pymkv

Release 1.0.6

Sheldon Woodward

Oct 17, 2020

CONTENTS

Welcome to the pymkv documentation! Here you will find links to the core modules and examples of how to use each.

CHAPTER

ONE

MODULES

The three primary modules of pymkv are *MKVFile*, *MKVTrack*, and *MKVAttachment*. The *MKVFile* class is used to import existing or create new MKV files. The *MKVTrack* class is used to add individual tracks to an *MKVFile*. The *MKVAttachment* class is used to add attachments to an *MKVFile*.

Each module supports or mimics many of the same operations as mkvmerge but are not necessarily complete. If there is functionality that is missing or an error in the docs, please open a new issue here.

1.1 MKVFile

MKVFile is the core class of pymkv. It is used to import, create, modify, and mux MKV files.

Examples

Below are some basic examples of how the *MKVFile* objects can be used.

Create and mux a new MKV. This example takes an standalone video and audio track and combines them into an MKV file.

```
>>> from pymkv import MKVFile
>>> mkv = MKVFile()
>>> mkv.add_track('/path/to/track.h264')
>>> mkv.add_track(MKVTrack('/path/to/another/track.aac'))
>>> mkv.mux('/path/to/output.mkv')
```

Generate the mkvmerge command to mux an MKV. This is example is identical to the first example except the command is only generated, not executed.

```
>>> mkv = MKVFile()
>>> mkv.add_track('/path/to/track.h264')
>>> mkv.add_track(MKVTrack('/path/to/another/track.aac'))
>>> mkv.command('/path/to/output.mkv')
```

Import an existing MKV and remove a track. This example will import an MKV that already exists on your filesystem, remove a track and allow you to mux that change into a new file.

```
>>> mkv = MKVFile('/path/to/file.mkv')
>>> mkv.remove_track(0)
>>> mkv.mux('/path/to/output.mkv')
```

Combine two MKVs. This example takes two existing MKVs and combines their tracks into a single MKV file.

```
>>> mkv1 = MKVFile('/path/to/file1.mkv')
>>> mkv2 = MKVFile('/path/to/file2.mkv')
>>> mkv1.add_file(mkv2)
>>> mkv1.mux('/path/to/output.mkv')
```

class pymkv.**MKVFile**(*file_path=None*, *title=None*)

A class that represents an MKV file.

The *MKVFile* class can either import a pre-existing MKV file or create a new one. After an *MKVFile* object has been instantiated, *MKVTrack* objects or other *MKVFile* objects can be added using *add_track()* and *add_file()* respectively.

Tracks are always added in the same order that they exist in a file or are added in. They can be reordered using *move_track_front()*, *move_track_end()*, *move_track_forward()*, *move_track_backward()*, or *swap_tracks()*.

After an *MKVFile* has been created, an mkvmerge command can be generated using *command()* or the file can be muxed using *mux()*.

Parameters

- **file_path** (*str, optional*) – Path to a pre-existing MKV file. The file will be imported into the new *MKVFile* object.
- **title** (*str, optional*) – The internal title given to the *MKVFile*. If *title* is not specified, the title of the pre-existing file will be used if it exists.

Raises **FileNotFoundError** – Raised if the path to mkvmerge could not be verified.

add_attachment(*attachment*)

Add an attachment to the *MKVFile*.

Parameters **attachment** (str, *MKVAttachment*) – The attachment to be added to the *MKVFile* object.

Raises **TypeError** – Raised if if *attachment* is not a string-like path to an attachment file or an *MKVAttachment*.

add_file(*file*)

Add an MKV file into the *MKVFile* object.

Parameters **file** (str, *MKVFile*) – The file to be combined with the *MKVFile* object.

Raises **TypeError** – Raised if if *file* is not a string-like path to an MKV file or an *MKVFile* object.

add_track(*track*)

Add a track to the *MKVFile*.

Parameters **track** (str, *MKVTrack*) – The track to be added to the *MKVFile* object.

Raises **TypeError** – Raised if *track* is not a string-like path to a track file or an *MKVTrack*.

property chapter_language

The language code of the chapters in the *MKVFile* object.

