

---

# **vardautomation**

***Release 1.2.4***

**Ichunjo**

**Dec 12, 2022**



# CONTENTS

<b>1</b>	<b>API Reference</b>	<b>1</b>
1.1	Configuration . . . . .	1
1.1.1	Presets . . . . .	6
1.2	Tools . . . . .	7
1.2.1	Video encoders . . . . .	9
1.2.2	Audio extracters . . . . .	21
1.2.3	Audio cutters . . . . .	24
1.2.4	Audio encoders . . . . .	29
1.2.5	Muxing . . . . .	36
1.2.6	Utility . . . . .	43
1.3	Automation . . . . .	44
1.4	Chapters stuff . . . . .	47
1.5	Comparison . . . . .	56
1.6	Binary Path . . . . .	59
1.7	Language . . . . .	60
1.8	VPath . . . . .	61
1.9	Types . . . . .	64
1.10	Internal functions . . . . .	64
	<b>Python Module Index</b>	<b>67</b>
	<b>Index</b>	<b>69</b>



## API REFERENCE

Collection of classes and helper functions to automate encoding

### 1.1 Configuration

```
class vardautomation.config.FileInfo(path, /, trims_or_dfs=None, *, idx=None,
                                     preset=[Preset(idx=vapoursynth.core.lsmas.LWLibavSource,
                                     a_src=VPath('{work_filename:s}_track_{track_number:s}.wav'),
                                     a_src_cut=VPath('{work_filename:s}_cut_track_{track_number:s}.wav'),
                                     a_enc_cut=None, chapter=None, preset_type=<PresetType.VIDEO:
                                     10>), Preset(idx=vapoursynth.core.lsmas.LWLibavSource,
                                     a_src=VPath('{work_filename:s}_track_{track_number:s}.w64'),
                                     a_src_cut=VPath('{work_filename:s}_cut_track_{track_number:s}.w64'),
                                     a_enc_cut=None, chapter=None, preset_type=<PresetType.VIDEO:
                                     10>)],
                                     workdir=VPath('/home/docs/checkouts/readthedocs.org/user_builds/vardautomation/ch
```

FileInfo object. This is the first thing you should initialise.

Helper which allows to store the data related to your file to be encoded

#### Parameters

- **path** (PathLike[str] | str) – Path to your source file.
- **trims\_or\_dfs** (Union[List[Union[Trim, DuplicateFrame]], Trim, None]) – Adjust the clip length by trimming or duplicating frames. Python slicing. Defaults to None
- **idx** (Optional[Callable[[str], VideoNode]]) – Indexer used to index the video track, defaults to None
- **preset** (Union[Preset, Sequence[Preset]]) – Preset used to fill idx, a\_src, a\_src\_cut, a\_enc\_cut and chapter attributes, defaults to PresetGeneric
- **workdir** (PathLike[str] | str) – Work directory. Default to the current directorie where the script is launched.

**path:** *VPath*

Path of the video file

**path\_without\_ext:** *VPath*

Path of the video file without the extension

**work\_filename:** *str*

Work directory filename

**idx:** `Optional[Callable[[str], vapoursynth.VideoNode]]`

Vapoursynth Indexer

**preset:** `List[Preset]`

Preset(s) used

**name:** `str`

Name of the script

**workdir:** `VPath`

Work directory

**a\_src:** `Optional[VPath]`

Audio source path

**a\_src\_cut:** `Optional[VPath]`

Audio source trimmed/cut path

**a\_enc\_cut:** `Optional[VPath]`

Audio source encoded (and trimmed) path

**clip:** `vapoursynth.VideoNode`

VideoNode object loaded by the indexer

**clip\_cut:** `vapoursynth.VideoNode`

Clip trimmed

**name\_clip\_output:** `VPath`

Clip output path name

**name\_file\_final:** `VPath`

Final file output path

**set\_name\_clip\_output\_ext**(*extension*, /)

Set the extension of `FileInfo.name_clip_output`

**Parameters**

**extension** (`str`) – Extension in string format, eg. “.265”

**Return type**

`None`

**property chapter:** `Optional[VPath]`

Chapter file path

**Setter**

Set the chapter path

**Return type**

`Optional[VPath]`

**property trims\_or\_dfs:** `Optional[Union[List[Union[vardefunc.types.Trim, vardefunc.types.DuplicateFrame]], vardefunc.types.Trim]]`

Trims or DuplicateFrame objects of the current FileInfo

**Setter**

Set trims or duplicate frames

**Return type**

`Union[List[Union[Trim, DuplicateFrame]], Trim, None]`

**property media\_info:** `MediaInfo`

Get the MediaInfo of the video file loaded

**Return type**

`MediaInfo`

**property num\_prop:** `bool`

If the frame number is added to props

**Setter**

Add a prop `FileInfoFrameNumber` to the frame properties of `FileInfo.clip` and `FileInfo.clip_cut`

**Return type**

`bool`

```
class vardautomation.config.FileInfo2(path, /, trims_or_dfs=None, *, idx=None,
                                     preset=[Preset(idx=vapoursynth.core.lsmas.LWLibavSource,
                                     a_src=VPath('{work_filename:s}_track_{track_number:s}.wav'),
                                     a_src_cut=VPath('{work_filename:s}_cut_track_{track_number:s}.wav'),
                                     a_enc_cut=None, chapter=None,
                                     preset_type=<PresetType.VIDEO: 10>),
                                     Preset(idx=vapoursynth.core.lsmas.LWLibavSource,
                                     a_src=VPath('{work_filename:s}_track_{track_number:s}.w64'),
                                     a_src_cut=VPath('{work_filename:s}_cut_track_{track_number:s}.w64'),
                                     a_enc_cut=None, chapter=None,
                                     preset_type=<PresetType.VIDEO: 10>)],
                                     workdir=VPath('/home/docs/checkouts/readthedocs.org/user_builds/vardautomation/
```

Second version of FileInfo adding audio support

Helper which allows to store the data related to your file to be encoded

**Parameters**

- **path** (`PathLike[str] | str`) – Path to your source file.
- **trims\_or\_dfs** (`Union[List[Union[Trim, DuplicateFrame]], Trim, None]`) – Adjust the clip length by trimming or duplicating frames. Python slicing. Defaults to None
- **idx** (`Optional[Callable[[str], VideoNode]]`) – Indexer used to index the video track, defaults to None
- **preset** (`Union[Preset, Sequence[Preset]]`) – Preset used to fill `idx`, `a_src`, `a_src_cut`, `a_enc_cut` and `chapter` attributes, defaults to `PresetGeneric`
- **workdir** (`PathLike[str] | str`) – Work directory. Default to the current directorie where the script is launched.

**audios:** `List[vapoursynth.AudioNode]`

List of AudioNode indexed by BestAudioSource in the file

**audios\_cut:** `List[vapoursynth.AudioNode]`

List of AudioNode cut with the specified trims

**property trims\_or\_dfs:** `Optional[Union[List[Union[vardefunc.types.Trim, vardefunc.types.DuplicateFrame]], vardefunc.types.Trim]]`

Trims or DuplicateFrame objects of the current FileInfo

**Setter**

Set trims or duplicate frames

**Return type**

Union[List[Union[Trim, DuplicateFrame]], Trim, None]

**property audio:** vapoursynth.AudioNode

Return the first AudioNode track of the file.

**Return type**

AudioNode

**Returns**

AudioNode

**property audio\_cut:** vapoursynth.AudioNode

Return the first trimmed AudioNode track of the file.

**Return type**

AudioNode

**Returns**

AudioNode

**write\_a\_src**(*index*, *offset=-1*)

Using *audio\_async\_render* write the AudioNodes of the file as a WAV file to *a\_src* path

**Return type**

None

**Parameters**

- **index** (*int*) –
- **offset** (*int*) –

**write\_a\_src\_cut**(*index*, *offset=-1*)

Using *audio\_async\_render* write the AudioNodes of the file as a WAV file to *a\_src\_cut* path

**Return type**

None

**Parameters**

- **index** (*int*) –
- **offset** (*int*) –

**class** vardautomation.config.BlurayShow(*episodes*, *global\_trims=None*, \*, *idx=None*, *preset=[PresetBD, PresetBDWAV64]*, *lang=UNDEFINED*, *fps=Fraction(24000, 1001)*)

Helper class for batching shows

**Parameters**

- **episodes** (Dict[VPath, List[VPath]]) – A dictionary of episodes. Keys are the path of each bdmv folder. Values are the episodes inside the current bdmv folder key.
- **global\_trims** (Union[List[Union[Trim, DuplicateFrame]], Trim, None]) – Adjust the clips length by trimming or duplicating frames. Python slicing. Defaults to None
- **idx** (Optional[Callable[[str], VideoNode]]) – Indexer used to index the video track, defaults to None
- **preset** (Union[Preset, Sequence[Preset]]) – Preset used to fill idx, a\_src, a\_src\_cut, a\_enc\_cut and chapter attributes, defaults to PresetGeneric



- **lang** (*Lang*) – Chapters language, defaults to UNDEFINED
- **fps** (*Fraction*) –

**register\_ncops**(\**path*)

Add NCOP paths to the class

**Return type**

None

**Parameters**

**path** (*VPath*) –

**register\_nceds**(\**path*)

Add NCED paths to the class

**Return type**

None

**Parameters**

**path** (*VPath*) –

**ncops**(*file\_info\_t*)

Get all the NCOPs

**Return type**

List[TypeVar(\_FileInfoType, bound= *FileInfo*)]

**Returns**

List of FileInfo

**Parameters**

**file\_info\_t** (*Type[\_FileInfoType]*) –

**ncop**(*num*, /, *file\_info\_t*, \*, *start\_from*=1)

Get a specified NCOP

**Parameters**

- **num** (int) – Numero of the NCOP
- **start\_from** (int) – Indexing starting value, defaults to 1
- **file\_info\_t** (*Type[\_FileInfoType]*) –

**Return type**

TypeVar(\_FileInfoType, bound= *FileInfo*)

**Returns**

FileInfo object

**nceds**(*file\_info\_t*)

Get all the NCEDs

**Return type**

List[TypeVar(\_FileInfoType, bound= *FileInfo*)]

**Returns**

List of FileInfo

**Parameters**

**file\_info\_t** (*Type[\_FileInfoType]*) –

**nced**(num, /, file\_info\_t, \*, start\_from=1)

Get a specified NCED

**Parameters**

- **num** (int) – Numero of the NCED
- **start\_from** (int) – Indexing starting value, defaults to 1
- **file\_info\_t** (Type[\_FileInfoType]) –

**Return type**

TypeVar(\_FileInfoType, bound= [FileInfo](#))

**Returns**

FileInfo object

**episodes**(file\_info\_t)

Get all the episodes

**Return type**

List[TypeVar(\_FileInfoType, bound= [FileInfo](#))]

**Returns**

List of FileInfo

**Parameters**

**file\_info\_t** (Type[\_FileInfoType]) –

**episode**(num, /, file\_info\_t, \*, start\_from=1)

Get a specified episode

**Parameters**

- **num** (int) – Numero of the episode
- **start\_from** (int) – Indexing starting value, defaults to 1
- **file\_info\_t** (Type[\_FileInfoType]) –

**Return type**

TypeVar(\_FileInfoType, bound= [FileInfo](#))

**Returns**

FileInfo object

## 1.1.1 Presets

```
vardautomation.config.PresetBD = Preset(idx=vapoursynth.core.lsmas.LWLibavSource,
a_src=VPath('{work_filename:s}_track_{track_number:s}.wav'),
a_src_cut=VPath('{work_filename:s}_cut_track_{track_number:s}.wav'), a_enc_cut=None,
chapter=None, preset_type=<PresetType.VIDEO: 10>)
```

Preset for BD encode. The indexer is core.lsmas.LWLibavSource and audio sources are .wav

```
vardautomation.config.PresetWEB = Preset(idx=vapoursynth.core.ffms2.Source, a_src=None,
a_src_cut=None, a_enc_cut=VPath('.'), chapter=None, preset_type=<PresetType.VIDEO: 10>)
```

Preset for WEB encode. The indexer is core.ffms2.Source and a\_enc\_cut is blocked.

```
vardautomation.config.PresetAAC = Preset(idx=None,
a_src=VPath('{work_filename:s}_track_{track_number:s}.aac'),
a_src_cut=VPath('{work_filename:s}_cut_track_{track_number:s}.aac'),
a_enc_cut=VPath('{work_filename:s}_cut_enc_track_{track_number:s}.m4a'), chapter=None,
preset_type=<PresetType.AUDIO: 20>)
```

Preset for AAC encode.

```
vardautomation.config.PresetOpus = Preset(idx=None,
a_src=VPath('{work_filename:s}_track_{track_number:s}.opus'),
a_src_cut=VPath('{work_filename:s}_cut_track_{track_number:s}.opus'),
a_enc_cut=VPath('{work_filename:s}_cut_enc_track_{track_number:s}.opus'), chapter=None,
preset_type=<PresetType.AUDIO: 20>)
```

Preset for Opus encode.

```
vardautomation.config.PresetEAC3 = Preset(idx=None,
a_src=VPath('{work_filename:s}_track_{track_number:s}.eac3'),
a_src_cut=VPath('{work_filename:s}_cut_track_{track_number:s}.eac3'),
a_enc_cut=VPath('{work_filename:s}_cut_enc_track_{track_number:s}.eac3'), chapter=None,
preset_type=<PresetType.AUDIO: 20>)
```

Preset for EAC3 encode.

```
vardautomation.config.PresetFLAC = Preset(idx=None,
a_src=VPath('{work_filename:s}_track_{track_number:s}.flac'),
a_src_cut=VPath('{work_filename:s}_cut_track_{track_number:s}.flac'),
a_enc_cut=VPath('{work_filename:s}_cut_enc_track_{track_number:s}.flac'), chapter=None,
preset_type=<PresetType.AUDIO: 20>)
```

Preset for FLAC encode.

```
vardautomation.config.PresetChapOGM = Preset(idx=None, a_src=None, a_src_cut=None,
a_enc_cut=None, chapter=VPath('chapters/{name:s}.txt'), preset_type=<PresetType.CHAPTER:
30>)
```

Preset for OGM based chapters.

```
vardautomation.config.PresetChapXML = Preset(idx=None, a_src=None, a_src_cut=None,
a_enc_cut=None, chapter=VPath('chapters/{name:s}.xml'), preset_type=<PresetType.CHAPTER:
30>)
```

Preset for XML based chapters.

## 1.2 Tools

**class** vardautomation.tooling.abstract.**Tool**(*binary, settings, \*, check\_binary=True*)

Bases: ABC

Abstract Tool interface.

Most of the tools inherit from it.

```
# This
>>> cat settings
-o {clip_output:s} - --y4m --preset slower --crf 51

# is equivalent to this:
settings: List[str] = ['-o', '{clip_output:s}', '-', '--y4m', '--preset', 'slower',
```

(continues on next page)

(continued from previous page)

```

↪ '--crf', '51']

# and is equivalent to this:
settings: Dict[str, Any] = {
    '-o': '{clip_output:s}',
    '-': None,
    '--y4m': None,
    '--preset': 'slower',
    '--crf': 51
}

```

**Parameters**

- **binary** (PathLike[str] | str) – Path to your binary file.
- **settings** (Union[PathLike[str], str, List[str], Dict[str, Any]]) – Path to your settings file or list of string or a dict containing your settings. Special variable names can be specified and are replaced at runtime. Supported variable names are defined in [set\\_variable\(\)](#) docstring.
- **check\_binary** (bool) – Check binary's availability.

**binary:** [VPath](#)

Binary path

**params:** List[str]

Settings normalised and parsed

**abstract run()**

Tooling chain

**Return type**

Optional[NoReturn]

**abstract set\_variable()**

Set variables in the settings

**Return type**

Dict[str, Any]

**class** vardautomation.tooling.base.**BasicTool**(binary, settings, /, file=None, check\_binary=True)Bases: [Tool](#)

BasicTool interface.

Helper allowing the use of CLI programs for basic tasks like video or audio track extraction.

**Parameters**

- **binary** (PathLike[str] | str) – See [Tool.binary](#)
- **settings** (Union[PathLike[str], str, List[str], Dict[str, Any]]) – See [Tool.settings](#)
- **file** (Optional[[FileInfo](#)]) – Not used in BasicTool implementation, defaults to None
- **check\_binary** (bool) – Check binary's availability.

**file:** `Optional[FileInfo]`

FileInfo object.

**run()**

Tooling chain

**Return type**

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

**Return type**

`Dict[str, Any]`

**binary:** `VPath`

Binary path

**params:** `List[str]`

Settings normalised and parsed

### 1.2.1 Video encoders

**class** vardautomation.tooling.video.VideoEncoder(*binary, settings*)

Bases: `Tool`

General VideoEncoder interface

```
# This
>>> cat settings
-o {clip_output:s} - --y4m --preset slower --crf 51

# is equivalent to this:
settings: List[str] = ['-o', '{clip_output:s}', '-', '--y4m', '--preset', 'slower',
    ↪ '--crf', '51']

# and is equivalent to this:
settings: Dict[str, Any] = {
    '-o': '{clip_output:s}',
    '-': None,
    '--y4m': None,
    '--preset': 'slower',
    '--crf': 51
}
```

#### Parameters

- **binary** (`PathLike[str] | str`) – Path to your binary file.
- **settings** (`Union[PathLike[str], str, List[str], Dict[str, Any]]`) – Path to your settings file or list of string or a dict containing your settings. Special variable names can be specified and are replaced at runtime. Supported variable names are defined in `set_variable()` docstring.

**file:** *FileInfo*

FileInfo object

**clip:** *vapoursynth.VideoNode*

Your filtered VideoNode clip

**y4m:** *bool = True*

YUV4MPEG2 headers

More informations <http://www.vapoursynth.com/doc/pythonreference.html#VideoNode.output>

**progress\_update:** *Optional[Callable[[int, int], None]] = None*

Progress update function to be used in *vapoursynth.VideoNode.output*

**prefetch:** *int = 0*

Max number of concurrent rendered frames

**backlog:** *int = -1*

Defines how many unconsumed frames (including those that did not finish rendering yet) vapoursynth buffers at most before it stops rendering additional frames.

This argument is there to limit the memory this function uses storing frames.

