float element type

`enzyme.parsers.ebml.STRING`
ASCII-encoded string element type

`enzyme.parsers.ebml.UNICODE`
UTF-8-encoded string element type

`enzyme.parsers.ebml.DATE`
Date element type

`enzyme.parsers.ebml.BINARY`
Binary element type

`enzyme.parsers.ebml.MASTER`
Container element type

Main interface

`enzyme.parsers.ebml.SPEC_TYPES`
Specification types to *Element types* mapping

`enzyme.parsers.ebml.READERS`
Element types to reader functions mapping. See *Readers*

You can override a reader to use one of your choice here:

```
>>> def my_binary_reader(stream, size):
...     data = stream.read(size)
...     return data
>>> READERS[BINARY] = my_binary_reader
```

`class enzyme.parsers.ebml.Element` (*id=None, type=None, name=None, level=None, position=None, size=None, data=None*)

Base object of EBML

Parameters

- **id** (*int*) – id of the element, best represented as hexadecimal (0x18538067 for Matroska Segment element)
- **type** (*INTEGER*, *UINTINTEGER*, *FLOAT*, *STRING*, *UNICODE*, *DATE*, *MASTER* or *BINARY*) – type of the element
- **name** (*string*) – name of the element
- **level** (*int*) – level of the element
- **position** (*int*) – position of element's data
- **size** (*int*) – size of element's data
- **data** – data as read by the corresponding *READERS*

class `enzyme.parsers.ebml.MasterElement` (*id=None*, *name=None*, *level=None*, *position=None*, *size=None*, *data=None*)
Element of type *MASTER* that has a list of *Element* as its data

Parameters

- **id** (*int*) – id of the element, best represented as hexadecimal (0x18538067 for Matroska Segment element)
- **name** (*string*) – name of the element
- **level** (*int*) – level of the element
- **position** (*int*) – position of element's data
- **size** (*int*) – size of element's data
- **data** (list of *Element*) – child elements

MasterElement implements some magic methods to ease manipulation. Thus, a *MasterElement* supports the *in* keyword to test for the presence of a child element by its name and gives access to it with a container getter:

```
>>> ebml_element = parse(open('test1.mkv', 'rb'), get_matroska_specs())[0]
>>> 'EBMLVersion' in ebml_element
False
>>> 'DocType' in ebml_element
True
>>> ebml_element['DocType']
Element(DocType, u'matroska')
```

load (*stream*, *specs*, *ignore_element_types=None*, *ignore_element_names=None*, *max_level=None*)
Load children *Elements* with level lower or equal to the *max_level* from the *stream* according to the *specs*

Parameters

- **stream** – file-like object from which to read
- **specs** (*dict*) – see *Specifications*
- **max_level** (*int*) – maximum level for children elements
- **ignore_element_types** (*list*) – list of element types to ignore
- **ignore_element_names** (*list*) – list of element names to ignore
- **max_level** – maximum level of elements

get (*name*, *default=None*)

Convenience method for `master_element[name].data` if *name* in `master_element` else *default*

Parameters

- **name** (*string*) – the name of the child to get
- **default** – default value if *name* is not in the `MasterElement`

Returns the data of the child `Element` or *default*

`enzyme.parsers.ebml.parse` (*stream*, *specs*, *size=None*, *ignore_element_types=None*, *ignore_element_names=None*, *max_level=None*)

Parse a stream for *size* bytes according to the *specs*

Parameters

- **stream** – file-like object from which to read
- **size** (*int* or *None*) – maximum number of bytes to read, *None* to read all the stream
- **specs** (*dict*) – see *Specifications*
- **ignore_element_types** (*list*) – list of element types to ignore
- **ignore_element_names** (*list*) – list of element names to ignore
- **max_level** (*int*) – maximum level of elements

Returns parsed data as a tree of `Element`

Return type `list`

Note: If *size* is reached in a middle of an element, reading will continue until the element is fully parsed.

`enzyme.parsers.ebml.parse_element` (*stream*, *specs*, *load_children=False*, *ignore_element_types=None*, *ignore_element_names=None*, *max_level=None*)

Extract a single `Element` from the *stream* according to the *specs*

Parameters

- **stream** – file-like object from which to read
- **specs** (*dict*) – see *Specifications*
- **load_children** (*bool*) – load children elements if the parsed element is a `MasterElement`
- **ignore_element_types** (*list*) – list of element types to ignore
- **ignore_element_names** (*list*) – list of element names to ignore
- **max_level** (*int*) – maximum level for children elements

Returns the parsed element

Return type `Element`

`enzyme.parsers.ebml.get_matroska_specs` (*webm_only=False*)

Get the Matroska specs

Parameters **webm_only** (*bool*) – load *only* WebM specs

Returns the specs in the appropriate format. See *Specifications*

Return type dict

Readers

`enzyme.parsers.ebml.readers.read_element_id(stream)`

Read the Element ID

Parameters `stream` – file-like object from which to read

Raises `ReadError` – when not all the required bytes could be read

Returns the id of the element

Return type int

`enzyme.parsers.ebml.readers.read_element_size(stream)`

Read the Element Size

Parameters `stream` – file-like object from which to read

Raises `ReadError` – when not all the required bytes could be read

Returns the size of element's data

Return type int

`enzyme.parsers.ebml.readers.read_element_integer(stream, size)`

Read the Element Data of type INTEGER

Parameters

- `stream` – file-like object from which to read
- `size (int)` – size of element's data

Raises

- `ReadError` – when not all the required bytes could be read
- `SizeError` – if size is incorrect