Raises **ValueError** – Raised if not a valid ISO 639-2 language code.

Type str

chapters(*file_path*, *language=None*)

Add a chapters file to the *MKVFile* object.

Parameters

- **file_path** (*str*) – The chapters file to be added to the *MKVFile* object.
- **language** (*str, optional*) – Must be an ISO639-2 language code. Only applied if no existing language information exists in chapters.

Raises

- **FileNotFoundError** – Raised if the file at *file_path* does not exist.
- **TypeError** – Raised if *file_path* is not of type str.

command(*output_path*, *subprocess=False*)

Generates an mkvmerge command based on the configured *MKVFile*.

Parameters

- **output_path** (*str*) – The path to be used as the output file in the mkvmerge command.
- **subprocess** (*bool*) – Will return the command as a list so it can be used easily with the `subprocess` module.

Returns The full command to mux the *MKVFile* as a string containing spaces. Will be returned as a list of strings with no spaces if *subprocess* is True.

Return type str, list of str

static flatten(*item*)

Flatten a list or a tuple.

Takes a list or a tuple that contains other lists or tuples and flattens into a non-nested list.

Examples

```
>>> tup = ((1, 2), (3, (4, 5)))
>>> print(MKVFile.flatten(tup))
[1, 2, 3, 4, 5]
```

Parameters **item** (*list, tuple*) – A list or a tuple object with nested lists or tuples to be flattened.

Returns A flattened version of *item*.

Return type list

get_track(*track_num=None*)

Get a *MKVTrack* from the *MKVFile* object.

Parameters **track_num** (*int, optional*) – Index of track to retrieve. Will return list of *MKVTrack* objects if argument is not provided.

Returns A list of all *MKVTrack* objects in an *MKVFile*. Returns a specific *MKVTrack* if *track_num* is specified.

Return type *MKVTrack*, list of *MKVTrack*

global_tags(*file_path*)

Add global tags to the *MKVFile* object.

Parameters **file_path** (*str*) – The tags file to be added to the *MKVFile* object.

Raises

- **FileNotFoundError** – Raised if the file at *file_path* does not exist.

- **TypeError** – Raised if *file_path* is not of type str.

link_to_next(*file_path*)
Link the output file as the successor of the *file_path* file.

Parameters **file_path** (*str*) – Path of the file to be linked to.

Raises

- **TypeError** – Raised if *file_path* is not of type str.
- **ValueError** – Raised if file at *file_path* cannot be verified as an MKV.

link_to_none()
Remove all linking to previous and next options.

link_to_previous(*file_path*)
Link the output file as the predecessor of the *file_path* file.

Parameters **file_path** (*str*) – Path of the file to be linked to.

Raises

- **TypeError** – Raised if *file_path* is not of type str.
- **ValueError** – Raised if file at *file_path* cannot be verified as an MKV.

move_track_backward(*track_num*)
Move a track backward in the *MKVFile* object.

Parameters **track_num** (*int*) – The track number of the track to move backward.

Raises **IndexError** – Raised if *track_num* is is out of range of the track list.

move_track_end(*track_num*)
Set as track as the last in the *MKVFile* object.

Parameters **track_num** (*int*) – The track number of the track to move to the back.

Raises **IndexError** – Raised if *track_num* is is out of range of the track list.

move_track_forward(*track_num*)
Move a track forward in the *MKVFile* object.

Parameters **track_num** (*int*) – The track number of the track to move forward.

Raises **IndexError** – Raised if *track_num* is is out of range of the track list.

move_track_front(*track_num*)
Set a track as the first in the *MKVFile* object.

Parameters **track_num** (*int*) – The track number of the track to move to the front.

Raises **IndexError** – Raised if *track_num* is is out of range of the track list.

mux(*output_path*, *silent=False*)
Muxes the specified *MKVFile*.

Parameters

- **output_path** (*str*) – The path to be used as the output file in the mkvmerge command.
- **silent** (*bool, optional*) – By default the mkvmerge output will be shown unless silent is True.