**run\_enc**(*clip: vs.VideoNode, file: FileInfo*) → None

**run\_enc**(*clip: vs.VideoNode, file: None*) → None

Run encoding toolchain

**Parameters**

- **clip** (*VideoNode*) – Clip to be encoded
- **file** (*Optional[FileInfo]*) – FileInfo object

**Return type**

None

**run()**

Shouldn't be used in VideoEncoder object. Use *run\_enc()* instead

**Return type**

NoReturn

**set\_variable()**

Set variables in the settings

Replaces {clip\_output:s} by *self.file.name\_clip\_output*

Replaces {filename:s} by *self.file.name*

**Return type**

Dict[str, Any]

**binary:** *VPath*

Binary path

**params:** *List[str]*

Settings normalised and parsed

```
class vardautomation.tooling.video.VideoLanEncoder(settings, /, zones=None, override_params=None,
                                                  progress_update=progress_update_func)
```

Bases: SupportManualVFR, SupportResume, SupportQpfile, HasZone, HasOverrideParams, VideoEncoder, ABC

Abstract VideoEncoder interface for VideoLan based encoders such as x265 and x264.

```
# This
>>> cat settings
-o {clip_output:s} - --y4m --preset slower --crf 51

# is equivalent to this:
settings: List[str] = ['-o', '{clip_output:s}', '-', '--y4m', '--preset', 'slower',
↳ '--crf', '51']

# and is equivalent to this:
settings: Dict[str, Any] = {
    '-o': '{clip_output:s}',
    '-': None,
    '--y4m': None,
    '--preset': 'slower',
    '--crf': 51
}
```

#### Parameters

- **binary** – Path to your binary file.
- **settings** (Union[PathLike[str], str, List[str], Dict[str, Any]]) – Path to your settings file or list of string or a dict containing your settings Special variable names can be specified and are replaced at runtime. Supported variable names are defined in [set\\_variable\(\)](#) docstring.
- **zones** (Optional[Dict[Tuple[int, int], Dict[str, Any]]]) –
- **override\_params** (Optional[Dict[str, Any]]) –
- **progress\_update** (Optional[Callable[[int, int], None]]) –

```
override_params: Dict[str, Any] = {'--crf': 10, '--preset': 'ultrafast'}
```

#### Parameters

- **override\_params** (Optional[Dict[str, Any]]) – Parameters to be overridden in settings
- **settings** (Union[PathLike[str], str, List[str], Dict[str, Any]]) –
- **zones** (Optional[Dict[Tuple[int, int], Dict[str, Any]]]) –
- **progress\_update** (Optional[Callable[[int, int], None]]) –

**backlog:** int = -1

Defines how many unconsumed frames (including those that did not finish rendering yet) vapoursynth buffers at most before it stops rendering additional frames.

This argument is there to limit the memory this function uses storing frames.

**property params\_asdict:** Dict[str, Any]

Get *params* as a dictionary

**Return type**

Dict[str, Any]

**prefetch:** int = 0

Max number of concurrent rendered frames

**progress\_update:** Optional[UpdateFunc] = None

Progress update function to be used in *vapoursynth.VideoNode.output*

**run()**

Shouldn't be used in VideoEncoder object. Use *run\_enc()* instead

**Return type**

NoReturn

**run\_enc**(clip, file, \*, qfile\_clip=None, qfile\_func=<function make\_qfile>)

Run encoding toolchain

**Parameters**

- **clip** (Union[vapoursynth.VideoNode, Sequence[vapoursynth.VideoNode]]) – Clip to be encoded
- **file** (vardautomation.config.FileInfo / None) – FileInfo object
- **qfile\_clip** (vs.VideoNode / None) – Clip to be used to generate the Qpfile
- **qfile\_func** (Callable[[vapoursynth.VideoNode, os.PathLike[str] / str], Qpfile]) – Function to be used to generate the Qpfile

**Return type**

None

**y4m:** bool = True

YUV4MPEG2 headers

More informations <http://www.vapoursynth.com/doc/pythonreference.html#VideoNode.output>

**file:** FileInfo

FileInfo object

**clip:** vs.VideoNode

Your filtered VideoNode clip

**binary:** VPath

Binary path

**params:** List[str]

Settings normalised and parsed

**resumable:** bool = False

Enable resumable encodes

**set\_variable()**

Set variables in the settings

Replaces {clip\_output:s} with self.file.name\_clip\_output

Replaces {filename:s} with self.file.name



Replaces {frames:d} with self.clip.num\_frames  
 Replaces {fps\_num:d} with self.clip.fps.numerator  
 Replaces {fps\_den:d} with self.clip.fps.denominator  
 Replaces {bits:d} with Properties.get\_depth(self.clip)  
 Replaces {min\_keyint:d} with round(self.clip.fps)  
 Replaces {keyint:d} with round(self.clip.fps) \* 10

**Return type**

Dict[str, Any]

```
class vardautomation.tooling.video.X265(settings,/, zones=None, override_params=None,
                                       progress_update=progress_update_func)
```

Bases: *VideoLanEncoder*

Video encoder using x265 for HEVC

```
# This
>>> cat settings
-o {clip_output:s} - --y4m --preset slower --crf 51

# is equivalent to this:
settings: List[str] = ['-o', '{clip_output:s}', '-', '--y4m', '--preset', 'slower',
↳ '--crf', '51']

# and is equivalent to this:
settings: Dict[str, Any] = {
    '-o': '{clip_output:s}',
    '-': None,
    '--y4m': None,
    '--preset': 'slower',
    '--crf': 51
}
```

**Parameters**

- **binary** – Path to your binary file.
- **settings** (Union[PathLike[str], str, List[str], Dict[str, Any]]) – Path to your settings file or list of string or a dict containing your settings Special variable names can be specified and are replaced at runtime. Supported variable names are defined in *set\_variable()* docstring.
- **zones** (Optional[Dict[Tuple[int, int], Dict[str, Any]]]) –
- **override\_params** (Optional[Dict[str, Any]]) –
- **progress\_update** (Optional[Callable[[int, int], None]]) –

```
override_params: Dict[str, Any] = {'--crf': 10, '--preset': 'ultrafast'}
```

**Parameters**

- **override\_params** (Optional[Dict[str, Any]]) – Parameters to be overridden in settings

- **settings** (*Union*[*PathLike*[*str*], *str*, *List*[*str*], *Dict*[*str*, *Any*]]) –
- **zones** (*Optional*[*Dict*[*Tuple*[*int*, *int*], *Dict*[*str*, *Any*]]]) –
- **progress\_update** (*Optional*[*Callable*[[*int*, *int*], *None*]]) –

**backlog:** *int* = -1

Defines how many unconsumed frames (including those that did not finish rendering yet) vapoursynth buffers at most before it stops rendering additional frames.

This argument is there to limit the memory this function uses storing frames.

**property params\_asdict:** *Dict*[*str*, *Any*]

Get *params* as a dictionary

**Return type**

*Dict*[*str*, *Any*]

**prefetch:** *int* = 0

Max number of concurrent rendered frames

**progress\_update:** *Optional*[*UpdateFunc*] = *None*

Progress update function to be used in *vapoursynth.VideoNode.output*

**run()**

Shouldn't be used in *VideoEncoder* object. Use *run\_enc()* instead

**Return type**

*NoReturn*

**run\_enc**(*clip*, *file*, \*, *qpfile\_clip*=*None*, *qpfile\_func*=<function make\_qpfile>)

Run encoding toolchain

**Parameters**

- **clip** (*Union*[*vapoursynth.VideoNode*, *Sequence*[*vapoursynth.VideoNode*]]) – Clip to be encoded
- **file** (*vardautomation.config.FileInfo* / *None*) – *FileInfo* object
- **qpfile\_clip** (*vs.VideoNode* / *None*) – Clip to be used to generate the Qpfile
- **qpfile\_func** (*Callable*[[*vapoursynth.VideoNode*, *os.PathLike*[*str*] / *str*], *Qpfile*]) – Function to be used to generate the Qpfile

**Return type**

*None*

**y4m:** *bool* = *True*

YUV4MPEG2 headers

More informations <http://www.vapoursynth.com/doc/pythonreference.html#VideoNode.output>

**file:** *FileInfo*

*FileInfo* object

**clip:** *vs.VideoNode*

Your filtered *VideoNode* clip

**binary:** *VPath*

Binary path

**params:** List[str]

Settings normalised and parsed

**resumable:** bool = False

Enable resumable encodes

**set\_variable()**

Set variables in the settings

Replaces {clip\_output:s} with self.file.name\_clip\_output

Replaces {filename:s} with self.file.name

Replaces {frames:d} with self.clip.num\_frames

Replaces {fps\_num:d} with self.clip.fps.numerator

Replaces {fps\_den:d} with self.clip.fps.denominator

Replaces {bits:d} with Properties.get\_depth(self.clip)

Replaces {min\_keyint:d} with round(self.clip.fps)

Replaces {keyint:d} with round(self.clip.fps) \* 10

Replaces {min\_luma:d} and {max\_luma:d} with Properties.get\_colour\_range(self.params, self.clip)

Replaces {matrix:d} with Properties.get\_prop(self.clip.get\_frame(0), '\_Matrix', int)

Replaces {primaries:d} with Properties.get\_prop(self.clip.get\_frame(0), '\_Primaries', int)

Replaces {transfer:d} with Properties.get\_prop(self.clip.get\_frame(0), '\_Transfer', int)

**Return type**

Dict[str, Any]

**class** vardautomation.tooling.video.X264(settings,/, zones=None, override\_params=None, progress\_update=progress\_update\_func)

Bases: [VideoLanEncoder](#)

Video encoder using x264 for AVC

```
# This
>>> cat settings
-o {clip_output:s} - --y4m --preset slower --crf 51

# is equivalent to this:
settings: List[str] = ['-o', '{clip_output:s}', '-', '--y4m', '--preset', 'slower',
↳ '--crf', '51']

# and is equivalent to this:
settings: Dict[str, Any] = {
    '-o': '{clip_output:s}',
    '-': None,
    '--y4m': None,
    '--preset': 'slower',
    '--crf': 51
}
```

**Parameters**

- **binary** – Path to your binary file.
- **settings** (`Union[PathLike[str], str, List[str], Dict[str, Any]]`) – Path to your settings file or list of string or a dict containing your settings Special variable names can be specified and are replaced at runtime. Supported variable names are defined in `set_variable()` docstring.
- **zones** (`Optional[Dict[Tuple[int, int], Dict[str, Any]]]`) –
- **override\_params** (`Optional[Dict[str, Any]]`) –
- **progress\_update** (`Optional[Callable[[int, int], None]]`) –

`override_params: Dict[str, Any] = {'--crf': 10, '--preset': 'ultrafast'}`

**Parameters**

- **override\_params** (`Optional[Dict[str, Any]]`) – Parameters to be overridden in settings
- **settings** (`Union[PathLike[str], str, List[str], Dict[str, Any]]`) –
- **zones** (`Optional[Dict[Tuple[int, int], Dict[str, Any]]]`) –
- **progress\_update** (`Optional[Callable[[int, int], None]]`) –

**backlog:** `int = -1`

Defines how many unconsumed frames (including those that did not finish rendering yet) vapoursynth buffers at most before it stops rendering additional frames.

This argument is there to limit the memory this function uses storing frames.

**property params\_asdict:** `Dict[str, Any]`

Get `params` as a dictionary

**Return type**`Dict[str, Any]`**prefetch:** `int = 0`

Max number of concurrent rendered frames

**progress\_update:** `Optional[UpdateFunc] = None`

Progress update function to be used in `vapoursynth.VideoNode.output`

**run()**

Shouldn't be used in `VideoEncoder` object. Use `run_enc()` instead

**Return type**`NoReturn`**run\_enc**(`clip, file, *, qpfile_clip=None, qpfile_func=<function make_qpfile>`)

Run encoding toolchain

**Parameters**

- **clip** (`Union[vapoursynth.VideoNode, Sequence[vapoursynth.VideoNode]]`) – Clip to be encoded
- **file** (`vardautomation.config.FileInfo / None`) – `FileInfo` object

- **qpfile\_clip** (*vs.VideoNode* / *None*) – Clip to be used to generate the Qpfile
- **qpfile\_func** (*Callable[[vapoursynth.VideoNode, os.PathLike[str] / str], Qpfile]*) – Function to be used to generate the Qpfile

**Return type**

None

**y4m:** **bool = True**

YUV4MPEG2 headers

More informations <http://www.vapoursynth.com/doc/pythonreference.html#VideoNode.output>**file:** *FileInfo*

FileInfo object

**clip:** **vs.VideoNode**

Your filtered VideoNode clip

**binary:** *VPath*

Binary path

**params:** **List[str]**

Settings normalised and parsed

**resumable:** **bool = False**

Enable resumable encodes

**set\_variable()**

Set variables in the settings

Replaces {clip\_output:s} with self.file.name\_clip\_output

Replaces {filename:s} with self.file.name

Replaces {frames:d} with self.clip.num\_frames

Replaces {fps\_num:d} with self.clip.fps.numerator

Replaces {fps\_den:d} with self.clip.fps.denominator

Replaces {bits:d} with Properties.get\_depth(self.clip)

Replaces {min\_keyint:d} with round(self.clip.fps)

Replaces {keyint:d} with round(self.clip.fps) \* 10

Replaces {csp:s} with Properties.get\_csp(self.clip)

Replaces {matrix:s} with Properties.get\_prop(self.clip.get\_frame(0), '\_Matrix', int)

Replaces {primaries:s} with Properties.get\_prop(self.clip.get\_frame(0), '\_Primaries', int)

Replaces {transfer:s} with Properties.get\_prop(self.clip.get\_frame(0), '\_Transfer', int)

**Return type**

Dict[str, Any]

**class** vardautomation.tooling.video.**LosslessEncoder**(*binary, settings*)Bases: *VideoEncoder*

Video encoder for lossless encoding

```
# This
>>> cat settings
-o {clip_output:s} - --y4m --preset slower --crf 51

# is equivalent to this:
settings: List[str] = ['-o', '{clip_output:s}', '-', '--y4m', '--preset', 'slower',
↳ '--crf', '51']

# and is equivalent to this:
settings: Dict[str, Any] = {
    '-o': '{clip_output:s}',
    '-': None,
    '--y4m': None,
    '--preset': 'slower',
    '--crf': 51
}
```

### Parameters

- **binary** (PathLike[str] | str) – Path to your binary file.
- **settings** (Union[PathLike[str], str, List[str], Dict[str, Any]]) – Path to your settings file or list of string or a dict containing your settings Special variable names can be specified and are replaced at runtime. Supported variable names are defined in [set\\_variable\(\)](#) docstring.

**suffix\_name:** str = '\_lossless'

Suffix name for the lossless output

**set\_variable()**

Set variables in the settings

Replaces {clip\_output\_lossless:s} by self.file.name\_clip\_output.append\_stem(self.suffix\_name)

Replaces {bits:s} by Properties.get\_depth(self.clip)

### Return type

Dict[str, Any]

**backlog:** int = -1

Defines how many unconsumed frames (including those that did not finish rendering yet) vapoursynth buffers at most before it stops rendering additional frames.

This argument is there to limit the memory this function uses storing frames.

**prefetch:** int = 0

Max number of concurrent rendered frames

**progress\_update:** Optional[UpdateFunc] = None

Progress update function to be used in *vapoursynth.VideoNode.output*

**run()**

Shouldn't be used in VideoEncoder object. Use [run\\_enc\(\)](#) instead

### Return type

NoReturn

**run\_enc**(clip, file)

Run encoding toolchain

**Parameters**

- **clip** (VideoNode) – Clip to be encoded
- **file** (Optional[FileInfo]) – FileInfo object

**Return type**

None

**y4m:** bool = True

YUV4MPEG2 headers

More informations <http://www.vapoursynth.com/doc/pythonreference.html#VideoNode.output>

**file:** FileInfo

FileInfo object

**clip:** vs.VideoNode

Your filtered VideoNode clip

**binary:** VPath

Binary path

**params:** List[str]

Settings normalised and parsed

**class** vardautomation.tooling.video.NVEncClossless(\*, hevc=True)

Bases: LosslessEncoder

Built-in NvencC encoder.

Use NvencC to output a lossless encode in HEVC

**Parameters**

**hevc** (bool) – If True use HEVC codec for output. Keep in mind that 10 bit support depends on your NVenc, driver and CUDA version

**suffix\_name:** str = '\_lossless.mkv'

Suffix name for the lossless output

**progress\_update:** Optional[UpdateFunc] = None

Progress update function to be used in vapoursynth.VideoNode.output

**backlog:** int = -1

Defines how many unconsumed frames (including those that did not finish rendering yet) vapoursynth buffers at most before it stops rendering additional frames.

This argument is there to limit the memory this function uses storing frames.

**prefetch:** int = 0

Max number of concurrent rendered frames

**run**()

Shouldn't be used in VideoEncoder object. Use [run\\_enc\(\)](#) instead

**Return type**

NoReturn

**run\_enc**(clip, file)

Run encoding toolchain

**Parameters**

- **clip** (VideoNode) – Clip to be encoded
- **file** (Optional[FileInfo]) – FileInfo object

**Return type**

None

**set\_variable**()

Set variables in the settings

Replaces {clip\_output\_lossless:s} by self.file.name\_clip\_output.append\_stem(self.suffix\_name)

Replaces {bits:s} by Properties.get\_depth(self.clip)

**Return type**

Dict[str, Any]

**y4m: bool = True**

YUV4MPEG2 headers

More informations <http://www.vapoursynth.com/doc/pythonreference.html#VideoNode.output>

**file: FileInfo**

FileInfo object

**clip: vs.VideoNode**

Your filtered VideoNode clip

**binary: VPath**

Binary path

**params: List[str]**

Settings normalised and parsed

**class** vardautomation.tooling.video.FFV1(\*, threads=0)

Bases: *LosslessEncoder*

Built-in FFV1 encoder.

Use Ffmpeg to output a lossless encode in FFV1

**Parameters**

**threads** (int) – Number of threads to be used, defaults to 0 (auto selection)

**suffix\_name: str = '\_lossless.mkv'**

Suffix name for the lossless output

**progress\_update: Optional[UpdateFunc] = None**

Progress update function to be used in vapoursynth.VideoNode.output

**backlog: int = -1**

Defines how many unconsumed frames (including those that did not finish rendering yet) vapoursynth buffers at most before it stops rendering additional frames.

This argument is there to limit the memory this function uses storing frames.