Returns the read integer

Return type int

`enzyme.parsers.ebml.readers.read_element_uinteger(stream, size)`

Read the Element Data of type UINTeger

Parameters

- `stream` – file-like object from which to read
- `size (int)` – size of element's data

Raises

- `ReadError` – when not all the required bytes could be read
- `SizeError` – if size is incorrect

Returns the read unsigned integer

Return type int

`enzyme.parsers.ebml.readers.read_element_float(stream, size)`

Read the Element Data of type FLOAT

Parameters

- **stream** – file-like object from which to read
- **size** (*int*) – size of element’s data

Raises

- **ReadError** – when not all the required bytes could be read
- **SizeError** – if size is incorrect

Returns the read float**Return type** float`enzyme.parsers.ebml.readers.read_element_string(stream, size)`

Read the Element Data of type STRING

Parameters

- **stream** – file-like object from which to read
- **size** (*int*) – size of element’s data

Raises

- **ReadError** – when not all the required bytes could be read
- **SizeError** – if size is incorrect

Returns the read ascii-decoded string**Return type** unicode`enzyme.parsers.ebml.readers.read_element_unicode(stream, size)`

Read the Element Data of type UNICODE

Parameters

- **stream** – file-like object from which to read
- **size** (*int*) – size of element’s data

Raises

- **ReadError** – when not all the required bytes could be read
- **SizeError** – if size is incorrect

Returns the read utf-8-decoded string**Return type** unicode`enzyme.parsers.ebml.readers.read_element_date(stream, size)`

Read the Element Data of type DATE

Parameters

- **stream** – file-like object from which to read
- **size** (*int*) – size of element’s data

Raises

- **ReadError** – when not all the required bytes could be read
- **SizeError** – if size is incorrect

Returns the read date**Return type** datetime

`enzyme.parsers.ebml.readers.read_element_binary(stream, size)`

Read the Element Data of type BINARY

Parameters

- **stream** – file-like object from which to read
- **size** (*int*) – size of element's data

Raises

- **ReadError** – when not all the required bytes could be read
- **SizeError** – if size is incorrect

Returns raw binary data

Return type bytes

Specifications

The XML specification for Matroska can be found [here](#). It is included with enzyme and can be converted to the appropriate format with `get_matroska_specs()`.

The appropriate format of the *specs* parameter for `parse()`, `parse_element()` and `load()` is `{id: (type, name, level)}`

e

`enzyme.mkv`, [7](#)

`enzyme.parsers.ebml`, [9](#)

`enzyme.parsers.ebml.readers`, [12](#)

A

AUDIO_TRACK (in module enzyme.mkv), 7
 AudioTrack (class in enzyme.mkv), 8

B

BINARY (in module enzyme.parsers.ebml), 9

C

Chapter (class in enzyme.mkv), 8

D

DATE (in module enzyme.parsers.ebml), 9

E

Element (class in enzyme.parsers.ebml), 9
 enzyme.mkv (module), 7
 enzyme.parsers.ebml (module), 9
 enzyme.parsers.ebml.readers (module), 12

F

FLOAT (in module enzyme.parsers.ebml), 9
 fromelement() (enzyme.mkv.AudioTrack class method), 8
 fromelement() (enzyme.mkv.Chapter class method), 8
 fromelement() (enzyme.mkv.Info class method), 7
 fromelement() (enzyme.mkv.SimpleTag class method), 8
 fromelement() (enzyme.mkv.Tag class method), 8
 fromelement() (enzyme.mkv.Track class method), 8
 fromelement() (enzyme.mkv.VideoTrack class method), 8

G

get() (enzyme.parsers.ebml.MasterElement method), 10
 get_matroska_specs() (in module enzyme.parsers.ebml), 11

I

Info (class in enzyme.mkv), 7
 INTEGER (in module enzyme.parsers.ebml), 9

L

load() (enzyme.parsers.ebml.MasterElement method), 10

M

MASTER (in module enzyme.parsers.ebml), 9
 MasterElement (class in enzyme.parsers.ebml), 10
 MKV (class in enzyme.mkv), 7

P

parse() (in module enzyme.parsers.ebml), 11
 parse_element() (in module enzyme.parsers.ebml), 11

R

read_element_binary() (in module enzyme.parsers.ebml.readers), 13
 read_element_date() (in module enzyme.parsers.ebml.readers), 13
 read_element_float() (in module enzyme.parsers.ebml.readers), 12
 read_element_id() (in module enzyme.parsers.ebml.readers), 12
 read_element_integer() (in module enzyme.parsers.ebml.readers), 12
 read_element_size() (in module enzyme.parsers.ebml.readers), 12
 read_element_string() (in module enzyme.parsers.ebml.readers), 13
 read_element_uinteger() (in module enzyme.parsers.ebml.readers), 12
 read_element_unicode() (in module enzyme.parsers.ebml.readers), 13
 READERS (in module enzyme.parsers.ebml), 9

S

SimpleTag (class in enzyme.mkv), 8
 SPEC_TYPES (in module enzyme.parsers.ebml), 9
 STRING (in module enzyme.parsers.ebml), 9
 SUBTITLE_TRACK (in module enzyme.mkv), 7
 SubtitleTrack (class in enzyme.mkv), 8

T

Tag (class in enzyme.mkv), [8](#)

Track (class in enzyme.mkv), [7](#)

U

UINTeger (in module enzyme.parsers.ebml), [9](#)

UNICODE (in module enzyme.parsers.ebml), [9](#)

V

VIDEO_TRACK (in module enzyme.mkv), [7](#)

VideoTrack (class in enzyme.mkv), [8](#)