Raises **FileNotFoundError** – Raised if the path to mkvmerge could not be verified.

no_attachments()
Ignore the existing attachments of the *MKVFile* object.

no_chapters()
Ignore the existing chapters of the *MKVFile* object.

no_global_tags()
Ignore the existing global tags of the *MKVFile* object.

no_track_tags()
Ignore the existing track tags of the *MKVFile* object.

remove_track(*track_num*)
Remove a track from the *MKVFile* object.

Parameters **track_num** (*int*) – The track number of the track to remove.

Raises **IndexError** – Raised if *track_num* is is out of range of the track list.

replace_track(*track_num*, *track*)
Replace a track with another track in the *MKVFile* object.

Parameters

- **track_num** (*int*) – The track number of the track to replace.
- **track** (*MKVTrack*) – The *MKVTrack* to be replaced into the file.

Raises **IndexError** – Raised if *track_num* is is out of range of the track list.

split_chapters(**chapters*, *link=False*)
Split the output file into parts by chapters.

Parameters

- ***chapters** (*int, list, tuple*) – The chapters to split the file by. Can be passed as any combination of ints, inside or outside an `Iterable` object. Any lists will be flattened. Chapters must be ints.
- **link** (*bool, optional*) – Determines if the split files should be linked together after splitting.

Raises

- **TypeError** – Raised if any chapters in **chapters* are not of type int.
- **ValueError** – Raised if **chapters* contains improperly formatted chapters.

split_duration(*duration*, *link=False*)
Split the output file into parts by duration.

Parameters

- **duration** (*str, int*) – The duration of each split file. Takes either a str formatted to HH:MM:SS.nnnnnnnnn or an integer representing the number of seconds. The duration string requires formatting of at least M:S.
- **link** (*bool, optional*) – Determines if the split files should be linked together after splitting.

split_frames(**frames*, *link=False*)
Split the output file into parts by frames.

Parameters

- ***frames** (*int, list, tuple*) – The frames to split the file by. Can be passed as any combination of ints, inside or outside an `Iterable` object. Any lists will be flattened. Frames must be ints.
- **link** (*bool, optional*) – Determines if the split files should be linked together after splitting.

Raises

- **TypeError** – Raised if non-int frames are passed in for **frames* or within the **frames* iterable.
- **ValueError** – Raised if improperly formatted frames are passed in for **frames*.

split_none()
Remove all splitting options.

split_parts_frames(*frame_parts*, *link=False*)
Split the output in parts by frames.

Parameters

- **frame_parts** (*list, tuple*) – An Iterable of frame sets. Each frame set should be an `Iterable` object of an even number of frames or any number of frame pairs. The very first and last frames are permitted to be None. Frame sets containing four or more frames will output as one file containing the parts specified.
- **link** (*bool, optional*) – Determines if the split files should be linked together after splitting.

Raises

- **TypeError** – Raised if any of the frame sets are not a list or tuple.
- **ValueError** – Raised if *frame_parts* contains improperly formatted parts.

split_size(*size*, *link=False*)
Split the output file into parts by size.

Parameters

- **size** (`bitmath`, int) – The size of each split file. Takes either a `bitmath` size object or an integer representing the number of bytes.
- **link** (*bool, optional*) – Determines if the split files should be linked together after splitting.

Raises **TypeError** – Raised if if *size* is not a bitmath object or an integer.

split_timestamp_parts(*timestamp_parts*, *link=False*)
Split the output in parts by time parts.

Parameters

- **timestamp_parts** (*list, tuple*) – An Iterable of timestamp sets. Each timestamp set should be an Iterable of an even number of timestamps or any number of timestamp pairs. The very first and last timestamps are permitted to be None. Timestamp sets containing 4 or more timestamps will output as one file containing the parts specified.
- **link** (*bool, optional*) – Determines if the split files should be linked together after splitting.

Raises

- **TypeError** – Raised if any of the timestamp sets are not a list or tuple.

- **ValueError** – Raised if *timestamp_parts* contains improperly formatted parts.

split_timestamps(**timestamps*, *link=False*)
Split the output file into parts by timestamps.

Parameters

- ***timestamps** (*str, int, list, tuple*) – The timestamps to split the file by. Can be passed as any combination of strs and ints, inside or outside an `Iterable` object. Any lists will be flattened. Timestamps must be ints, representing seconds, or strs in the form HH:MM:SS.nnnnnnnnn. The timestamp string requires formatting of at least M:S.
- **link** (*bool, optional*) – Determines if the split files should be linked together after splitting.