**prefetch:** `int = 0`

Max number of concurrent rendered frames

**run()**

Shouldn't be used in VideoEncoder object. Use `run_enc()` instead

**Return type**

NoReturn

**run\_enc**(*clip, file*)

Run encoding toolchain

**Parameters**

- **clip** (VideoNode) – Clip to be encoded
- **file** (Optional[FileInfo]) – FileInfo object

**Return type**

None

**set\_variable()**

Set variables in the settings

Replaces {clip\_output\_lossless:s} by `self.file.name_clip_output.append_stem(self.suffix_name)`

Replaces {bits:s} by `Properties.get_depth(self.clip)`

**Return type**

Dict[str, Any]

**y4m:** `bool = True`

YUV4MPEG2 headers

More informations <http://www.vapoursynth.com/doc/pythonreference.html#VideoNode.output>

**file:** `FileInfo`

FileInfo object

**clip:** `vs.VideoNode`

Your filtered VideoNode clip

**binary:** `VPath`

Binary path

**params:** `List[str]`

Settings normalised and parsed

## 1.2.2 Audio extracters

**class** vardautomation.tooling.audio.**AudioExtractor**(*binary, settings, /, file*)

Bases: `BasicTool`

Audio base extractor interface for audio extration

**Parameters**

- **binary** (PathLike[str] | str) – See `Tool.binary`

- **settings** (Union[PathLike[str], str, List[str], Dict[str, Any]]) – See Tool.settings
- **file** (*FileInfo*) – FileInfo object, needed

**file:** *FileInfo*

FileInfo object

**track\_in:** Sequence[int]

Track number(s) of the input file

**track\_out:** Sequence[int]

Track number(s) of the output file

**run()**

Tooling chain

**Return type**

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

**Return type**

Dict[str, Any]

**binary:** *VPath*

Binary path

**params:** List[str]

Settings normalised and parsed

**class** vardautomation.tooling.audio.**MKVAudioExtractor**(*file*, /, \*, *track\_in*=-1, *track\_out*=-1, *mkvextract\_args*=None)

AudioExtractor using MKVExtract

**Parameters**

- **file** (*FileInfo*) – FileInfo object, needed
- **track\_in** (Union[int, Sequence[int]]) – Input track(s) number
- **track\_out** (Union[int, Sequence[int]]) – Output track(s) number
- **mkvextract\_args** (Optional[List[str]]) – <https://mkvtoolnix.download/doc/mkvextract.html>, defaults to None

**run()**

Tooling chain

**Return type**

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

**Return type**

Dict[str, Any]

**file:** *FileInfo*

FileInfo object

**track\_in:** *Sequence[int]*

Track number(s) of the input file

**track\_out:** *Sequence[int]*

Track number(s) of the output file

**binary:** *VPath*

Binary path

**params:** *List[str]*

Settings normalised and parsed

**class** vardautomation.tooling.audio.Eac3toAudioExtractor(*file*, /, \*, *track\_in*=-1, *track\_out*=-1, *eac3to\_args*=None)

AudioExtractor using Eac3to

#### Parameters

- **file** (*FileInfo*) – FileInfo object, needed
- **track\_in** (Union[int, Sequence[int]]) – Input track(s) number
- **track\_out** (Union[int, Sequence[int]]) – Output track(s) number
- **eac3to\_args** (Optional[List[str]]) – [https://en.wikibooks.org/wiki/Eac3to/How\\_to\\_Use](https://en.wikibooks.org/wiki/Eac3to/How_to_Use), eg. ['-log=nul']

**run()**

Tooling chain

#### Return type

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

#### Return type

Dict[str, Any]

**file:** *FileInfo*

FileInfo object

**track\_in:** *Sequence[int]*

Track number(s) of the input file

**track\_out:** *Sequence[int]*

Track number(s) of the output file

**binary:** *VPath*

Binary path

**params:** *List[str]*

Settings normalised and parsed

**class** vardautomation.tooling.audio.FFmpegAudioExtractor(*file*, /, \*, *track\_in*=-1, *track\_out*=-1)

AudioExtractor using Ffmpeg

**Parameters**

- **file** (*FileInfo*) – FileInfo object, needed
- **track\_in** (Union[int, Sequence[int]]) – Input track(s) number
- **track\_out** (Union[int, Sequence[int]]) – Output track(s) number

**run()**

Tooling chain

**Return type**

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

**Return type**

Dict[str, Any]

**file:** *FileInfo*

FileInfo object

**track\_in:** Sequence[int]

Track number(s) of the input file

**track\_out:** Sequence[int]

Track number(s) of the output file

**binary:** *VPath*

Binary path

**params:** List[str]

Settings normalised and parsed

## 1.2.3 Audio cutters

**class** vardautomation.tooling.audio.AudioCutter(*file*, /, \*, *track*, *\*\*kwargs*)

Bases: ABC

Abstract interface implementing audio trimming

**Parameters**

- **file** (*FileInfo*) – FileInfo object
- **track** (int) – Track number
- **kwargs** (Any) – Additionnal arguments

**file:** *FileInfo*

FileInfo object

**track:** int

Track number

**kwargs:** Dict[str, Any]

Additionnal arguments

**abstract run()**

Trimming toolchain

**Return type**

None

**abstract classmethod generate\_silence**(*s, output, num\_ch=2, sample\_rate=48000, bitdepth=16*)

Generate silence if supported by the current interface

**Parameters**

- **s** (float) – Seconds
- **output** (PathLike[str] | str) – Output file path
- **num\_ch** (int) – Number of channels, defaults to 2
- **sample\_rate** (int) – Sample rate in Hz, defaults to 48000
- **bitdepth** (int) – Bit depth, defaults to 16

**Return type**

Optional[NoReturn]

**class** vardautomation.tooling.audio.**ScipyCutter**(*file, /, \*, track, \*\*kwargs*)

Bases: [AudioCutter](#)

Audio cutter using scipy.io.wavfile module

**Parameters**

- **file** ([FileInfo](#)) – FileInfo object
- **track** (int) – Track number
- **kwargs** (Any) – Additionnal arguments

**file:** [FileInfo](#)

FileInfo object

**track:** int

Track number

**kwargs:** Dict[str, Any]

Additionnal arguments

**run()**

Trimming toolchain

**Return type**

None

**classmethod scipytrim**(*src, output, /, trims, ref\_clip, \*, combine=True*)

Simple trimming function that follows VapourSynth/Python slicing syntax. End frame is NOT inclusive.

**Parameters**

- **src** (PathLike[str] | str) – Input file
- **output** (PathLike[str] | str) – Output file

- **trims** (Union[Trim, DuplicateFrame, List[Trim], List[Union[Trim, DuplicateFrame]]]) – Either a list of 2-tuples, one tuple of 2 ints, a DuplicateFrame object or a list of 2-tuples and/or DuplicateFrame object.
- **ref\_clip** (VideoNode) – Vapoursynth clip used to determine framerate AND the number of frames.
- **combine** (bool) – Keep all performed trims in the same file, defaults to True

**Return type**

None

**classmethod generate\_silence**(*s, output, num\_ch=2, sample\_rate=48000, bitdepth=16*)

Generate silence if supported by the current interface

**Parameters**

- **s** (float) – Seconds
- **output** (PathLike[str] | str) – Output file path
- **num\_ch** (int) – Number of channels, defaults to 2
- **sample\_rate** (int) – Sample rate in Hz, defaults to 48000
- **bitdepth** (int) – Bit depth, defaults to 16

**Return type**

None

**class** vardautomation.tooling.audio.**EztrimCutter**(*file, /, \*, track, \*\*kwargs*)Bases: [AudioCutter](#)AudioCutter using `acsuite.eztrim()`.It fallbacks on [EztrimCutter.ezpztrim\(\)](#) if some DuplicateFrame objects are detected in the FileInfo object specified.**Parameters**

- **file** ([FileInfo](#)) – FileInfo object
- **track** (int) – Track number
- **kwargs** (Any) – Additionnal arguments

**file:** [FileInfo](#)

FileInfo object

**track:** `int`

Track number

**kwargs:** `Dict[str, Any]`

Additionnal arguments

**run()**

Trimming toolchain

**Return type**

None

**classmethod ezipztrim**(*src, output, /, trims, ref\_clip, \*, combine=True, cleanup=True*)

Simple trimming function that follows VapourSynth/Python slicing syntax. End frame is NOT inclusive.

**Parameters**

- **src** (PathLike[str] | str) – Input file
- **output** (PathLike[str] | str) – Output file
- **trims** (Union[Trim, DuplicateFrame, List[Trim], List[Union[Trim, DuplicateFrame]]]) – Either a list of 2-tuples, one tuple of 2 ints, a DuplicateFrame object or a list of 2-tuples and/or DuplicateFrame object.
- **ref\_clip** (VideoNode) – Vapoursynth clip used to determine framerate AND the number of frames.
- **combine** (bool) – Keep all performed trims in the same file, defaults to True
- **cleanup** (bool) – Delete temporary files, defaults to True

**Return type**

None

**classmethod generate\_silence**(cls, s, output, num\_ch=2, sample\_rate=48000, bitdepth=16)

Generate silence if supported by the current interface

**Parameters**

- **s** (float) – Seconds
- **output** (PathLike[str] | str) – Output file path
- **num\_ch** (int) – Number of channels, defaults to 2
- **sample\_rate** (int) – Sample rate in Hz, defaults to 48000
- **bitdepth** (int) – Bit depth, defaults to 16

**Return type**

None

**class** vardautomation.tooling.audio.**SoxCutter**(file, /, \*, track, \*\*kwargs)

Bases: [AudioCutter](#)

Audio cutter using Sox

**Parameters**

- **file** ([FileInfo](#)) – FileInfo object
- **track** (int) – Track number
- **kwargs** (Any) – Additionnal arguments

**file:** [FileInfo](#)

FileInfo object

**track:** **int**

Track number

**kwargs:** **Dict[str, Any]**

Additionnal arguments

**run()**

Trimming toolchain

**Return type**

None

**classmethod** `soxtrim(src, output, /, trims, ref_clip, *, combine=True, cleanup=True)`

Simple trimming function that follows VapourSynth/Python slicing syntax. End frame is NOT inclusive.

**Parameters**

- **src** (PathLike[str] | str) – Input file
- **output** (PathLike[str] | str) – Output file
- **trims** (Union[Trim, DuplicateFrame, List[Trim], List[Union[Trim, DuplicateFrame]]]) – Either a list of 2-tuples, one tuple of 2 ints, a DuplicateFrame object or a list of of 2-tuples and/or DuplicateFrame object.
- **ref\_clip** (VideoNode) – Vapoursynth clip used to determine framerate AND the number of frames.
- **combine** (bool) – Keep all performed trims in the same file, defaults to True
- **cleanup** (bool) – Delete temporary files, defaults to True

**Return type**

None

**classmethod** `generate_silence(s, output, num_ch=2, sample_rate=48000, bitdepth=16)`

Generate silence if supported by the current interface

**Parameters**

- **s** (float) – Seconds
- **output** (PathLike[str] | str) – Output file path
- **num\_ch** (int) – Number of channels, defaults to 2
- **sample\_rate** (int) – Sample rate in Hz, defaults to 48000
- **bitdepth** (int) – Bit depth, defaults to 16

**Return type**

None

**class** `vardautomation.tooling.audio.PassthroughCutter(file, /, *, track, **kwargs)`

Bases: [`AudioCutter`](#)

Special AudioCutter that will copy `vardautomation.config.FileInfo.a_src` to `vardautomation.config.FileInfo.a_src_cut`

**Parameters**

- **file** ([`FileInfo`](#)) – FileInfo object
- **track** (int) – Track number
- **kwargs** (Any) – Additionnal arguments

**file:** [`FileInfo`](#)

FileInfo object

**track:** `int`

Track number

**kwargs:** `Dict[str, Any]`

Additionnal arguments



**run()**

Trimming toolchain

**Return type**

None

**classmethod generate\_silence**(cls, s, output, num\_ch=2, sample\_rate=48000, bitdepth=16)

You can't generate silence from this class

**Return type**

NoReturn

**Parameters**

- **s** (float) –
- **output** (os.PathLike[str] | str) –
- **num\_ch** (int) –
- **sample\_rate** (int) –
- **bitdepth** (int) –

## 1.2.4 Audio encoders

**class** vardautomation.tooling.audio.**AudioEncoder**(binary, settings, /, file, \*, track=-1, xml\_tag=None, check\_binary=True)

Bases: *BasicTool*

BasicTool interface helper for audio encoding

**Parameters**

- **binary** (PathLike[str] | str) – See Tool.binary
  - **settings** (Union[PathLike[str], str, List[str], Dict[str, Any]]) – See Tool.settings
  - **file** (*FileInfo*) – FileInfo object, needed.
  - **track** (int) – Track number
  - **xml\_tag** (Union[PathLike[str], str, None]) – See *AudioEncoder.xml\_tag*, defaults to None
- If specified, will write a file containing the encoder info to be passed to the muxer.
- **check\_binary** (bool) – Check binary's availability.

**file:** *FileInfo*

FileInfo object

**track:** int

Track number

**xml\_tag:** Optional[Union[PathLike[str], str]]

XML tags suitable for mkvmerge

Curently only write the encoder name

More info here: <https://mkvtoolnix.download/doc/mkvmerge.html#mkvmerge.tags>

**run()**

Tooling chain

**Return type**

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

**Return type**

Dict[str, Any]

**binary:** *VPath*

Binary path

**params:** List[str]

Settings normalised and parsed

**class** vardautomation.tooling.audio.PassthroughAudioEncoder(*file*, \*, *track=-1*)

Bases: *AudioEncoder*

Special AudioEncoder that will copy FileInfo.a\_src\_cut to FileInfo.a\_enc\_cut

**Parameters**

- **file** (*FileInfo*) – FileInfo object
- **track** (int) – Track number
- **xml\_tag** – See *AudioEncoder.xml\_tag*, defaults to None

**run()**

Tooling chain

**Return type**

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

**Return type**

Dict[str, Any]

**file:** *FileInfo*

FileInfo object

**track:** int

Track number

**xml\_tag:** Optional[AnyPath]

XML tags suitable for mkvmerge

Curently only write the encoder name

More info here: <https://mkvtoolnix.download/doc/mkvmerge.html#mkvmerge.tags>

**binary:** *VPath*

Binary path

**params:** List[str]

Settings normalised and parsed

**class** vardautomation.tooling.audio.QAACEncoder(*file*, \*, *track=-1*, *mode=BitrateMode.TVBR*,  
*bitrate=127*, *xml\_tag=None*, *qaac\_args=None*)

Bases: [AudioEncoder](#)

AudioEncoder using QAAC, an open-source wrapper for Core Audio's AAC and ALAC encoder.

These following options are automatically added:

- --no-delay --no-optimize --threading

#### Parameters

- **file** ([FileInfo](#)) – FileInfo object
- **track** (int) – Track number
- **mode** (Literal[<BitrateMode.ABR>, <BitrateMode.CBR>, <BitrateMode.CVBR>, <BitrateMode.TVBR>]) – Bitrate mode. QAAC supports ABR, CBR, CVBR and TVBR, defaults to BitrateMode.TVBR
- **bitrate** (int) – Matches the bitrate for ABR, CBR and CVBR in kbit/s and quality Q for TVBR, defaults to 127
- **xml\_tag** (Union[PathLike[str], str, None]) – See [AudioEncoder.xml\\_tag](#), defaults to None
- **qaac\_args** (Optional[List[str]]) – Additional options, see <https://github.com/nu774/qaac/wiki/Command-Line-Options>, defaults to None

**run()**

Tooling chain

**Return type**

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

**Return type**

Dict[str, Any]

**file:** [FileInfo](#)

FileInfo object

**track:** int

Track number

**xml\_tag:** Optional[AnyPath]

XML tags suitable for mkvmerge

Curently only write the encoder name

More info here: <https://mkvtoolnix.download/doc/mkvmerge.html#mkvmerge.tags>

**binary:** [VPath](#)

Binary path

**params:** List[str]

Settings normalised and parsed

**class** vardautomation.tooling.audio.**OpusEncoder**(file, \*, track=-1, mode=BitrateMode.VBR, bitrate=160, xml\_tag=None, use\_ffmpeg=True, opus\_args=None)

Bases: [AudioEncoder](#)

AudioEncoder using Opus, open, royalty-free, highly versatile audio codec.

**Parameters**

- **file** ([FileInfo](#)) – FileInfo object
- **track** (int) – Track number
- **mode** (Literal[<BitrateMode.VBR>, <BitrateMode.CVBR>, <BitrateMode.CBR>]) – Bitrate mode. libopus supports VBR, CVBR, and HARD\_CBR (aka CBR), defaults to BitrateMode.VBR
- **bitrate** (int) – Target bitrate in kbit/s, defaults to 160
- **xml\_tag** (Union[PathLike[str], str, None]) – See [AudioEncoder.xml\\_tag](#), defaults to None
- **use\_ffmpeg** (bool) – Use opusenc if False, defaults to True
- **opus\_args** (Optional[List[str]]) – Additionnal arguments, defaults to None

**run()**

Tooling chain

**Return type**

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

**Return type**

Dict[str, Any]

**file:** [FileInfo](#)

FileInfo object

**track:** int

Track number

**xml\_tag:** Optional[AnyPath]

XML tags suitable for mkvmerge

Curently only write the encoder name

More info here: <https://mkvtoolnix.download/doc/mkvmerge.html#mkvmerge.tags>

**binary:** [VPath](#)

Binary path

**params:** List[str]

Settings normalised and parsed

```
class vardautomation.tooling.audio.FDKAACEncoder(file, *, track=-1, mode=BitrateMode.CBR,
                                                bitrate=256, cutoff=20000, xml_tag=None,
                                                use_ffmpeg=True, fdk_args=None)
```

Bases: [AudioEncoder](#)

AudioEncoder using fdkaac.

The libfdk-aac library is based on the Fraunhofer FDK AAC code from the Android project

#### Parameters

- **file** ([FileInfo](#)) – FileInfo object
- **track** (int) – Track number
- **mode** (Literal[<BitrateMode.CBR>, <BitrateMode.VBR>]) – Bitrate mode, fdkaac supports CBR and VBR, defaults to BitrateMode.CBR
- **bitrate** (int) – Matches the bitrate for CBR in kbit/s and quality Q for VBR, defaults to 256
- **cutoff** (int) – Set cutoff frequency. If not specified (or explicitly set to 0) it will use a value automatically computed by the library. Correspond to frequency bandwidth in Hz in fdkaac library, defaults to 20000
- **xml\_tag** (Union[PathLike[str], str, None]) – See [AudioEncoder.xml\\_tag](#), defaults to None
- **use\_ffmpeg** (bool) – Use fdkaac if False, defaults to True
- **fdk\_args** (Optional[List[str]]) – Additional options see <https://www.ffmpeg.org/ffmpeg-codecs.html#Options-11> or <https://github.com/nu774/fdkaac/blob/master/README>, defaults to None

#### run()

Tooling chain

#### Return type

None

#### set\_variable()

Set variables in the settings

No variable are replaced there.