Raises **ValueError** – Raised if invalid or improperly formatted timestamps are passed in for **timestamps*.

swap_tracks(*track_num_1*, *track_num_2*)
Swap the position of two tracks in the *MKVFile* object.

Parameters

- **track_num_1** (*int*) – The track number of one track to swap.
- **track_num_2** (*int*) – The track number of the other track to swap

Raises **IndexError** – Raised if *track_num_1* or *track_num_2* are out of range of the track list.

track_tags(**track_ids*, *exclusive=False*)
Include or exclude tags from specific tracks.

Parameters

- ***track_ids** (*int, list, tuple*) – Track ids to have tags included or excluded from.
- **exclusive** (*bool, optional*) – Determines if the *track_ids* should have their tags kept or removed. *exclusive* is False by default and will remove tags from unspecified tracks.

Raises

- **IndexError** – Raised if any ids from **track_ids* is is out of range of the track list.
- **TypeError** – Raised if an ids from **track_ids* are not of type int.
- **ValueError** – Raised if **track_ids* are improperly formatted.

1.2 MKVTrack

MKVTrack classes are used to represent tracks within an MKV or to be used in an MKV. They can represent a video, audio, or subtitle track.

Examples

Below are some basic examples of how the *MKVTrack* objects can be used.

Create a new *MKVTrack* from a track file. This example takes a standalone track file and uses it in an *MKVTrack*.

```
>>> from pymkv import MKVTrack
>>> track1 = MKVTrack('path/to/track.h264')
>>> track1.track_name = 'Some Name'
>>> track1.language = 'eng'
```

Create a new *MKVTrack* from an MKV file. This example will take a specific track from an MKV and also prevent any global tags from being included if the *MKVTrack* is muxed into an *MKVFile*.

```
>>> track2 = MKVTrack('path/to/track.aac')
>>> track2.language = 'eng'
```

Create a new *MKVTrack* from an MKV file. This example will take a specific track from an MKV and also prevent any global tags from being included if the *MKVTrack* is muxed into an *MKVFile*.

```
>>> track3 = MKVTrack('path/to/MKV.mkv', track_id=1)
>>> track3.no_global_tags = True
```

Now all these tracks can be added to an *MKVFile* object and muxed together.

```
>>> from pymkv import MKVFile
>>> file = MKVFile()
>>> file.add_track(track1)
>>> file.add_track(track2)
>>> file.add_track(track3)
>>> file.mux('path/to/output.mkv')
```

class pymkv.**MKVTrack**(*file_path*, *track_id=0*, *track_name=None*, *language=None*, *default_track=False*, *forced_track=False*)

A class that represents a track for an *MKVFile* object.

MKVTrack objects are video, audio, or subtitles. Tracks can be standalone files or a single track within an MKV file, both can be handled by pymkv. An *MKVTrack* object can be added to an *MKVFile* and will be included when the MKV is muxed.

Parameters

- **file_path** (*str*) – Path to the track file. This can also be an MKV where the *track_id* is the track represented in the MKV.
- **track_id** (*int, optional*) – The id of the track to be used from the file. *track_id* only needs to be set when importing a track from an MKV. In this case, you can specify *track_id* to indicate which track from the MKV should be used. If not set, it will import the first track. Track 0 is imported by default because mkvmerge sees standalone track files as having one track with track_id set as 0.
- **track_name** (*str, optional*) – The name that will be given to the track when muxed into a file.
- **language** (*str, optional*) – The language of the track. It must be an ISO639-2 language code.
- **default_track** (*bool, optional*) – Determines if the track should be the default track of its type when muxed into an MKV file.

- **forced_track** (*bool, optional*) – Determines if the track should be a forced track when muxed into an MKV file.

mkvmerge_path
The path where pymkv looks for the mkvmerge executable. pymkv relies on the mkvmerge executable to parse files. By default, it is assumed mkvmerge is in your shell's $PATH variable. If it is not, you need to set *mkvmerge_path* to the executable location.