#### Return type

Dict[str, Any]

**file:** [FileInfo](#)

FileInfo object

**track:** int

Track number

**xml\_tag:** Optional[AnyPath]

XML tags suitable for mkvmerge

Curently only write the encoder name

More info here: <https://mkvtoolnix.download/doc/mkvmerge.html#mkvmerge.tags>

**binary:** [VPath](#)

Binary path

**params:** List[str]

Settings normalised and parsed

```
class vardautomation.tooling.audio.FlacEncoder(file, *, track=-1, xml_tag=None,
                                              level=FlacCompressionLevel.VARDOU,
                                              use_ffmpeg=True, flac_args=None)
```

Bases: [AudioEncoder](#)

AudioEncoder using FLAC, Free Lossless Audio Codec

**Parameters**

- **file** ([FileInfo](#)) – FileInfo object
- **track** (int) – Track number
- **xml\_tag** (Union[PathLike[str], str, None]) – See [AudioEncoder.xml\\_tag](#), defaults to None
- **level** ([FlacCompressionLevel](#)) – See [FlacCompressionLevel](#) for all levels available, defaults to [FlacCompressionLevel.VARDOU](#)
- **use\_ffmpeg** (bool) – Will use flac if false, defaults to True
- **flac\_args** (Optional[List[str]]) – Additionnal arguments, defaults to None

**run()**

Tooling chain

**Return type**

None

**set\_variable()**

Set variables in the settings

No variable are replaced there.

**Return type**

Dict[str, Any]

**file:** [FileInfo](#)

FileInfo object

**track:** int

Track number

**xml\_tag:** Optional[AnyPath]

XML tags suitable for mkvmerge

Curently only write the encoder name

More info here: <https://mkvtoolnix.download/doc/mkvmerge.html#mkvmerge.tags>

**binary:** [VPath](#)

Binary path

**params:** List[str]

Settings normalised and parsed

```
class vardautomation.tooling.audio.BitrateMode(value)
```

Common bitrate mode enumeration

**ABR = 1**

Average BitRate

**CBR = 2**

Constant BitRate

**VBR = 3**

Variable BitRate

**CVBR = 4**

Constrained Variable BitRate

**TVBR = 5**

True Variable BitRate

**HARD\_CBR = 2**

Hard Constant BitRate

**class vardautomation.tooling.audio.FlacCompressionLevel(value)**

Flac compression level.

Keep in mind that the max FLAC can handle is 8 and ffmpeg 12

**ZERO = 0**

ffmpeg: compression\_level 0

flac: -compression-level-0

**ONE = 1**

ffmpeg: compression\_level 1

flac: -compression-level-1

**TWO = 2**

ffmpeg: compression\_level 2

flac: -compression-level-2

**THREE = 3**

ffmpeg: compression\_level 3

flac: -compression-level-3

**FOUR = 4**

ffmpeg: compression\_level 4

flac: -compression-level-4

**FIVE = 5**

ffmpeg: compression\_level 5

flac: -compression-level-5

This is the default for both ffmpeg and flac encoders

**SIX = 6**

ffmpeg: compression\_level 6

flac: -compression-level-6

**SEVEN = 7**  
ffmpeg: compression\_level 7  
flac: -compression-level-7

**EIGHT = 8**  
ffmpeg: compression\_level 8  
flac: -compression-level-8

**NINE = 9**  
ffmpeg: compression\_level 9

**TEN = 10**  
ffmpeg: compression\_level 10

**ELEVEN = 11**  
ffmpeg: compression\_level 11

**TWELVE = 12**  
ffmpeg: compression\_level 12

**FAST = 0**  
Fastest compression. Currently synonymous with 0  
ffmpeg: compression\_level 0  
flac: -compression-level-0

**BEST = 8**  
Highest compression. Currently synonymous with -8  
ffmpeg: compression\_level 0  
flac: -compression-level-0

**VARDOU = 99**  
My custom ffmpeg command:

```
['-compression_level', '12', '-lpc_type', 'cholesky', '-lpc_passes', '3', '-  
→exact_rice_parameters', '1']
```

## 1.2.5 Muxing

**class** vardautomation.tooling.mux.**MatroskaFile**(output, tracks=None, /, \*global\_opts)

Bases: `_AbstractMatroskaFile`

Matroska file interface

Register a new matroska file to be merged/splitted/appended

### Parameters

- **output** (PathLike[str] | str) – Output path
- **tracks** (Union[PathLike[str], str, [Track](#), Iterable[PathLike[str] | str | [Track](#)], None]) – A path or an iterable of path/Track
- **global\_opts** (str) – Global options



**global\_opts:** `Tuple[str, ...]`

Global options and other options that affect the whole process

**property command:** `List[str]`

Get the mkvmerge command

**Return type**

`List[str]`

**classmethod autotrack**(*file*, *lang=None*)

Automatically get the tracks from a `FileInfo` object and make a `MatroskaFile` from it

**Parameters**

- **file** (*FileInfo*) – `FileInfo` object
- **lang** (*Optional[Lang]*) –

**Return type**

*MatroskaFile*

**Returns**

`MatroskaFile` object

**static automux**(*file*)

Call `MatroskaFile.autotrack` and mux it.

**Parameters**

**file** (*FileInfo*) – `FileInfo` object

**Return type**

`None`

**property track\_lang:** `Optional[Union[List[vardautomation.language.Lang | None], Lang]]`

Lang(s) of the tracks of the current `MatroskaFile` object

**Setter**

Change the Lang of the tracks

**Return type**

`Union[List[Optional[Lang]], Lang, None]`

**mux**(*return\_workfiles: Literal[True] = True*) → `CleanupSet`

**mux**(*return\_workfiles: Literal[False]*) → `None`

Launch a merge command

**Return type**

`Optional[CleanupSet]`

**Returns**

Return workfiles if `True`

**split**(*mode*, *param*)

Split function ruled by “mode”

**Parameters**

- **mode** (*SplitMode*) – Split mode
- **param** (`str`) – Full command after the mode

**Return type**

None

**split\_size**(*size*)

Split the output file after a given size

**Parameters****size** (*str*) – d[k|m|g]**Return type**

None

**split\_duration**(*duration*)

Split the output file after a given duration

**Parameters****duration** (*str*) – HH:MM:SS.nnnnnnnnn|ds**Return type**

None

**split\_timestamps**(*timestamps*)

Split the output file after specific timestamps

**Parameters****timestamps** (*Iterable*[*str*]) – A[,B[,C...]]**Return type**

None

**split\_parts**(*parts*)

Keep specific parts by specifying timestamp ranges while discarding others

**Parameters****parts** (*List*[*Tuple*[*Optional*[*str*], *Optional*[*str*]]]) – start1-end1[,][+]start2-end2[,][+]start3-end3...]**Return type**

None

**split\_parts\_frames**(*parts*)

Keep specific parts by specifying frame/field number ranges while discarding others

**Parameters****parts** (*List*[*Tuple*[*Optional*[*int*], *Optional*[*int*]]]) – start1-end1[,][+]start2-end2[,][+]start3-end3...]**Return type**

None

**split\_frames**(*frames*)

Split after specific frames/fields

**Parameters****frames** (*Union*[*int*, *Iterable*[*int*]]) – A[,B[,C...]]**Return type**

None

**split\_chapters**(*indices*)

Split before specific chapters

**Parameters****indices** (Union[Literal['all'], Iterable[int]]) – “all” or A[,B[,C...]]**Return type**

None

**append\_to**(*files*, *ids=None*)

Enable append mode

**Parameters**

- **files** (Iterable[PathLike[str] | str]) – Files to be appended
- **ids** (Optional[Iterable[Tuple[int, int, int, int]]]) – Controls to which track another track is appended.

**Return type**

None

**add\_timestamps**(*path*, *id\_=0*)

Add timestamps global command

**Parameters**

- **path** (PathLike[str] | str) – Timecode path
- **id** – [description], defaults to 0
- **id\_** (*int*) –

**Return type**

None

**class** vardautomation.tooling.mux.**SplitMode**(*value*)

MKVMerge split modes

**SIZE** = 'size'

Split by size

**DURATION** = 'duration'

Split by duration

**TIMESTAMPS** = 'timestamps'

Split by timestamps

**PARTS** = 'parts'

Keep specific parts by specifying timestamp ranges while discarding others

**PARTS\_FRAMES** = 'parts-frames'

Keep specific parts by specifying frame/field number ranges while discarding others

**FRAMES** = 'frames'

Split by frames

**CHAPTERS** = 'chapters'

Split by chapters

**class** vardautomation.tooling.mux.**Track**(*path*, *\*opts*)Bases: `_AbstractTrack`

Standard Track interface for to be passed to mkvmerge

Register a new track

**Parameters**

- **path** (PathLike[str] | str) – Path to the file
- **opts** (str) – Additional options

**path:** *VPath*

VPath to the file

**opts:** *Tuple[str, ...]*

Additional options for this track

**class** vardautomation.tooling.mux.**MediaTrack**(*path, name=None, lang=UNDEFINED, tid=0, /, \*opts*)

Bases: *\_LanguageTrack*

Interface for medias track based to be passed to mkvmerge

Register a new track

**Parameters**

- **path** (PathLike[str] | str) – Path to the file
- **name** (Optional[str]) – Name of the track
- **lang** (*Lang* | str) – Language of the track
- **tid** (int) – Track ID
- **opts** (str) – Additional options

**lang:** *Lang*

Language of the track

**name:** *Optional[str]*

Name of the track

**count**(*value*) → integer -- return number of occurrences of value

**index**(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

**path:** *VPath*

VPath to the file

**opts:** *Tuple[str, ...]*

Additional options for this track

**class** vardautomation.tooling.mux.**VideoTrack**(*path, name=None, lang=UNDEFINED, tid=0, /, \*opts*)

Bases: *MediaTrack*

Register a new track

**Parameters**

- **path** (PathLike[str] | str) – Path to the file
- **name** (Optional[str]) – Name of the track
- **lang** (*Lang* | str) – Language of the track
- **tid** (int) – Track ID

- **opts** (str) – Additional options

**count**(value) → integer -- return number of occurrences of value

**index**(value[, start[, stop]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

**name:** Optional[str]

Name of the track

**lang:** *Lang*

Language of the track

**path:** *VPath*

VPath to the file

**opts:** Tuple[str, ...]

Additional options for this track

**class** vardautomation.tooling.mux.**AudioTrack**(path, name=None, lang=UNDEFINED, tid=0, /, \*opts)

Bases: *MediaTrack*

Register a new track

#### Parameters

- **path** (PathLike[str] | str) – Path to the file
- **name** (Optional[str]) – Name of the track
- **lang** (*Lang* | str) – Language of the track
- **tid** (int) – Track ID
- **opts** (str) – Additional options

**count**(value) → integer -- return number of occurrences of value

**index**(value[, start[, stop]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

**name:** Optional[str]

Name of the track

**lang:** *Lang*

Language of the track

**path:** *VPath*

VPath to the file

**opts:** Tuple[str, ...]

Additional options for this track

**class** vardautomation.tooling.mux.**SubtitleTrack**(path, name=None, lang=UNDEFINED, tid=0, /, \*opts)

Bases: *MediaTrack*

Register a new track

**Parameters**

- **path** (`PathLike[str] | str`) – Path to the file
- **name** (`Optional[str]`) – Name of the track
- **lang** (`Lang | str`) – Language of the track
- **tid** (`int`) – Track ID
- **opts** (`str`) – Additional options

**count**(*value*) → integer -- return number of occurrences of value

**index**(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises `ValueError` if the value is not present.

Supporting start and stop arguments is optional, but recommended.

**name:** `Optional[str]`

Name of the track

**lang:** `Lang`

Language of the track

**path:** `VPath`

VPath to the file

**opts:** `Tuple[str, ...]`

Additional options for this track

**class** vardautomation.tooling.mux.**ChaptersTrack**(*path*, *lang*=`UNDEFINED`, *charset*=`None`, */*, *\*opts*)

Bases: `_LanguageTrack`

Interface for chapters track based to be passed to mkvmerge

Register a new chapters track

**Parameters**

- **path** (`PathLike[str] | str`) – Path to the file
- **lang** (`Lang | str`) – Language of the track
- **charset** (`Optional[str]`) – Character set that is used for the conversion to UTF-8 for simple chapter files
- **opts** (`Tuple[str, ...]`) –

**charset:** `Optional[str]`

Character set that is used for the conversion to UTF-8 for simple chapter files.

**count**(*value*) → integer -- return number of occurrences of value

**index**(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises `ValueError` if the value is not present.

Supporting start and stop arguments is optional, but recommended.

**lang:** `Lang`

Language of the track

**path:** *VPath*

VPath to the file

**opts:** `Tuple[str, ...]`

Additional options for this track

## 1.2.6 Utility

**class** vardautomation.tooling.misc.**Qpfile**(*path, frames=None*)

Simple namedtuple for a qpfile

Create new instance of Qpfile(*path, frames*)

### Parameters

- **path** (*VPath*) –
- **frames** (*Optional[List[int]]*) –

**path:** *VPath*

Qpfile path

**frames:** `Optional[List[int]]`

List of keyframes

vardautomation.tooling.misc.**make\_qpfile**(*clip, path=None, /, overwrite=True, mode=SCM.WWXD | SCM.SCXVID*)

Convenience function for making a qpfile

### Parameters

- **clip** (*VideoNode*) – Source clip
- **path** (*Union[PathLike[str], str, None]*) – Path where the qpfile will be written. Default to the name of the script that run this function with the “.log” extension
- **overwrite** (*bool*) – If True, will overwrite the file
- **mode** (*Union[int, SceneChangeMode]*) – Scene change mode, defaults to SCM.WWXD\_SCXVID\_UNION

### Return type

*Qpfile*

### Returns

A Qpfile

vardautomation.tooling.misc.**get\_vs\_core**(*threads=None, max\_cache\_size=None*)

Get the VapourSynth singleton core. Optionaly, set the number of threads used and the maximum cache size

### Parameters

- **threads** (*Optional[Iterable[int]]*) – An iterable of thread numbers, defaults to None.
- **max\_cache\_size** (*Optional[int]*) – Set the upper framebuffer cache size after which memory is aggressively freed. The value is in megabytes, defaults to None.

### Return type

Core

### Returns

Vapoursynth Core.

## 1.3 Automation

**class** vardautomation.automation.**SelfRunner**(clip, file, /, config)

Self runner interface

**Parameters**

- **clip** (Union[VideoNode, Sequence[VideoNode]]) – Clip to be encoded
- **file** ([FileInfo](#)) – FileInfo object
- **config** ([RunnerConfig](#)) – Confif of the runner

**clip:** Union[vapoursynth.VideoNode, Sequence[vapoursynth.VideoNode]]

Clip to be encoded

**file:** [FileInfo](#)

FileInfo object

**config:** [RunnerConfig](#)

Config of the runner

**work\_files:** CleanupSet

Intermediate working files:

The SelfRunner class will add everything it can in this set-like, meaning if you want to delete the files you can just do:

```
runner = SelfRunner(...)
runner.run()
runner.work_files.clear()
```

**The runner will add these attributes to be deleted:**

- FileInfo.name\_clip\_output
- FileInfo.a\_src
- FileInfo.a\_src\_cut
- FileInfo.a\_enc\_cut
- FileInfo.chapter

So if you want to keep some of these files, this is possible:

```
runner.work_files.discard(self.file.name_clip_output)
runner.work_files.discard(self.file.chapter)
```

**plp\_function:** Union[\_PLPFunction, Callable[[[VPath](#)], vapoursynth.VideoNode]]

Post Lossless Processing function. Set this function if you need some adjustments on the lossless clip before running the encode. If set, it will be called with the lossless path as argument and must return a VideoNode.

**run**(\*, show\_logo=True)

Main tooling chain

**Parameters**

**show\_logo** (bool) – Print vardoto logo.



**Return type**

None

**inject\_qpfile\_params**(qpfile\_clip, qpfile\_func=make\_qpfile)**Parameters**

- **qpfile\_clip** (VideoNode) – Clip to be used to generate the Qpfile
- **qpfile\_func** (Callable[[VideoNode, PathLike[str] | str], Qpfile]) – Function to be used to generate the Qpfile

**Return type**

None

**rename\_final\_file**(name)

Rename the file.name\_file\_final

**Parameters****name** (PathLike[str] | str) – New filename**Return type**

None

**upload\_ftp**(ftp\_name, destination, rclone\_args=None)

Upload the name\_file\_final to a given FTP using rclone

**Parameters**

- **ftp\_name** (str) – FTP name
- **destination** (PathLike[str] | str) – Path destination
- **rclone\_args** (Optional[List[str]]) – Additionnal options, defaults to None

**Return type**

None

```
class vardautomation.automation.RunnerConfig(v_encoder, v_lossless_encoder=None,
                                             a_extracters=None, a_cutters=None, a_encoders=None,
                                             mkv=None, order=Order.VIDEO, clear_outputs=True)
```

Config for the SelfRunner

**Parameters**

- **v\_encoder** (VideoEncoder) –
- **v\_lossless\_encoder** (Optional[LosslessEncoder]) –
- **a\_extracters** (Optional[Union[AudioExtractor, Sequence[AudioExtractor]]]) –
- **a\_cutters** (Optional[Union[AudioCutter, Sequence[AudioCutter]]]) –
- **a\_encoders** (Optional[Union[AudioEncoder, Sequence[AudioEncoder]]]) –
- **mkv** (vardautomation.tooling.mux.MatroskaFile | None) –
- **order** (Order) –
- **clear\_outputs** (bool) –

**class Order**(value)

Simple enum for priority order

**v\_encoder:** *VideoEncoder*

Video encoder

**v\_lossless\_encoder:** *Optional[LosslessEncoder]*

Lossless video encoder

**a\_extracters:** *Optional[Union[AudioExtractor, Sequence[AudioExtractor]]]*

Audio extractor(s)

**a\_cutters:** *Optional[Union[AudioCutter, Sequence[AudioCutter]]]*

Audio cutter(s)

**a\_encoders:** *Optional[Union[AudioEncoder, Sequence[AudioEncoder]]]*

Audio encoder(s)

**mkv:** *vardautomation.tooling.mux.MatroskaFile | None*

Muxer

**order:** *Order*

Priority order

**clear\_outputs:** *bool*

Clears all clips set for output in the current environment

**class** vardautomation.automation.Patch(*encoder, clip, file, ranges, output\_filename=None, \*, debug=False*)

Easy video patching interface

#### Parameters

- **encoder** (*VideoEncoder*) – VideoEncoder to be used
- **clip** (*VideoNode*) – Clip where the patch will pick the fixed ranges
- **file** (*FileInfo*) – FileInfo object. The file that will be fixed is the file defined in *vardautomation.config.FileInfo.name\_file\_final*
- **ranges** (*Union[Range, List[Range]]*) – Ranges of frames that need to be fixed
- **output\_filename** (*Optional[str]*) – Optional filename. If not specified a suffix *\_new* will be added, defaults to *None*
- **debug** (*bool*) – Debug argument, defaults to *False*

**encoder:** *VideoEncoder*

VideoEncoder to be used

**clip:** *vapoursynth.VideoNode*

Clip where the patch will pick the fixed ranges

**file:** *FileInfo*

FileInfo object

**The file that will be fixed is the file defined in**

*vardautomation.config.FileInfo.name\_file\_final*

**ranges:** *List[Tuple[int, int]]*

Normalised ranges

**workdir:** *VPath*

Work directory path

**output\_filename:** *VPath*

Output filename path

**debug:** **bool**

Debug boolean

**run()**

Launch patch

**Return type**

None

**do\_cleanup()**

Delete working directory folder

**Return type**

None

## 1.4 Chapters stuff

**class** vardautomation.chapterisation.**Chapter**(*name, start\_frame, end\_frame,*  
*lang=<vardautomation.language.Lang object>*)

Chapter object

Create new instance of Chapter(name, start\_frame, end\_frame, lang)

**Parameters**

- **name** (*str*) –
- **start\_frame** (*int*) –
- **end\_frame** (*Optional[int]*) –
- **lang** (*Lang*) –

**name:** **str**

Name of the chapter

**start\_frame:** **int**

Start frame

**end\_frame:** **Optional[int]**

Optional end frame

**lang:** *Lang*

Language of the chapter

**class** vardautomation.chapterisation.**Chapters**(*chapter\_file*)

Bases: ABC

Abstract Chapters interface

Register a new Chapters object

**Parameters**

**chapter\_file** (*PathLike[str] | str*) – Chapters file path

**chapter\_file:** [VPath](#)

Chapters file path

**abstract create**(*chapters, fps*)

Create the current Chapters object by passing a list of Chapter

**Parameters**

- **chapters** (List[[Chapter](#)]) – List of Chapter
- **fps** (Fraction) – Framerate Per Second

**Return type**

Optional[NoReturn]

**abstract set\_names**(*names*)

Change/set names of the current Chapters object

**Parameters**

**names** (Sequence[Optional[str]]) – List of optional names. A None value won't change the name of the current chapter list

**Return type**

Optional[NoReturn]

**abstract shift\_times**(*frames, fps*)

Shift timestamps by given number of frames.

**Parameters**

- **frames** (int) – Corresponding number of frames to be shifted
- **fps** (Fraction) – Framerate Per Second

**Return type**

Optional[NoReturn]

**abstract to\_chapters**(*fps, lang*)

Convert the Chapters object to a list of chapter

**Parameters**

- **fps** (Fraction) – Framerate Per Second
- **lang** (Optional[[Lang](#)]) – Language of the chapter. If specified it will override the current language of the Chapters object

**Return type**

List[[Chapter](#)]

**copy**(*destination*)

Copy source chapter to destination and change target of [Chapters.chapter\\_file](#) to the destination one.

**Parameters**

**destination** (PathLike[str] | str) – Destination path

**Return type**

None

**create\_qpfile**(*qpfile, fps*)

Create a qp file from the current Chapters object

**Parameters**

- **qpf**file (PathLike[str] | str) – Qpfile path
- **fps** (Fraction) – Framerate Per Second

**Return type**

None

**class** vardautomation.chapterisation.**OGMChapters**(*chapter\_file*, *extension*='.txt')

Bases: [Chapters](#)

OGMChapters object.

An OGM based Chapters is a TXT file

Register a new OGMChapters object

**Parameters**

- **chapter\_file** (PathLike[str] | str) – Chapters file path
- **extension** (*str*) –

**chapter\_file:** [VPath](#)

Chapters file path

**create**(*chapters*, *fps*)

Create the current Chapters object by passing a list of Chapter

**Parameters**

- **chapters** (List[[Chapter](#)]) – List of Chapter
- **fps** (Fraction) – Framerate Per Second

**Return type**

None

**set\_names**(*names*)

Change/set names of the current Chapters object

**Parameters**

**names** (Sequence[Optional[str]]) – List of optional names. A None value won't change the name of the current chapter list

**Return type**

None

**shift\_times**(*frames*, *fps*)

Shift timestamps by given number of frames.

**Parameters**

- **frames** (int) – Corresponding number of frames to be shifted
- **fps** (Fraction) – Framerate Per Second

**Return type**

None

**to\_chapters**(*fps*, *lang*)

Convert the Chapters object to a list of chapter

**Parameters**

- **fps** (Fraction) – Framerate Per Second

- **lang** (Optional[[Lang](#)]) – Language of the chapter. If specified it will override the current language of the Chapters object

**Return type**List[[Chapter](#)]**copy**(*destination*)

Copy source chapter to destination and change target of [Chapters.chapter\\_file](#) to the destination one.

**Parameters****destination** (PathLike[str] | str) – Destination path**Return type**

None

**create\_qpfile**(*qpfile*, *fps*)

Create a qp file from the current Chapters object

**Parameters**

- **qpfile** (PathLike[str] | str) – Qpfile path
- **fps** (Fraction) – Framerate Per Second

**Return type**

None

```
class vardautomation.chapterisation.MatroskaXMLChapters(chapter_file, extension='.xml')
```

Bases: [Chapters](#)

MatroskaXMLChapters object

An MatroskaXML based Chapters is a .xml file

Register a new MatroskaXMLChapters object

**Parameters**

- **chapter\_file** (PathLike[str] | str) – Chapters file path
- **extension** (str) –

**chapter\_file:** [VPath](#)

Chapters file path

**create**(*chapters*, *fps*)

Create the current Chapters object by passing a list of Chapter

**Parameters**

- **chapters** (List[[Chapter](#)]) – List of Chapter
- **fps** (Fraction) – Framerate Per Second

**Return type**

None

**set\_names**(*names*)

Change/set names of the current Chapters object

**Parameters**

**names** (Sequence[Optional[str]]) – List of optional names. A None value won't change the name of the current chapter list

**Return type**

None

**shift\_times**(*frames*, *fps*)

Shift timestamps by given number of frames.

**Parameters**

- **frames** (int) – Corresponding number of frames to be shifted
- **fps** (Fraction) – Framerate Per Second

**Return type**

None

**to\_chapters**(*fps*, *lang=None*)

Convert the Chapters object to a list of chapter

**Parameters**

- **fps** (Fraction) – Framerate Per Second
- **lang** (Optional[[Lang](#)]) – Language of the chapter. If specified it will override the current language of the Chapters object

**Return type**List[[Chapter](#)]**copy**(*destination*)Copy source chapter to destination and change target of [Chapters.chapter\\_file](#) to the destination one.**Parameters****destination** (PathLike[str] | str) – Destination path**Return type**

None

**create\_qpfile**(*qpfile*, *fps*)

Create a qp file from the current Chapters object

**Parameters**

- **qpfile** (PathLike[str] | str) – Qpfile path
- **fps** (Fraction) – Framerate Per Second

**Return type**

None

**class** vardautomation.chapterisation.**MplsChapters**(*chapter\_file*)Bases: [Chapters](#)

MplsChapters object

Register a new Chapters object

**Parameters****chapter\_file** (PathLike[str] | str) – Chapters file path**m2ts:** [VPath](#)

Associated m2ts file name

**chapters:** List[[Chapter](#)]

Current list of chapters of this MplsChapters

**fps:** **Fraction**

Framerate Per Second

**create**(*chapter\_file*) = <function MplsChapters>

**Parameters**

**chapter\_file** (*os.PathLike[str] | str*) –

**Return type**

None

**set\_names**(*chapter\_file*) = <function MplsChapters>

**Parameters**

**chapter\_file** (*os.PathLike[str] | str*) –

**Return type**

None

**shift\_times**(*chapter\_file*) = <function MplsChapters>

**Parameters**

**chapter\_file** (*os.PathLike[str] | str*) –

**Return type**

None

**to\_chapters**(*fps=None, lang=None*)

Convert the Chapters object to a list of chapter

**Parameters**

- **fps** (Optional[Fraction]) – Framerate Per Second
- **lang** (Optional[Lang]) – Language of the chapter. If specified it will override the current language of the Chapters object

**Return type**

List[Chapter]

**copy**(*destination*)

Copy source chapter to destination and change target of *Chapters.chapter\_file* to the destination one.

**Parameters**

**destination** (*PathLike[str] | str*) – Destination path

**Return type**

None

**create\_qpfile**(*qpfile, fps*)

Create a qp file from the current Chapters object

**Parameters**

- **qpfile** (*PathLike[str] | str*) – Qpfile path
- **fps** (Fraction) – Framerate Per Second

**Return type**

None

**chapter\_file:** *VPath*

Chapters file path



**class** vardautomation.chapterisation.IfoChapters(*chapter\_file*)

Bases: [Chapters](#)

IfoChapters object

Register a new Chapters object

**Parameters**

**chapter\_file** (PathLike[str] | str) – Chapters file path

**chapters:** List[[Chapter](#)]

Current list of chapters of this IfoChapters

**fps:** Fraction

Framerate Per Second

**create**(*chapter\_file*) = <function IfoChapters>

**Parameters**

**chapter\_file** (os.PathLike[str] | str) –

**Return type**

None

**set\_names**(*chapter\_file*) = <function IfoChapters>

**Parameters**

**chapter\_file** (os.PathLike[str] | str) –

**Return type**

None

**shift\_times**(*chapter\_file*) = <function IfoChapters>

**Parameters**

**chapter\_file** (os.PathLike[str] | str) –

**Return type**

None

**to\_chapters**(*fps=None, lang=None*)

Convert the Chapters object to a list of chapter

**Parameters**

- **fps** (Optional[Fraction]) – Framerate Per Second
- **lang** (Optional[[Lang](#)]) – Language of the chapter. If specified it will override the current language of the Chapters object

**Return type**

List[[Chapter](#)]

**copy**(*destination*)

Copy source chapter to destination and change target of [Chapters.chapter\\_file](#) to the destination one.

**Parameters**

**destination** (PathLike[str] | str) – Destination path

**Return type**

None

**create\_qpfile**(*qpfile*, *fps*)

Create a qp file from the current Chapters object

**Parameters**

- **qpfile** (PathLike[str] | str) – Qpfile path
- **fps** (Fraction) – Framerate Per Second

**Return type**

None

**chapter\_file:** *VPath*

Chapters file path

**class** vardautomation.chapterisation.**MplsReader**(*bd\_folder*, *lang*=*UNDEFINED*,  
  *default\_chap\_name*='Chapter')

MPLS reader

Initialise a MplsReader.

**Parameters**

- **bd\_folder** (PathLike[str] | str) – A valid bluray folder path should contain a BDMV and CERTIFICATE folders
- **lang** (*Lang*) – Language to be set, defaults to UNDEFINED
- **default\_chap\_name** (str) – Prefix used as default name for the generated chapters, defaults to 'Chapter'

**class** **MplsFile**(*mpls\_file*, *mpls\_chapters*)

Class for MPLS file

Create new instance of MplsFile(*mpls\_file*, *mpls\_chapters*)

**Parameters**

- **mpls\_file** (*VPath*) –
- **mpls\_chapters** (*List*[*MplsChapters*]) –

**mpls\_file:** *VPath*

MPLS file path

**mpls\_chapters:** *List*[*MplsChapters*]

MPLS Chapters list

**bd\_folder:** *VPath*

Bluray Disc folder path (usually PLAYLIST)

**mpls\_folder:** *VPath*

MPLS folder path

**m2ts\_folder:** *VPath*

M2TS folder path (usually STREAMS)

**lang:** *Lang*

Language

**default\_chap\_name:** *str*

Prefix used as default name for the generated chapters

**get\_playlist()**

Returns a list of all the mpls files contained in the folder specified in the constructor.

**Return type**

List[MplsFile]

**write\_playlist(output\_folder=None, \*, chapters\_obj=MatroskaXMLChapters)**

Extract and write the playlist folder

**Parameters**

- **output\_folder** (Union[PathLike[str], str, None]) – Output path where the chapters will be written. If not specified will write in the MPLS folder, defaults to None
- **chapters\_obj** (Type[Chapters]) – Type of wanted chapters, defaults to MatroskaXMLChapters

**Return type**

None

**parse\_mpls(mpls\_file)**

Parse a mpls file and return a list of chapters that were in the MPLS file.

**Parameters**

**mpls\_file** (PathLike[str] | str) – MPL file path

**Return type**

List[MplsChapters]

**class** vardautomation.chapterisation.IfoReader(dvd\_folder, lang=UNDEFINED,  
default\_chap\_name='Chapter')

Ifo reader

Initialise a IfoReader

**Parameters**

- **dvd\_folder** (PathLike[str] | str) – A valid dvd folder path should contain at least a VIDEO\_TS folder.
- **lang** (Lang) – Language to be set, defaults to UNDEFINED
- **default\_chap\_name** (str) – Prefix used as default name for the generated chapters, defaults to 'Chapter'

**dvd\_folder:** VPath

DVD Folder path

**ifo\_folder:** VPath

Current IFO folder path

**lang:** Lang

Language

**default\_chap\_name:** str

Prefix used as default name for the generated chapters

**write\_programs(output\_folder=None, \*, chapters\_obj=MatroskaXMLChapters, ifo\_file='VTS\_01\_0.IFO')**

Extract and write the programs from the IFO file to chapters files

**Parameters**

- **output\_folder** (Union[PathLike[str], str, None]) – Output path where the chapters will be written. If not specified will write in the IFO folder, defaults to None
- **chapters\_obj** (Type[*Chapters*]) – Type of wanted chapters, defaults to MatroskaXMLChapters
- **ifo\_file** (str) – Name of the ifo file, defaults to 'VTS\_01\_0.IFO'

**Return type**

None

**parse\_ifo**(*ifo\_file*)

Parse a mpls file and return a list of chapters that were in the ifo file.

**Parameters**

**ifo\_file** (PathLike[str] | str) – IFO file path

**Return type**List[*IfoChapters*]

## 1.5 Comparison

```
class vardautomation.comp.Writer(value)
```

Writer to be used to extract frames

**FFMPEG** = 1

FFmpeg encoder

**IMWRI** = 2

vapoursynth.core.imwri Vapoursynth plugin

**OPENCV** = 3

opencv library

**PILLOW** = 4

Pillow library

**PYQT** = 5

PyQt library

**PYTHON** = 6

Pure python implementation

```
class vardautomation.comp.PictureType(value)
```

A simple enum for picture types.

**I** = b'I'

I frames

**P** = b'P'

P frames

**B** = b'B'

B frames

```
class vardautomation.comp.SlowPicsConf(collection_name='runpy', public=True, optimise=True,  
                                       nsfw=False, remove_after=None)
```

Slow.pics configuration

Create new instance of SlowPicsConf(collection\_name, public, optimise, nsfw, remove\_after)

#### Parameters

- **collection\_name** (*str*) –
- **public** (*bool*) –
- **optimise** (*bool*) –
- **nsfw** (*bool*) –
- **remove\_after** (*Optional[int]*) –

**collection\_name:** **str**

Slowpics's collection name.

Default is the name of the current script

**public:** **bool**

Make the comparison public

**optimise:** **bool**

If 'true', images will be losslessly optimised

**nsfw:** **bool**

If images not suitable for minors (nudity, gore, etc.)

**remove\_after:** **Optional[int]**

Remove after N days

```
class vardautomation.comp.Comparison(clips, path='comps', num=15, frames=None, picture_type=None)
```

Extract frames, make diff between two clips and upload to slow.pics

#### Parameters

- **clips** (*Dict[str, VideoNode]*) – Named clips.
- **path** (*PathLike[str] | str*) – Path to your comparison folder, defaults to 'comps'
- **num** (*int*) – Number of frames to extract, defaults to 15
- **frames** (*Optional[Iterable[int]]*) – Additionnal frame numbers that will be added to the total of num, defaults to None
- **picture\_type** (*Union[PictureType, List[PictureType], None]*) – Select picture types to pick, default to None

```
extract(writer=Writer.PYTHON, compression=-1, force_bt709=False)
```

Extract images from the specified clips in the constructor

#### Parameters

- **writer** (*Writer*) – Writer method to be used, defaults to Writer.PYTHON
- **compression** (*int*) – Compression level. It depends of the writer used, defaults to -1 which means automatic selection
- **force\_bt709** (*bool*) – Force BT709 matrix before conversion to RGB24, defaults to False

**Return type**

None

**magick\_compare()**

Make an image of differences between the first and second clip using ImageMagick. Will raise an exception if more than 2 clips are passed to the constructor.

**Return type**

None

**upload\_to\_slowpics(*config*)**

Upload to slow.pics with given configuration

**Parameters**

**config** (*SlowPicsConf*) – NamedTuple which contains the uploading configuration

**Return type**

None

```
vardautomation.comp.make_comps(clips, path='comps', num=15, frames=None, *, picture_types=None,
                               force_bt709=False, writer=Writer.PYTHON, compression=-1,
                               magick_compare=False, slowpics_conf=None)
```

Convenience function for [Comparison](#).

**Parameters**

- **clips** (Dict[str, VideoNode]) – Named clips.
- **path** (PathLike[str] | str) – Path to your comparison folder, defaults to 'comps'
- **num** (int) – Number of frames to extract, defaults to 15
- **frames** (Optional[Iterable[int]]) – Additionnal frame numbers that will be added to the total of num, defaults to None
- **picture\_types** (Union[PictureType, List[PictureType], None]) – Select picture types to pick, default to None
- **force\_bt709** (bool) – Force BT709 matrix before conversion to RGB24, defaults to False
- **writer** (*Writer*) – Writer method to be used, defaults to Writer.PYTHON
- **compression** (int) – Compression level. It depends of the writer used, defaults to -1 which means automatic selection
- **magick\_compare** (bool) – Make diffs between the first and second clip. Will raise an exception if more than 2 clips are passed to clips, defaults to False
- **slowpics\_conf** (Optional[*SlowPicsConf*]) – slow.pics configuration. If specified, images will be uploaded following this configuration

**Return type**

None

## 1.6 Binary Path

**class** vardautomation.binary\_path.BinaryPath

Class storing the path of the variable binaries used in vardautomation.

Just edit one of these attributes if the binary is not in your environment path

**Return type**

*NoReturn*

**eac3to:** *VPath* = *VPath*('eac3to')

<https://www.videohelp.com/software/eac3to>

[https://en.wikibooks.org/wiki/Eac3to/How\\_to\\_Use](https://en.wikibooks.org/wiki/Eac3to/How_to_Use)

**fdkaac:** *VPath* = *VPath*('fdkaac')

<https://github.com/nu774/fdkaac>

[https://en.wikipedia.org/wiki/Fraunhofer\\_FDK\\_AAC](https://en.wikipedia.org/wiki/Fraunhofer_FDK_AAC)

Also available in ffmpeg with --enable-libfdk-aac

**ffmpeg:** *VPath* = *VPath*('ffmpeg')

<https://www.ffmpeg.org/>

**ffmsindex:** *VPath* = *VPath*('ffmsindex')

<https://github.com/FFMS/ffms2>

**flac:** *VPath* = *VPath*('flac')

<https://xiph.org/flac/index.html>

**mkvextract:** *VPath* = *VPath*('mkvextract')

<https://mkvtoolnix.download/>

<https://mkvtoolnix.download/doc/mkvextract.html>

**mkvmerge:** *VPath* = *VPath*('mkvmerge')

<https://mkvtoolnix.download/>

<https://mkvtoolnix.download/doc/mkvextract.html>

**nvenc:** *VPath* = *VPath*('nvenc')

<https://github.com/rigaya/NVEnc>

**opusenc:** *VPath* = *VPath*('opusenc')

<https://github.com/xiph/opus-tools>

Also available in ffmpeg

**qaac:** *VPath* = *VPath*('qaac')

<https://sites.google.com/site/qaacpage/>

**rclone:** *VPath* = *VPath*('rclone')

<https://rclone.org/>

**sox:** *VPath* = *VPath*('sox')

<http://sox.sourceforge.net/>

**x264:** *VPath* = *VPath*('x264')

<https://www.videolan.org/developers/x264.html>

```
x265: VPath = VPath('x265')  
      http://msystem.waw.pl/x265/  
      https://bitbucket.org/multicoreware/x265\_git/wiki/Home
```

## 1.7 Language

```
class vardautomation.language.Lang(language, *, iso639_variant='B')
```

Basic language class

### Parameters

- **language** (Language) – Language class of the package langcodes
- **iso639\_variant** (str) – Optional variant to get the ‘bibliographic’ code instead, defaults to ‘B’

**name:** str

Name of the language

**ietf:** str

IETF BCP 47 language code

**iso639:** str

ISO-639 language code

**classmethod make**(*ietf*)

Make a new Lang based on IETF

### Parameters

**ietf** (Optional[str]) – IETF BCP 47 language code

### Return type

*Lang*

### Returns

A new Lang object

```
vardautomation.language.FRENCH = <vardautomation.language.Lang object>
```

French Lang object

```
vardautomation.language.ENGLISH = <vardautomation.language.Lang object>
```

English Lang object

```
vardautomation.language.JAPANESE = <vardautomation.language.Lang object>
```

Japanese Lang object

```
vardautomation.language.UNDEFINED = <vardautomation.language.Lang object>
```

Undefined Lang object



## 1.8 VPath

**class** vardautomation.vpathlib.VPath(\*args, \*\*kwargs)

Modified version of pathlib.Path

Construct a PurePath from one or several strings and or existing PurePath objects. The strings and path objects are combined so as to yield a canonicalized path, which is incorporated into the new PurePath object.

**format**(\*args, \*\*kwargs)

**Return type**

*VPath*

**Returns**

Formatted version of *vpath*, using substitutions from args and kwargs. The substitutions are identified by braces ('{' and '}')

**Parameters**

- **args** (Any) –
- **kwargs** (Any) –

**set\_track**(track\_number, /)

Set the track number by replacing the substitution "{track\_number}" by the track\_number specified

**Parameters**

**track\_number** (int) – Track number

**Return type**

*VPath*

**Returns**

Formatted VPath

**to\_str**()

**Return type**

str

**Returns**

String representation of the path, suitable for passing to system calls.

**append\_stem**(stem)

Append stem at the end of the VPath stem

**Parameters**

**stem** (str) – Stem to add

**Return type**

*VPath*

**Returns**

New VPath with the stem appended

**append\_suffix**(suffix)

Append stem at the end of the VPath suffix. Stolen from pathlib3x

**Parameters**

**suffix** (str) – Suffix to add. It has to start with '.'

**Return type***VPath***Returns**

New VPath with the file suffix appended

**copy**(*target*, \*, *follow\_symlinks=True*)

Wraps shutil.copy. Stolen from pathlib3x.

<https://docs.python.org/3/library/shutil.html#shutil.copy>**Parameters**

- **target** (PathLike[str] | str) – See Python official documentation
- **follow\_symlinks** (bool) – See Python official documentation

**Return type**

None

**copy2**(*target*, *follow\_symlinks=True*)

Wraps shutil.copy2. Stolen from pathlib3x.

<https://docs.python.org/3/library/shutil.html#shutil.copy2>**Parameters**

- **target** (PathLike[str] | str) – See Python official documentation
- **follow\_symlinks** (bool) – See Python official documentation

**Return type**

None

**copyfile**(*target*, *follow\_symlinks=True*)

Wraps shutil.copyfile. Stolen from pathlib3x.

<https://docs.python.org/3/library/shutil.html#shutil.copyfile>**Parameters**

- **target** (*VPath*) – See Python official documentation
- **follow\_symlinks** (bool) – See Python official documentation

**Return type**

None

**copymode**(*target*, *follow\_symlinks=True*)

Wraps shutil.copymode. Stolen from pathlib3x.

<https://docs.python.org/3/library/shutil.html#shutil.copymode>**Parameters**

- **target** (PathLike[str] | str) – See Python official documentation
- **follow\_symlinks** (bool) – See Python official documentation

**Return type**

None

**copystat**(*target*, *follow\_symlinks=True*)

Wraps shutil.copystat. Stolen from pathlib3x.

<https://docs.python.org/3/library/shutil.html#shutil.copystat>

**Parameters**

- **target** (PathLike[str] | str) – See Python official documentation
- **follow\_symlinks** (bool) – See Python official documentation

**Return type**

None

**copytree**(*target*, *symlinks=False*, *ignore=None*, *copy\_function=<function copy2>*,  
*ignore\_dangling\_symlinks=True*, *dirs\_exist\_ok=False*)

Wraps shutil.copytree. Stolen from pathlib3x.

<https://docs.python.org/3/library/shutil.html#shutil.copytree>

**Parameters**

- **target** (PathLike[str] | str) – See Python official documentation
- **symlinks** (bool) – See Python official documentation
- **ignore** (Optional[Callable[[PathLike[str] | str, List[str]], Iterable[str]]]) – See Python official documentation
- **copy\_function** (Callable[[PathLike[str] | str, PathLike[str] | str], Any]) – See Python official documentation
- **ignore\_dangling\_symlinks** (bool) – See Python official documentation
- **dirs\_exist\_ok** (bool) – See Python official documentation

**Return type**

None

**rmtree**(*ignore\_errors=False*, *onerror=None*)

Wraps shutil.rmtree. Stolen from pathlib3x.

<https://docs.python.org/3/library/shutil.html#shutil.rmtree>

**Parameters**

- **ignore\_errors** (bool) – See Python official documentation
- **onerror** (Optional[Callable[[Callable[... Any], str, Union[Tuple[Type[BaseException], BaseException, traceback], Tuple[None, None, None]], Any]]]) – See Python official documentation

**Return type**

None

**rm**(*ignore\_errors=False*)

Wraps os.remove.

**Parameters**

- **ignore\_errors** (bool) – Ignore errors emitted by os.remove

**Return type**

None

## 1.9 Types

`vardautomation.types.AnyPath = os.PathLike[str] | str`

Represent a PEP 604 union type

E.g. for `int | str`

`vardautomation.types.UpdateFunc`

alias of `Callable[[int, int], None]`

`vardautomation.types.VPSIdx`

alias of `Callable[[str], VideoNode]`

## 1.10 Internal functions

**class** `vardautomation.utils.Properties`

Collection of methods to get some properties from the parameters and/or the clip

**classmethod** `get_colour_range(cls, params, clip)`

Get the luma colour range specified in the params. Fallback to the clip properties.

**Parameters**

- **params** (`List[str]`) – Settings of the encoder.
- **clip** (`VideoNode`) – Source

**Return type**

`Tuple[int, int]`

**Returns**

A tuple of `min_luma` and `max_luma` value

**static** `get_depth(clip, /)`

Returns the bit depth of a `VideoNode` as an integer.

**Parameters**

**clip** (`VideoNode`) – Source clip

**Return type**

`int`

**Returns**

Bitdepth

**static** `get_csp(clip)`

Get the colourspace a the given clip based on its format

**Parameters**

**clip** (`VideoNode`) – Source clip

**Return type**

`str`

**Returns**

Colourspace suitable for x264

**static get\_encoder\_name**(*path*)

Get the encoder name from the file's tags

**Parameters**

**path** (PathLike[str] | str) – File path

**Return type**

str

**Returns**

Encoder name

**static get\_matrix\_name**(*frame, key*)

Gets FrameProp prop from frame *frame* with expected type *t* and then returns a corresponding string. This is necessary because x264 does not accept integers for the matrix/primaries/transfer.

**For a full list of accepted matrices, please check**

[http://www.chaneru.com/Roku/HLS/X264\\_Settings.htm#colormatrix](http://www.chaneru.com/Roku/HLS/X264_Settings.htm#colormatrix)

**Parameters**

- **frame** (VideoFrame) – Frame containing props
- **key** (str) – Prop to get. Must be \_Matrix, \_Primaries, or \_Transfer!
- **t** – Type of prop

**Return type**

str

**Returns**

string signalling the clip's matrix

**static get\_prop**(*frame, key, t*)

Gets FrameProp prop from frame *frame* with expected type *t* to satisfy the type checker. Function borrowed from lvsfunc.

**Parameters**

- **frame** (VideoFrame) – Frame containing props
- **key** (str) – Prop to get
- **t** (Type[TypeVar(T)]) – Type of prop

**Return type**

TypeVar(T)

**Returns**

frame.prop[key]

`vardautomation.render.clip_async_render`(*clip: vs.VideoNode, outfile: Optional[BinaryIO] = None, timecodes: None = None, progress: Optional[str] = 'Rendering clip...', callback: RenderCallback | List[RenderCallback] | None = None*) → None

`vardautomation.render.clip_async_render`(*clip: vs.VideoNode, outfile: Optional[BinaryIO] = None, timecodes: TextIO = None, progress: Optional[str] = 'Rendering clip...', callback: RenderCallback | List[RenderCallback] | None = None*) → List[float]

Render a clip by requesting frames asynchronously using clip.frames, providing for callback with frame number and frame object.

This is mostly a re-implementation of `VideoNode.output`, but a little bit slower since it's pure python. You only really need this when you want to render a clip while operating on each frame in order or you want timecodes without using `vspipe`.

Original function borrowed from `lvsfunc.render.clip_async_render`.

#### Parameters

- **clip** (`VideoNode`) – Clip to render.
- **outfile** (`Optional[BinaryIO]`) – Y4MPEG render output `BinaryIO` handle. If `None`, no Y4M output is performed. Use `sys.stdout.buffer` for stdout. (Default: `None`)
- **timecodes** (`Optional[TextIO]`) – Timecode v2 file `TextIO` handle. If `None`, timecodes will not be written.
- **progress** (`Optional[str]`) – String to use for render progress display. If empty or `None`, no progress display.
- **callback** (`Union[Callable[[int, VideoFrame], None], List[Callable[[int, VideoFrame], None]], None]`) – Single or list of callbacks to be preformed. The callbacks are called when each sequential frame is output, not when each frame is done.

#### Return type

`Optional[List[float]]`

#### Returns

List of timecodes from rendered clip.

```
vardautomation.render.audio_async_render(audio, outfile, header=WaveHeader.AUTO,  
                                         progress='Rendering audio...')
```

Render an audio by requesting frames asynchronously using `audio.frames`.

Implementation-like of `VideoNode.output` for an `AudioNode` that isn't in the Cython side yet.

#### Parameters

- **audio** (`AudioNode`) – Audio to render.
- **outfile** (`BinaryIO`) – Render output `BinaryIO` handle.
- **header** ([`WaveHeader`](#)) – Kind of Wave header. `WaveHeader.AUTO` adds a Wave64 header if the audio
  - Has more than 2 channels
  - Has a bitdepth > 16
  - Has more than 44100 samples
- **progress** (`Optional[str]`) – String to use for render progress display. If empty or `None`, no progress display.

#### Return type

`None`

```
class vardautomation.render.WaveFormat(value)
```

WAVE form `wFormatTag` IDs Complete list is in `mmreg.h` in Windows 10 SDK.

```
class vardautomation.render.WaveHeader(value)
```

Wave headers

## PYTHON MODULE INDEX

### V

`vardautomation`, [1](#)





## INDEX

### A

`a_cutters` (*vardautomation.automation.RunnerConfig* attribute), 46  
`a_enc_cut` (*vardautomation.config.FileInfo* attribute), 2  
`a_encoders` (*vardautomation.automation.RunnerConfig* attribute), 46  
`a_extracters` (*vardautomation.automation.RunnerConfig* attribute), 46  
`a_src` (*vardautomation.config.FileInfo* attribute), 2  
`a_src_cut` (*vardautomation.config.FileInfo* attribute), 2  
`ABR` (*vardautomation.tooling.audio.BitrateMode* attribute), 34  
`add_timestamps()` (*vardautomation.tooling.mux.MatroskaFile* method), 39  
`AnyPath` (in module *vardautomation.types*), 64  
`append_stem()` (*vardautomation.vpathlib.VPath* method), 61  
`append_suffix()` (*vardautomation.vpathlib.VPath* method), 61  
`append_to()` (*vardautomation.tooling.mux.MatroskaFile* method), 39  
`audio` (*vardautomation.config.FileInfo2* property), 4  
`audio_async_render()` (in module *vardautomation.render*), 66  
`audio_cut` (*vardautomation.config.FileInfo2* property), 4  
`AudioCutter` (class in *vardautomation.tooling.audio*), 24  
`AudioEncoder` (class in *vardautomation.tooling.audio*), 29  
`AudioExtractor` (class in *vardautomation.tooling.audio*), 21  
`audios` (*vardautomation.config.FileInfo2* attribute), 3  
`audios_cut` (*vardautomation.config.FileInfo2* attribute), 3  
`AudioTrack` (class in *vardautomation.tooling.mux*), 41  
`automux()` (*vardautomation.tooling.mux.MatroskaFile* static method), 37  
`autotrack()` (*vardautomation*

*tion.tooling.mux.MatroskaFile* class method), 37

### B

`B` (*vardautomation.comp.PictureType* attribute), 56  
`backlog` (*vardautomation.tooling.video.FFV1* attribute), 20  
`backlog` (*vardautomation.tooling.video.LosslessEncoder* attribute), 18  
`backlog` (*vardautomation.tooling.video.NVEncC* *Lossless* attribute), 19  
`backlog` (*vardautomation.tooling.video.VideoEncoder* attribute), 10  
`backlog` (*vardautomation.tooling.video.VideoLanEncoder* attribute), 11  
`backlog` (*vardautomation.tooling.video.X264* attribute), 16  
`backlog` (*vardautomation.tooling.video.X265* attribute), 14  
`BasicTool` (class in *vardautomation.tooling.base*), 8  
`bd_folder` (*vardautomation.chapterisation.MplsReader* attribute), 54  
`BEST` (*vardautomation.tooling.audio.FlacCompressionLevel* attribute), 36  
`binary` (*vardautomation.tooling.abstract.Tool* attribute), 8  
`binary` (*vardautomation.tooling.audio.AudioEncoder* attribute), 30  
`binary` (*vardautomation.tooling.audio.AudioExtractor* attribute), 22  
`binary` (*vardautomation.tooling.audio.Eac3toAudioExtractor* attribute), 23  
`binary` (*vardautomation.tooling.audio.FDKAACEncoder* attribute), 33  
`binary` (*vardautomation.tooling.audio.FFmpegAudioExtractor* attribute), 24  
`binary` (*vardautomation.tooling.audio.FlacEncoder* attribute), 34  
`binary` (*vardautomation.tooling.audio.MKVAudioExtractor* attribute), 23  
`binary` (*vardautomation.tooling.audio.OpusEncoder* attribute), 32

- `binary` (`vardautomation.tooling.audio.PassthroughAudioEncoder` attribute), 30
- `binary` (`vardautomation.tooling.audio.QAACEncoder` attribute), 31
- `binary` (`vardautomation.tooling.base.BasicTool` attribute), 9
- `binary` (`vardautomation.tooling.video.FFV1` attribute), 21
- `binary` (`vardautomation.tooling.video.LosslessEncoder` attribute), 19
- `binary` (`vardautomation.tooling.video.NVEncC` `Lossless` attribute), 20
- `binary` (`vardautomation.tooling.video.VideoEncoder` attribute), 10
- `binary` (`vardautomation.tooling.video.VideoLanEncoder` attribute), 12
- `binary` (`vardautomation.tooling.video.X264` attribute), 17
- `binary` (`vardautomation.tooling.video.X265` attribute), 14
- `BinaryPath` (class in `vardautomation.binary_path`), 59
- `BitrateMode` (class in `vardautomation.tooling.audio`), 34
- `BlurayShow` (class in `vardautomation.config`), 4
- C**
- `CBR` (`vardautomation.tooling.audio.BitrateMode` attribute), 35
- `Chapter` (class in `vardautomation.chapterisation`), 47
- `chapter` (`vardautomation.config.FileInfo` property), 2
- `chapter_file` (`vardautomation.chapterisation.Chapters` attribute), 47
- `chapter_file` (`vardautomation.chapterisation.IfoChapters` attribute), 54
- `chapter_file` (`vardautomation.chapterisation.MatroskaXMLChapters` attribute), 50
- `chapter_file` (`vardautomation.chapterisation.MplsChapters` attribute), 52
- `chapter_file` (`vardautomation.chapterisation.OGMChapters` attribute), 49
- `Chapters` (class in `vardautomation.chapterisation`), 47
- `chapters` (`vardautomation.chapterisation.IfoChapters` attribute), 53
- `chapters` (`vardautomation.chapterisation.MplsChapters` attribute), 51
- `CHAPTERS` (`vardautomation.tooling.mux.SplitMode` attribute), 39
- `ChaptersTrack` (class in `vardautomation.tooling.mux`), 42
- `charset` (`vardautomation.tooling.mux.ChaptersTrack` attribute), 42
- `clear_outputs` (`vardautomation.automation.RunnerConfig` attribute), 46
- `clip` (`vardautomation.automation.Patch` attribute), 46
- `clip` (`vardautomation.automation.SelfRunner` attribute), 44
- `clip` (`vardautomation.config.FileInfo` attribute), 2
- `clip` (`vardautomation.tooling.video.FFV1` attribute), 21
- `clip` (`vardautomation.tooling.video.LosslessEncoder` attribute), 19
- `clip` (`vardautomation.tooling.video.NVEncC` `Lossless` attribute), 20
- `clip` (`vardautomation.tooling.video.VideoEncoder` attribute), 10
- `clip` (`vardautomation.tooling.video.VideoLanEncoder` attribute), 12
- `clip` (`vardautomation.tooling.video.X264` attribute), 17
- `clip` (`vardautomation.tooling.video.X265` attribute), 14
- `clip_async_render()` (in module `vardautomation.render`), 65
- `clip_cut` (`vardautomation.config.FileInfo` attribute), 2
- `collection_name` (`vardautomation.comp.SlowPicsConf` attribute), 57
- `command` (`vardautomation.tooling.mux.MatroskaFile` property), 37
- `Comparison` (class in `vardautomation.comp`), 57
- `config` (`vardautomation.automation.SelfRunner` attribute), 44
- `copy()` (`vardautomation.chapterisation.Chapters` method), 48
- `copy()` (`vardautomation.chapterisation.IfoChapters` method), 53
- `copy()` (`vardautomation.chapterisation.MatroskaXMLChapters` method), 51
- `copy()` (`vardautomation.chapterisation.MplsChapters` method), 52
- `copy()` (`vardautomation.chapterisation.OGMChapters` method), 50
- `copy()` (`vardautomation.vpathlib.VPath` method), 62
- `copy2()` (`vardautomation.vpathlib.VPath` method), 62
- `copyfile()` (`vardautomation.vpathlib.VPath` method), 62
- `copymode()` (`vardautomation.vpathlib.VPath` method), 62
- `copystat()` (`vardautomation.vpathlib.VPath` method), 62
- `copytree()` (`vardautomation.vpathlib.VPath` method), 63
- `count()` (`vardautomation.tooling.mux.AudioTrack` method), 41
- `count()` (`vardautomation.tooling.mux.ChaptersTrack` method), 42

- count() (vardautomation.tooling.mux.MediaTrack method), 40
- count() (vardautomation.tooling.mux.SubtitleTrack method), 42
- count() (vardautomation.tooling.mux.VideoTrack method), 41
- create (vardautomation.chapterisation.IfoChapters attribute), 53
- create (vardautomation.chapterisation.MplsChapters attribute), 52
- create() (vardautomation.chapterisation.Chapters method), 48
- create() (vardautomation.chapterisation.MatroskaXMLChapters method), 50
- create() (vardautomation.chapterisation.OGMChapters method), 49
- create\_qpfile() (vardautomation.chapterisation.Chapters method), 48
- create\_qpfile() (vardautomation.chapterisation.IfoChapters method), 53
- create\_qpfile() (vardautomation.chapterisation.MatroskaXMLChapters method), 51
- create\_qpfile() (vardautomation.chapterisation.MplsChapters method), 52
- create\_qpfile() (vardautomation.chapterisation.OGMChapters method), 50
- CVBR (vardautomation.tooling.audio.BitrateMode attribute), 35
- ## D
- debug (vardautomation.automation.Patch attribute), 47
- default\_chap\_name (vardautomation.chapterisation.IfoReader attribute), 55
- default\_chap\_name (vardautomation.chapterisation.MplsReader attribute), 54
- do\_cleanup() (vardautomation.automation.Patch method), 47
- DURATION (vardautomation.tooling.mux.SplitMode attribute), 39
- dvd\_folder (vardautomation.chapterisation.IfoReader attribute), 55
- ## E
- eac3to (vardautomation.binary\_path.BinaryPath attribute), 59
- Eac3toAudioExtractor (class in vardautomation.tooling.audio), 23
- EIGHT (vardautomation.tooling.audio.FlacCompressionLevel attribute), 36
- ELEVEN (vardautomation.tooling.audio.FlacCompressionLevel attribute), 36
- encoder (vardautomation.automation.Patch attribute), 46
- end\_frame (vardautomation.chapterisation.Chapter attribute), 47
- ENGLISH (in module vardautomation.language), 60
- episode() (vardautomation.config.BlurayShow method), 6
- episodes() (vardautomation.config.BlurayShow method), 6
- extract() (vardautomation.comp.Comparison method), 57
- ezpztrim() (vardautomation.tooling.audio.EztrimCutter class method), 26
- EztrimCutter (class in vardautomation.tooling.audio), 26
- ## F
- FAST (vardautomation.tooling.audio.FlacCompressionLevel attribute), 36
- fdkaac (vardautomation.binary\_path.BinaryPath attribute), 59
- FDKAACEncoder (class in vardautomation.tooling.audio), 32
- ffmpeg (vardautomation.binary\_path.BinaryPath attribute), 59
- FFMPEG (vardautomation.comp.Writer attribute), 56
- FfmpegAudioExtractor (class in vardautomation.tooling.audio), 23
- ffmsindex (vardautomation.binary\_path.BinaryPath attribute), 59
- FFV1 (class in vardautomation.tooling.video), 20
- file (vardautomation.automation.Patch attribute), 46
- file (vardautomation.automation.SelfRunner attribute), 44
- file (vardautomation.tooling.audio.AudioCutter attribute), 24
- file (vardautomation.tooling.audio.AudioEncoder attribute), 29
- file (vardautomation.tooling.audio.AudioExtractor attribute), 22
- file (vardautomation.tooling.audio.Eac3toAudioExtractor attribute), 23
- file (vardautomation.tooling.audio.EztrimCutter attribute), 26
- file (vardautomation.tooling.audio.FDKAACEncoder attribute), 33

file (vardautomation.tooling.audio.FFmpegAudioExtractor attribute), 24

file (vardautomation.tooling.audio.FlacEncoder attribute), 34

file (vardautomation.tooling.audio.MKVAudioExtractor attribute), 22

file (vardautomation.tooling.audio.OpusEncoder attribute), 32

file (vardautomation.tooling.audio.PassthroughAudioEncoder attribute), 30

file (vardautomation.tooling.audio.PassthroughCutter attribute), 28

file (vardautomation.tooling.audio.QAACEncoder attribute), 31

file (vardautomation.tooling.audio.ScipyCutter attribute), 25

file (vardautomation.tooling.audio.SoxCutter attribute), 27

file (vardautomation.tooling.base.BasicTool attribute), 8

file (vardautomation.tooling.video.FFV1 attribute), 21

file (vardautomation.tooling.video.LosslessEncoder attribute), 19

file (vardautomation.tooling.video.NVencLossless attribute), 20

file (vardautomation.tooling.video.VideoEncoder attribute), 9

file (vardautomation.tooling.video.VideoLanEncoder attribute), 12

file (vardautomation.tooling.video.X264 attribute), 17

file (vardautomation.tooling.video.X265 attribute), 14

FileInfo (class in vardautomation.config), 1

FileInfo2 (class in vardautomation.config), 3

FIVE (vardautomation.tooling.audio.FlacCompressionLevel attribute), 35

flac (vardautomation.binary\_path.BinaryPath attribute), 59

FlacCompressionLevel (class in vardautomation.tooling.audio), 35

FlacEncoder (class in vardautomation.tooling.audio), 34

format() (vardautomation.vpathlib.VPath method), 61

FOUR (vardautomation.tooling.audio.FlacCompressionLevel attribute), 35

fps (vardautomation.chapterisation.IfoChapters attribute), 53

fps (vardautomation.chapterisation.MplsChapters attribute), 51

frames (vardautomation.tooling.misc.Qpfile attribute), 43

FRAMES (vardautomation.tooling.mux.SplitMode attribute), 39

FRENCH (in module vardautomation.language), 60

generate\_silence() (vardautomation.tooling.audio.AudioCutter class method), 25

generate\_silence() (vardautomation.tooling.audio.EztrimCutter class method), 27

generate\_silence() (vardautomation.tooling.audio.PassthroughCutter class method), 29

generate\_silence() (vardautomation.tooling.audio.ScipyCutter class method), 26

generate\_silence() (vardautomation.tooling.audio.SoxCutter class method), 28

get\_colour\_range() (vardautomation.utils.Properties class method), 64

get\_csp() (vardautomation.utils.Properties static method), 64

get\_depth() (vardautomation.utils.Properties static method), 64

get\_encoder\_name() (vardautomation.utils.Properties static method), 64

get\_matrix\_name() (vardautomation.utils.Properties static method), 65

get\_playlist() (vardautomation.chapterisation.MplsReader method), 54

get\_prop() (vardautomation.utils.Properties static method), 65

get\_vs\_core() (in module vardautomation.tooling.misc), 43

global\_opts (vardautomation.tooling.mux.MatroskaFile attribute), 36

## H

HARD\_CBR (vardautomation.tooling.audio.BitrateMode attribute), 35

## I

I (vardautomation.comp.PictureType attribute), 56

idx (vardautomation.config.FileInfo attribute), 1

ietf (vardautomation.language.Lang attribute), 60

ifo\_folder (vardautomation.chapterisation.IfoReader attribute), 55

IfoChapters (class in vardautomation.chapterisation), 52

IfoReader (class in vardautomation.chapterisation), 55

IMWRI (vardautomation.comp.Writer attribute), 56

index() (vardautomation.tooling.mux.AudioTrack method), 41

- [index\(\)](#) ([vardautomation.tooling.mux.ChaptersTrack](#) method), 42  
[index\(\)](#) ([vardautomation.tooling.mux.MediaTrack](#) method), 40  
[index\(\)](#) ([vardautomation.tooling.mux.SubtitleTrack](#) method), 42  
[index\(\)](#) ([vardautomation.tooling.mux.VideoTrack](#) method), 41  
[inject\\_qpfile\\_params\(\)](#) ([vardautomation.automation.SelfRunner](#) method), 45  
[iso639](#) ([vardautomation.language.Lang](#) attribute), 60
- ## J
- [JAPANESE](#) (in module [vardautomation.language](#)), 60
- ## K
- [kwargs](#) ([vardautomation.tooling.audio.AudioCutter](#) attribute), 24  
[kwargs](#) ([vardautomation.tooling.audio.EztrimCutter](#) attribute), 26  
[kwargs](#) ([vardautomation.tooling.audio.PassthroughCutter](#) attribute), 28  
[kwargs](#) ([vardautomation.tooling.audio.ScipyCutter](#) attribute), 25  
[kwargs](#) ([vardautomation.tooling.audio.SoxCutter](#) attribute), 27
- ## L
- [Lang](#) (class in [vardautomation.language](#)), 60  
[lang](#) ([vardautomation.chapterisation.Chapter](#) attribute), 47  
[lang](#) ([vardautomation.chapterisation.IfoReader](#) attribute), 55  
[lang](#) ([vardautomation.chapterisation.MplsReader](#) attribute), 54  
[lang](#) ([vardautomation.tooling.mux.AudioTrack](#) attribute), 41  
[lang](#) ([vardautomation.tooling.mux.ChaptersTrack](#) attribute), 42  
[lang](#) ([vardautomation.tooling.mux.MediaTrack](#) attribute), 40  
[lang](#) ([vardautomation.tooling.mux.SubtitleTrack](#) attribute), 42  
[lang](#) ([vardautomation.tooling.mux.VideoTrack](#) attribute), 41  
[LosslessEncoder](#) (class in [vardautomation.tooling.video](#)), 17
- ## M
- [m2ts](#) ([vardautomation.chapterisation.MplsChapters](#) attribute), 51  
[m2ts\\_folder](#) ([vardautomation.chapterisation.MplsReader](#) attribute), 54  
[magick\\_compare\(\)](#) ([vardautomation.comp.Comparison](#) method), 58  
[make\(\)](#) ([vardautomation.language.Lang](#) class method), 60  
[make\\_comps\(\)](#) (in module [vardautomation.comp](#)), 58  
[make\\_qpfile\(\)](#) (in module [vardautomation.tooling.misc](#)), 43  
[MatroskaFile](#) (class in [vardautomation.tooling.mux](#)), 36  
[MatroskaXMLChapters](#) (class in [vardautomation.chapterisation](#)), 50  
[media\\_info](#) ([vardautomation.config.FileInfo](#) property), 2  
[MediaTrack](#) (class in [vardautomation.tooling.mux](#)), 40  
[mkv](#) ([vardautomation.automation.RunnerConfig](#) attribute), 46  
[MKVAudioExtractor](#) (class in [vardautomation.tooling.audio](#)), 22  
[mkvextract](#) ([vardautomation.binary\\_path.BinaryPath](#) attribute), 59  
[mkvmerge](#) ([vardautomation.binary\\_path.BinaryPath](#) attribute), 59  
[module](#)  
[vardautomation](#), 1  
[mpls\\_chapters](#) ([vardautomation.chapterisation.MplsReader.MplsFile](#) attribute), 54  
[mpls\\_file](#) ([vardautomation.chapterisation.MplsReader.MplsFile](#) attribute), 54  
[mpls\\_folder](#) ([vardautomation.chapterisation.MplsReader](#) attribute), 54  
[MplsChapters](#) (class in [vardautomation.chapterisation](#)), 51  
[MplsReader](#) (class in [vardautomation.chapterisation](#)), 54  
[MplsReader.MplsFile](#) (class in [vardautomation.chapterisation](#)), 54  
[mux\(\)](#) ([vardautomation.tooling.mux.MatroskaFile](#) method), 37
- ## N
- [name](#) ([vardautomation.chapterisation.Chapter](#) attribute), 47  
[name](#) ([vardautomation.config.FileInfo](#) attribute), 2  
[name](#) ([vardautomation.language.Lang](#) attribute), 60  
[name](#) ([vardautomation.tooling.mux.AudioTrack](#) attribute), 41  
[name](#) ([vardautomation.tooling.mux.MediaTrack](#) attribute), 40  
[name](#) ([vardautomation.tooling.mux.SubtitleTrack](#) attribute), 42  
[name](#) ([vardautomation.tooling.mux.VideoTrack](#) attribute), 41



- name\_clip\_output (*vardautomation.config.FileInfo* attribute), 2
- name\_file\_final (*vardautomation.config.FileInfo* attribute), 2
- nced() (*vardautomation.config.BlurayShow* method), 5
- nceds() (*vardautomation.config.BlurayShow* method), 5
- ncop() (*vardautomation.config.BlurayShow* method), 5
- ncops() (*vardautomation.config.BlurayShow* method), 5
- NINE (*vardautomation.tooling.audio.FlacCompressionLevel* attribute), 36
- nsfw (*vardautomation.comp.SlowPicsConf* attribute), 57
- num\_prop (*vardautomation.config.FileInfo* property), 3
- nvencc (*vardautomation.binary\_path.BinaryPath* attribute), 59
- NVencClossless (class in *vardautomation.tooling.video*), 19
- ## O
- OGMChapters (class in *vardautomation.chapterisation*), 49
- ONE (*vardautomation.tooling.audio.FlacCompressionLevel* attribute), 35
- OPENCV (*vardautomation.comp.Writer* attribute), 56
- optimise (*vardautomation.comp.SlowPicsConf* attribute), 57
- opts (*vardautomation.tooling.mux.AudioTrack* attribute), 41
- opts (*vardautomation.tooling.mux.ChaptersTrack* attribute), 43
- opts (*vardautomation.tooling.mux.MediaTrack* attribute), 40
- opts (*vardautomation.tooling.mux.SubtitleTrack* attribute), 42
- opts (*vardautomation.tooling.mux.Track* attribute), 40
- opts (*vardautomation.tooling.mux.VideoTrack* attribute), 41
- opusenc (*vardautomation.binary\_path.BinaryPath* attribute), 59
- OpusEncoder (class in *vardautomation.tooling.audio*), 32
- order (*vardautomation.automation.RunnerConfig* attribute), 46
- output\_filename (*vardautomation.automation.Patch* attribute), 46
- ## P
- P (*vardautomation.comp.PictureType* attribute), 56
- params (*vardautomation.tooling.abstract.Tool* attribute), 8
- params (*vardautomation.tooling.audio.AudioEncoder* attribute), 30
- params (*vardautomation.tooling.audio.AudioExtractor* attribute), 22
- params (*vardautomation.tooling.audio.Eac3toAudioExtractor* attribute), 23
- params (*vardautomation.tooling.audio.FDKAACEncoder* attribute), 33
- params (*vardautomation.tooling.audio.FFmpegAudioExtractor* attribute), 24
- params (*vardautomation.tooling.audio.FlacEncoder* attribute), 34
- params (*vardautomation.tooling.audio.MKVAudioExtractor* attribute), 23
- params (*vardautomation.tooling.audio.OpusEncoder* attribute), 32
- params (*vardautomation.tooling.audio.PassthroughAudioEncoder* attribute), 30
- params (*vardautomation.tooling.audio.QAACEncoder* attribute), 31
- params (*vardautomation.tooling.base.BasicTool* attribute), 9
- params (*vardautomation.tooling.video.FFV1* attribute), 21
- params (*vardautomation.tooling.video.LosslessEncoder* attribute), 19
- params (*vardautomation.tooling.video.NVencClossless* attribute), 20
- params (*vardautomation.tooling.video.VideoEncoder* attribute), 10
- params (*vardautomation.tooling.video.VideoLanEncoder* attribute), 12
- params (*vardautomation.tooling.video.X264* attribute), 17
- params (*vardautomation.tooling.video.X265* attribute), 14
- params\_asdict (*vardautomation.tooling.video.VideoLanEncoder* property), 11
- params\_asdict (*vardautomation.tooling.video.X264* property), 16
- params\_asdict (*vardautomation.tooling.video.X265* property), 14
- parse\_ifo() (*vardautomation.chapterisation.IfoReader* method), 56
- parse\_mpls() (*vardautomation.chapterisation.MplsReader* method), 55
- PARTS (*vardautomation.tooling.mux.SplitMode* attribute), 39
- PARTS\_FRAMES (*vardautomation.tooling.mux.SplitMode* attribute), 39
- PassthroughAudioEncoder (class in *vardautomation.tooling.audio*), 30
- PassthroughCutter (class in *vardautomation.tooling.audio*), 28
- Patch (class in *vardautomation.automation*), 46
- path (*vardautomation.config.FileInfo* attribute), 1

- [path \(vardautomation.tooling.misc.Qpfile attribute\), 43](#)  
[path \(vardautomation.tooling.mux.AudioTrack attribute\), 41](#)  
[path \(vardautomation.tooling.mux.ChaptersTrack attribute\), 42](#)  
[path \(vardautomation.tooling.mux.MediaTrack attribute\), 40](#)  
[path \(vardautomation.tooling.mux.SubtitleTrack attribute\), 42](#)  
[path \(vardautomation.tooling.mux.Track attribute\), 40](#)  
[path \(vardautomation.tooling.mux.VideoTrack attribute\), 41](#)  
[path\\_without\\_ext \(vardautomation.config.FileInfo attribute\), 1](#)  
[PictureType \(class in vardautomation.comp\), 56](#)  
[PILLOW \(vardautomation.comp.Writer attribute\), 56](#)  
[plp\\_function \(vardautomation.automation.SelfRunner attribute\), 44](#)  
[prefetch \(vardautomation.tooling.video.FFV1 attribute\), 20](#)  
[prefetch \(vardautomation.tooling.video.LosslessEncoder attribute\), 18](#)  
[prefetch \(vardautomation.tooling.video.NVencCLossless attribute\), 19](#)  
[prefetch \(vardautomation.tooling.video.VideoEncoder attribute\), 10](#)  
[prefetch \(vardautomation.tooling.video.VideoLanEncoder attribute\), 12](#)  
[prefetch \(vardautomation.tooling.video.X264 attribute\), 16](#)  
[prefetch \(vardautomation.tooling.video.X265 attribute\), 14](#)  
[preset \(vardautomation.config.FileInfo attribute\), 2](#)  
[PresetAAC \(in module vardautomation.config\), 6](#)  
[PresetBD \(in module vardautomation.config\), 6](#)  
[PresetChapOGM \(in module vardautomation.config\), 7](#)  
[PresetChapXML \(in module vardautomation.config\), 7](#)  
[PresetEAC3 \(in module vardautomation.config\), 7](#)  
[PresetFLAC \(in module vardautomation.config\), 7](#)  
[PresetOpus \(in module vardautomation.config\), 7](#)  
[PresetWEB \(in module vardautomation.config\), 6](#)  
[progress\\_update \(vardautomation.tooling.video.FFV1 attribute\), 20](#)  
[progress\\_update \(vardautomation.tooling.video.LosslessEncoder attribute\), 18](#)  
[progress\\_update \(vardautomation.tooling.video.NVencCLossless attribute\), 19](#)  
[progress\\_update \(vardautomation.tooling.video.VideoEncoder attribute\), 10](#)  
[progress\\_update \(vardautomation.tooling.video.VideoLanEncoder attribute\), 12](#)  
[progress\\_update \(vardautomation.tooling.video.X264 attribute\), 16](#)  
[progress\\_update \(vardautomation.tooling.video.X265 attribute\), 14](#)  
[Properties \(class in vardautomation.utils\), 64](#)  
[public \(vardautomation.comp.SlowPicsConf attribute\), 57](#)  
[PYQT \(vardautomation.comp.Writer attribute\), 56](#)  
[PYTHON \(vardautomation.comp.Writer attribute\), 56](#)
- ## Q
- [qaac \(vardautomation.binary\\_path.BinaryPath attribute\), 59](#)  
[QAALEncoder \(class in vardautomation.tooling.audio\), 31](#)  
[Qpfile \(class in vardautomation.tooling.misc\), 43](#)
- ## R
- [ranges \(vardautomation.automation.Patch attribute\), 46](#)  
[rclone \(vardautomation.binary\\_path.BinaryPath attribute\), 59](#)  
[register\\_nceds\(\) \(vardautomation.config.BlurayShow method\), 5](#)  
[register\\_ncops\(\) \(vardautomation.config.BlurayShow method\), 5](#)  
[remove\\_after \(vardautomation.comp.SlowPicsConf attribute\), 57](#)  
[rename\\_final\\_file\(\) \(vardautomation.automation.SelfRunner method\), 45](#)  
[resumable \(vardautomation.tooling.video.VideoLanEncoder attribute\), 12](#)  
[resumable \(vardautomation.tooling.video.X264 attribute\), 17](#)  
[resumable \(vardautomation.tooling.video.X265 attribute\), 15](#)  
[rm\(\) \(vardautomation.vpathlib.VPath method\), 63](#)  
[rmtree\(\) \(vardautomation.vpathlib.VPath method\), 63](#)  
[run\(\) \(vardautomation.automation.Patch method\), 47](#)  
[run\(\) \(vardautomation.automation.SelfRunner method\), 44](#)  
[run\(\) \(vardautomation.tooling.abstract.Tool method\), 8](#)  
[run\(\) \(vardautomation.tooling.audio.AudioCutter method\), 25](#)  
[run\(\) \(vardautomation.tooling.audio.AudioEncoder method\), 29](#)  
[run\(\) \(vardautomation.tooling.audio.AudioExtractor method\), 22](#)  
[run\(\) \(vardautomation.tooling.audio.Eac3toAudioExtractor method\), 23](#)

`run()` (*vardautomation.tooling.audio.EztrimCutter* method), 26  
`run()` (*vardautomation.tooling.audio.FDKAACEncoder* method), 33  
`run()` (*vardautomation.tooling.audio.FFmpegAudioExtractor* method), 24  
`run()` (*vardautomation.tooling.audio.FlacEncoder* method), 34  
`run()` (*vardautomation.tooling.audio.MKVAudioExtractor* method), 22  
`run()` (*vardautomation.tooling.audio.OpusEncoder* method), 32  
`run()` (*vardautomation.tooling.audio.PassthroughAudioEncoder* method), 30  
`run()` (*vardautomation.tooling.audio.PassthroughCutter* method), 28  
`run()` (*vardautomation.tooling.audio.QAACEncoder* method), 31  
`run()` (*vardautomation.tooling.audio.ScipyCutter* method), 25  
`run()` (*vardautomation.tooling.audio.SoxCutter* method), 27  
`run()` (*vardautomation.tooling.base.BasicTool* method), 9  
`run()` (*vardautomation.tooling.video.FFV1* method), 21  
`run()` (*vardautomation.tooling.video.LosslessEncoder* method), 18  
`run()` (*vardautomation.tooling.video.NVencCLossless* method), 19  
`run()` (*vardautomation.tooling.video.VideoEncoder* method), 10  
`run()` (*vardautomation.tooling.video.VideoLanEncoder* method), 12  
`run()` (*vardautomation.tooling.video.X264* method), 16  
`run()` (*vardautomation.tooling.video.X265* method), 14  
`run_enc()` (*vardautomation.tooling.video.FFV1* method), 21  
`run_enc()` (*vardautomation.tooling.video.LosslessEncoder* method), 18  
`run_enc()` (*vardautomation.tooling.video.NVencCLossless* method), 19  
`run_enc()` (*vardautomation.tooling.video.VideoEncoder* method), 10  
`run_enc()` (*vardautomation.tooling.video.VideoLanEncoder* method), 12  
`run_enc()` (*vardautomation.tooling.video.X264* method), 16  
`run_enc()` (*vardautomation.tooling.video.X265* method), 14  
`RunnerConfig` (class in *vardautomation.automation*), 45  
`RunnerConfig.Order` (class in *vardautomation.automation*), 45  
**S**  
`ScipyCutter` (class in *vardautomation.tooling.audio*), 25  
`scipytrim()` (*vardautomation.tooling.audio.ScipyCutter* class method), 25  
`SelfRunner` (class in *vardautomation.automation*), 44  
`set_name_clip_output_ext()` (*vardautomation.config.FileInfo* method), 2  
`set_names` (*vardautomation.chapterisation.IfoChapters* attribute), 53  
`set_names` (*vardautomation.chapterisation.MplsChapters* attribute), 52  
`set_names()` (*vardautomation.chapterisation.Chapters* method), 48  
`set_names()` (*vardautomation.chapterisation.MatroskaXMLChapters* method), 50  
`set_names()` (*vardautomation.chapterisation.OGMChapters* method), 49  
`set_track()` (*vardautomation.vpathlib.VPath* method), 61  
`set_variable()` (*vardautomation.tooling.abstract.Tool* method), 8  
`set_variable()` (*vardautomation.tooling.audio.AudioEncoder* method), 30  
`set_variable()` (*vardautomation.tooling.audio.AudioExtractor* method), 22  
`set_variable()` (*vardautomation.tooling.audio.Eac3toAudioExtractor* method), 23  
`set_variable()` (*vardautomation.tooling.audio.FDKAACEncoder* method), 33  
`set_variable()` (*vardautomation.tooling.audio.FFmpegAudioExtractor* method), 24  
`set_variable()` (*vardautomation.tooling.audio.FlacEncoder* method), 34  
`set_variable()` (*vardautomation.tooling.audio.MKVAudioExtractor* method), 22  
`set_variable()` (*vardautomation.tooling.audio.OpusEncoder* method), 32



- `set_variable()` (*vardautomation.tooling.audio.PassthroughAudioEncoder method*), 30  
`set_variable()` (*vardautomation.tooling.audio.QAACEncoder method*), 31  
`set_variable()` (*vardautomation.tooling.base.BasicTool method*), 9  
`set_variable()` (*vardautomation.tooling.video.FFV1 method*), 21  
`set_variable()` (*vardautomation.tooling.video.LosslessEncoder method*), 18  
`set_variable()` (*vardautomation.tooling.video.NVencClossless method*), 20  
`set_variable()` (*vardautomation.tooling.video.VideoEncoder method*), 10  
`set_variable()` (*vardautomation.tooling.video.VideoLanEncoder method*), 12  
`set_variable()` (*vardautomation.tooling.video.X264 method*), 17  
`set_variable()` (*vardautomation.tooling.video.X265 method*), 15  
SEVEN (*vardautomation.tooling.audio.FlacCompressionLevel attribute*), 35  
`shift_times` (*vardautomation.chapterisation.IfoChapters attribute*), 53  
`shift_times` (*vardautomation.chapterisation.MplsChapters attribute*), 52  
`shift_times()` (*vardautomation.chapterisation.Chapters method*), 48  
`shift_times()` (*vardautomation.chapterisation.MatroskaXMLChapters method*), 51  
`shift_times()` (*vardautomation.chapterisation.OGMChapters method*), 49  
SIX (*vardautomation.tooling.audio.FlacCompressionLevel attribute*), 35  
SIZE (*vardautomation.tooling.mux.SplitMode attribute*), 39  
SlowPicsConf (*class in vardautomation.comp*), 56  
sox (*vardautomation.binary\_path.BinaryPath attribute*), 59  
SoxCutter (*class in vardautomation.tooling.audio*), 27  
`soxtrim()` (*vardautomation.tooling.audio.SoxCutter class method*), 27  
`split()` (*vardautomation.tooling.mux.MatroskaFile method*), 37  
`split_chapters()` (*vardautomation.tooling.mux.MatroskaFile method*), 38  
`split_duration()` (*vardautomation.tooling.mux.MatroskaFile method*), 38  
`split_frames()` (*vardautomation.tooling.mux.MatroskaFile method*), 38  
`split_parts()` (*vardautomation.tooling.mux.MatroskaFile method*), 38  
`split_parts_frames()` (*vardautomation.tooling.mux.MatroskaFile method*), 38  
`split_size()` (*vardautomation.tooling.mux.MatroskaFile method*), 38  
`split_timestamps()` (*vardautomation.tooling.mux.MatroskaFile method*), 38  
SplitMode (*class in vardautomation.tooling.mux*), 39  
start\_frame (*vardautomation.chapterisation.Chapter attribute*), 47  
SubtitleTrack (*class in vardautomation.tooling.mux*), 41  
suffix\_name (*vardautomation.tooling.video.FFV1 attribute*), 20  
suffix\_name (*vardautomation.tooling.video.LosslessEncoder attribute*), 18  
suffix\_name (*vardautomation.tooling.video.NVencClossless attribute*), 19  
**T**  
TEN (*vardautomation.tooling.audio.FlacCompressionLevel attribute*), 36  
THREE (*vardautomation.tooling.audio.FlacCompressionLevel attribute*), 35  
TIMESTAMPS (*vardautomation.tooling.mux.SplitMode attribute*), 39  
to\_chapters() (*vardautomation.chapterisation.Chapters method*), 48  
to\_chapters() (*vardautomation.chapterisation.IfoChapters method*), 53  
to\_chapters() (*vardautomation.chapterisation.MatroskaXMLChapters method*), 51  
to\_chapters() (*vardautomation.chapterisation.MplsChapters method*), 52  
to\_chapters() (*vardautomation*

tion.chapterisation.OGMChapters method),  
 49  
 to\_str() (vardautomation.vpathlib.VPath method), 61  
 Tool (class in vardautomation.tooling.abstract), 7  
 Track (class in vardautomation.tooling.mux), 39  
 track (vardautomation.tooling.audio.AudioCutter  
 attribute), 24  
 track (vardautomation.tooling.audio.AudioEncoder at-  
 tribute), 29  
 track (vardautomation.tooling.audio.EztrimCutter at-  
 tribute), 26  
 track (vardautomation.tooling.audio.FDKAACEncoder  
 attribute), 33  
 track (vardautomation.tooling.audio.FlacEncoder at-  
 tribute), 34  
 track (vardautomation.tooling.audio.OpusEncoder at-  
 tribute), 32  
 track (vardautomation.tooling.audio.PassthroughAudioEncoder  
 attribute), 30  
 track (vardautomation.tooling.audio.PassthroughCutter  
 attribute), 28  
 track (vardautomation.tooling.audio.QAACEncoder at-  
 tribute), 31  
 track (vardautomation.tooling.audio.ScipyCutter at-  
 tribute), 25  
 track (vardautomation.tooling.audio.SoxCutter at-  
 tribute), 27  
 track\_in (vardautoma-  
 tion.tooling.audio.AudioExtractor attribute),  
 22  
 track\_in (vardautoma-  
 tion.tooling.audio.Eac3toAudioExtractor  
 attribute), 23  
 track\_in (vardautoma-  
 tion.tooling.audio.FFmpegAudioExtractor  
 attribute), 24  
 track\_in (vardautoma-  
 tion.tooling.audio.MKVAudioExtractor at-  
 tribute), 23  
 track\_lang (vardautomation.tooling.mux.MatroskaFile  
 property), 37  
 track\_out (vardautoma-  
 tion.tooling.audio.AudioExtractor attribute),  
 22  
 track\_out (vardautoma-  
 tion.tooling.audio.Eac3toAudioExtractor  
 attribute), 23  
 track\_out (vardautoma-  
 tion.tooling.audio.FFmpegAudioExtractor  
 attribute), 24  
 track\_out (vardautoma-  
 tion.tooling.audio.MKVAudioExtractor at-  
 tribute), 23  
 trims\_or\_dfs (vardautomation.config.FileInfo prop-  
 erty), 2  
 trims\_or\_dfs (vardautomation.config.FileInfo2 prop-  
 erty), 3  
 TVBR (vardautomation.tooling.audio.BitrateMode at-  
 tribute), 35  
 TWELVE (vardautomation.tooling.audio.FlacCompressionLevel  
 attribute), 36  
 TWO (vardautomation.tooling.audio.FlacCompressionLevel  
 attribute), 35  
**U**  
 UNDEFINED (in module vardautomation.language), 60  
 UpdateFunc (in module vardautomation.types), 64  
 upload\_ftp() (vardautomation.automation.SelfRunner  
 method), 45  
 upload\_to\_slowpics() (vardautoma-  
 tion.comp.Comparison method), 58  
**V**  
 v\_encoder (vardautomation.automation.RunnerConfig  
 attribute), 45  
 v\_lossless\_encoder (vardautoma-  
 tion.automation.RunnerConfig attribute),  
 46  
 vardautomation  
 module, 1  
 VARDOU (vardautomation.tooling.audio.FlacCompressionLevel  
 attribute), 36  
 VBR (vardautomation.tooling.audio.BitrateMode at-  
 tribute), 35  
 VideoEncoder (class in vardautomation.tooling.video),  
 9  
 VideoLanEncoder (class in vardautoma-  
 tion.tooling.video), 10  
 VideoTrack (class in vardautomation.tooling.mux), 40  
 VPath (class in vardautomation.vpathlib), 61  
 VPSIdx (in module vardautomation.types), 64  
**W**  
 WaveFormat (class in vardautomation.render), 66  
 WaveHeader (class in vardautomation.render), 66  
 work\_filename (vardautomation.config.FileInfo at-  
 tribute), 1  
 work\_files (vardautomation.automation.SelfRunner at-  
 tribute), 44  
 workdir (vardautomation.automation.Patch attribute),  
 46  
 workdir (vardautomation.config.FileInfo attribute), 2  
 write\_a\_src() (vardautomation.config.FileInfo2  
 method), 4  
 write\_a\_src\_cut() (vardautomation.config.FileInfo2  
 method), 4

`write_playlist()` (*vardautomation.chapterisation.MplsReader* method), 55

`write_programs()` (*vardautomation.chapterisation.IfoReader* method), 55

`Writer` (class in *vardautomation.comp*), 56

## X

`X264` (class in *vardautomation.tooling.video*), 15

`x264` (*vardautomation.binary\_path.BinaryPath* attribute), 59

`X265` (class in *vardautomation.tooling.video*), 13

`x265` (*vardautomation.binary\_path.BinaryPath* attribute), 59

`xml_tag` (*vardautomation.tooling.audio.AudioEncoder* attribute), 29

`xml_tag` (*vardautomation.tooling.audio.FDKAACEncoder* attribute), 33

`xml_tag` (*vardautomation.tooling.audio.FlacEncoder* attribute), 34

`xml_tag` (*vardautomation.tooling.audio.OpusEncoder* attribute), 32

`xml_tag` (*vardautomation.tooling.audio.PassthroughAudioEncoder* attribute), 30

`xml_tag` (*vardautomation.tooling.audio.QAACEncoder* attribute), 31

## Y

`y4m` (*vardautomation.tooling.video.FFV1* attribute), 21

`y4m` (*vardautomation.tooling.video.LosslessEncoder* attribute), 19

`y4m` (*vardautomation.tooling.video.NVEncCLossless* attribute), 20

`y4m` (*vardautomation.tooling.video.VideoEncoder* attribute), 10

`y4m` (*vardautomation.tooling.video.VideoLanEncoder* attribute), 12

`y4m` (*vardautomation.tooling.video.X264* attribute), 17

`y4m` (*vardautomation.tooling.video.X265* attribute), 14

## Z

`ZERO` (*vardautomation.tooling.audio.FlacCompressionLevel* attribute), 35