Type str

track_name
The name that will be given to the track when muxed into a file.

Type str

default_track
Determines if the track should be the default track of its type when muxed into an MKV file.

Type bool

forced_track
Determines if the track should be a forced track when muxed into an MKV file.

Type bool

no_chapters
If chapters exist in the track file, don't include them when this *MKVTrack* object is a track in an *MKVFile* mux operation. This option has no effect on standalone track files, only tracks that are already part of an MKV file.

Type bool

no_global_tags
If global tags exist in the track file, don't include them when this *MKVTrack* object is a track in an *MKVFile* mux operation. This option has no effect on standalone track files, only tracks that are already part of an MKV file.

Type bool

no_track_tags
If track tags exist in the specified track within the track file, don't include them when this *MKVTrack* object is a track in an *MKVFile* mux operation. This option has no effect on standalone track files, only tracks that are already part of an MKV file.

Type bool

no_attachments
If attachments exist in the track file, don't include them when this *MKVTrack* object is a track in an *MKVFile* mux operation. This option has no effect on standalone track files, only tracks that are already part of an MKV file.

Type bool

property file_path
The path to the track or MKV file containing the desired track.

Setting this property will verify the passed in file is supported by mkvmerge and set the track_id to 0. It is recommended to recreate MKVTracks instead of setting their file path after instantiation.

Raises **ValueError** – Raised if *file_path* is not a supported file type.

Type str

property language
The language of the track.

Setting this property will verify that the passed in language is an ISO-639 language code and use it as the language for the track.

Raises **ValueError** – Raised if the passed in language is not an ISO 639-2 language code.

Type str

property tags
The tags file to include with the track.

Setting this property will check that the file path passed in exists and set it as the tags file.

Raises

- **FileNotFoundError** – Raised if the passed in file does not exist or is not a file.
- **TypeError** – Raises if the passed in file is not of type str.

Type str

property track_codec
The codec of the track such as h264 or AAC.

Type str

property track_id
The ID of the track within the file.

Setting *track_id* will check that the ID passed in exists in the file. It will then look at the new track and set the codec and track type. Should be left at 0 unless extracting a specific track from an MKV.

Raises **IndexError** – Raised if the passed in index is out of range of the file's tracks.

Type int

property track_type
The type of track such as video or audio.

Type str

1.3 MKVAttachment

MKVAttachment classes are used to represent attachment files within an MKV or to be used in an MKV.

Examples

Below are some basic examples of how the *MKVAttachment* objects can be used.

Create a new *MKVAttachment* and add it to an *MKVFile*.

```
>>> from pymkv import MKVAttachment
>>> attachment = MKVAttachment('path/to/attachment.jpg', name='NAME')
>>> attachment.description = 'DESCRIPTION'
```

Attachments can also be added directly to an *MKVFile*.

```
>>> from pymkv import MKVFile
>>> mkv = MKVFile('path/to/file.mkv')
>>> mkv.add_attachment('path/to/other/attachment.png')
```

Now, the MKV can be muxed with both attachments.

```
>>> mkv.add_attachment(attachment)
>>> mkv.mux('path/to/output.mkv')
```

class pymkv.**MKVAttachment**(*file_path*, *name=None*, *description=None*, *attach_once=False*)

A class that represents an MKV attachment for an *MKVFile* object.

Parameters

- **file_path** (*str*) – The path to the attachment file.
- **name** (*str, optional*) – The name that will be given to the attachment when muxed into a file.
- **description** (*str, optional*) – The description that will be given to the attachment when muxed into a file.
- **attach_once** (*bool, optional*) – Determines if the attachment should be added to all split files or only the first. Default is False, which will attach to all files.

mime_type

The attachment's MIME type. The type will be guessed when *file_path* is set.

Type str

name

The name that will be given to the attachment when muxed into a file.

Type str

description

The description that will be given to the attachment when muxed into a file.

Type str

attach_once

Determines if the attachment should be added to all split files or only the first. Default is False, which will attach to all files.

Type bool

property file_path

The path to the attachment file.

Raises **FileNotFoundError** – Raised if *file_path* does not exist.

Type str

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX