# Video Converter Documentation

***Release 1.1.0***

**Dobar Kod**

**Nov 13, 2017**

# Contents

Contents:

CHAPTER 1

# Introduction

Video Converter is a Python module for converting video files from one format and codec to another.

It uses the FFmpeg multimedia framework for actual file processing, and adds an easy-to-use API for probing and converting media files on top of it.

## 1.1 Licensing and Patents

Although FFmpeg is licensed under LGPL/GPL, Video Converter only invokes the existing ffmpeg executables on the system (ie. doesn't link to the ffmpeg libraries), so it doesn't need to be LGPL/GPL as well.

The same applies to patents. If you're in a country which recognizes software patents, it's up to you to ensure you're complying with the patent laws. Please read the FFMpeg Legal FAQ for more information.

# CHAPTER 2

# Installation and Requirements

Video Converter requires a working Python installation, and a fairly recent version of ffmpeg libraries and utilities (ffmpeg and ffprobe).

## 2.1 Video Converter installation

To build the library, run:

```
python setup.py build
```

To run automated tests:

```
python setup.py test
```

To create this documentation:

```
python setup.py doc
```

To install the library:

```
python setup.py install
```

## 2.2 Custom compiling ffmpeg

The supported formats and codecs depend on the support compiled in to ffmpeg. Many distributors choose to enable only a subset of the supported codecs, so if the version installed by your OS/distribution doesn't support a particular feature, it's advisable to recompile ffmpeg yourself.

The latest development version of FFmpeg can be downloaded from the official git repository.

To build all the codecs that Video Converter can use, you can use the following configure options:

```
./configure --prefix=${TARGET_PREFIX} \
     --extra-cflags=-I${TARGET_PREFIX}/include \
     --extra-ldflags=-L${TARGET_PREFIX}/lib \
     --enable-libmp3lame \
     --enable-libvorbis \
     --enable-libtheora \
     --enable-libx264 --enable-gpl \
     --enable-libvpx \
     --enable-libxvid
 make
```

You will need to install (either the version built by your OS distributor if it's new enough, or a custom-compiled one) the mentioned extra libraries so ffmpeg can make use of them.

# CHAPTER 3

# Tutorial

Most of the tasks (video probing, converting, thumbnail generation) can be done using the high-level Converter object.

## 3.1 Creating the Converter object

First we need to import the module and create the object:

```
>>> from converter import Converter
>>> c = Converter()
```

By default, the converter uses ffmpeg and ffprobe binaries in your path. If it should use them from another location, you can specify the paths to them in the Converter constructor.

## 3.2 Getting existing multimedia file properties

Probing the media file will return a `MediaInfo` object, containing various information about the file format, media streams, codecs and properties:

```
>>> info = c.probe('test1.ogg')
>>> info.format.format
'ogg'
>>> info.format.duration
33.00
>>> len(info.streams)
2
>>> info.video.codec
'theora'
>>> info.video.video_width
720
>>> info.video.video_height
400
```

```
>>> info.audio.codec
'vorbis'
>>> info.audio.audio_channels
2
```

A full list of properties can be found in `MediaFormatInfo` and `MediaStreamInfo` documentation.

## 3.3 Converting a video into another format

To convert a media file into some other format (or to use some other codecs), you need to create a dictionary (map) of options specifying what to convert to.

The options dictionary looks like this:

```
{
    'format': 'mkv',
    'audio': {
        'codec': 'mp3',
        'samplerate': 11025,
        'channels': 2
    },
    'video': {
        'codec': 'h264',
        'width': 720,
        'height': 400,
        'fps': 15
    },
    'subtitle': {
        'codec': 'copy'
    },
    'map': 0
}
```

The full list of options can be found in `Converter` documentation.

To prepare the conversion process:

```
>>> conv = c.convert('test1.ogg', '/tmp/output.ogg', options)
```

This won't start the conversion, it will just prepare everything and return a generator. To run the conversion process, iterate the generator until it's finished. On each iteration, the generator will yield a timecode, specifying how far into the media file is the conversion process at the moment (ie. at which second in the movie is the process).

To just drive the conversion without using the timecode information:

```
>>> for timecode in conv:
...     pass
```

## 3.4 Getting audio from a video file

To just get the audio content from a video file, you can use the conversion as above, specifying in the options that the video should be dropped:

```
{
    'format': 'mp3',
    'audio': {
        'codec': 'mp3',
        'bitrate': '22050',
        'channels': 1
    }
}
```

Since the video is not specified in the output, the video stream will be dropped. Likewise, you can drop the audio stream from the output.

If you just want to copy audio or video stream as is, without conversion, you can do that by specifying the 'copy' codec.

## 3.5 Creating a thumbnail

To create a thumbnail form a video file (from 10 seconds in the movie):

```
>>> c.thumbnail('test1.ogg', 10, '/tmp/shot.png')
```

You can specify the screenshot dimensions:

```
>>> c.thumbnail('test1.ogg', 10, '/tmp/shot.png', '320x200')
```

CHAPTER 4

# API Reference

## 4.1 Converter high-level API

**class** `converter.Converter` (*ffmpeg_path=None*, *ffprobe_path=None*)

Converter class, encapsulates formats and codecs.

```
>>> c = Converter()
```

**`convert`** (*infile*, *outfile*, *options*, *twopass=False*, *timeout=10*)

Convert media file (infile) according to specified options, and save it to outfile. For two-pass encoding, specify the pass (1 or 2) in the twopass parameter.

**Options should be passed as a dictionary. The keys are:**

- format (mandatory, string) - container format; see formats.BaseFormat for list of supported formats
- audio (optional, dict) - audio codec and options; see avcodecs.AudioCodec for list of supported options
- video (optional, dict) - video codec and options; see avcodecs.VideoCodec for list of supported options
- map (optional, int) - can be used to map all content of stream 0

Multiple audio/video streams are not supported. The output has to have at least an audio or a video stream (or both).

Convert returns a generator that needs to be iterated to drive the conversion process. The generator will periodically yield timecode of currently processed part of the file (ie. at which second in the content is the conversion process currently).

The optional timeout argument specifies how long should the operation be blocked in case ffmpeg gets stuck and doesn't report back. This doesn't limit the total conversion time, just the amount of time Converter will wait for each update from ffmpeg. As it's usually less than a second, the default of 10 is a reasonable default. To disable the timeout, set it to None. You may need to do this if using Converter in a threading environment, since the way the timeout is handled (using signals) has special restriction when using threads.

```
>>> conv = Converter().convert('test1.ogg', '/tmp/output.mkv', {
...    'format': 'mkv',
...    'audio': { 'codec': 'aac' },
...    'video': { 'codec': 'h264' }
... })
```

```
>>> for timecode in conv:
...   pass # can be used to inform the user about the progress
```

**parse_options** (*opt*, *twopass=None*)
Parse format/codec options and prepare raw ffmpeg option list.

**probe** (*fname*, *posters_as_video=True*)
Examine the media file. See the documentation of converter.FFMpeg.probe() for details.

**Parameters** **posters_as_video** – Take poster images (mainly for audio files) as A video stream, defaults to True

**thumbnail** (*fname*, *time*, *outfile*, *size=None*, *quality=4*)
Create a thumbnail of the media file. See the documentation of converter.FFMpeg.thumbnail() for details.

**thumbnails** (*fname*, *option_list*)
Create one or more thumbnail of the media file. See the documentation of converter.FFMpeg.thumbnails() for details.

## 4.2 Container formats

**class** converter.formats.**AviFormat**
Avi container format, often used vith DivX video.

**class** converter.formats.**BaseFormat**
Base format class.

Supported formats are: ogg, avi, mkv, webm, flv, mov, mp4, mpeg

**class** converter.formats.**FlvFormat**
Flash Video container format.

**class** converter.formats.**MkvFormat**
Matroska format, often used with H.264 video.

**class** converter.formats.**MovFormat**
Mov container format, used mostly with H.264 video content, often for mobile platforms.

**class** converter.formats.**Mp3Format**
Mp3 container, used audio-only mp3 files

**class** converter.formats.**Mp4Format**
Mp4 container format, the default Format for H.264 video content.

**class** converter.formats.**MpegFormat**
MPEG(TS) container, used mainly for MPEG 1/2 video codecs.

**class** converter.formats.**OggFormat**
Ogg container format, mostly used with Vorbis and Theora.

**class** `converter.formats.`**`WebmFormat`**
WebM is Google's variant of Matroska containing only VP8 for video and Vorbis for audio content.

## 4.3 Audio and video codecs

**class** `converter.avcodecs.`**`AacCodec`**
AAC audio codec.

**class** `converter.avcodecs.`**`Ac3Codec`**
AC3 audio codec.

**class** `converter.avcodecs.`**`AudioCodec`**
Base audio codec class handles general audio options. Possible parameters are:

- codec (string) - audio codec name
- channels (integer) - number of audio channels
- bitrate (integer) - stream bitrate
- samplerate (integer) - sample rate (frequency)

Supported audio codecs are: null (no audio), copy (copy from original), vorbis, aac, mp3, mp2

**class** `converter.avcodecs.`**`AudioCopyCodec`**
Copy audio stream directly from the source.

**class** `converter.avcodecs.`**`AudioNullCodec`**
Null audio codec (no audio).

**class** `converter.avcodecs.`**`BaseCodec`**
Base audio/video codec class.

**class** `converter.avcodecs.`**`DVBSub`**
DVB subtitles.

**class** `converter.avcodecs.`**`DVDSub`**
DVD subtitles.

**class** `converter.avcodecs.`**`DivxCodec`**
DivX video codec.

**class** `converter.avcodecs.`**`DtsCodec`**
DTS audio codec.

**class** `converter.avcodecs.`**`FdkAacCodec`**
AAC audio codec.

**class** `converter.avcodecs.`**`FlacCodec`**
FLAC audio codec.

**class** `converter.avcodecs.`**`FlvCodec`**
Flash Video codec.

**class** `converter.avcodecs.`**`H263Codec`**
H.263 video codec.

**class** `converter.avcodecs.`**`H264Codec`**
H.264/AVC video codec. @see http://ffmpeg.org/trac/ffmpeg/wiki/x264EncodingGuide

**class** `converter.avcodecs.`**`MOVTextCodec`**
mov_text subtitle codec.

**class** `converter.avcodecs.`**`Mp2Codec`**
MP2 (MPEG layer 2) audio codec.

**class** `converter.avcodecs.`**`Mp3Codec`**
MP3 (MPEG layer 3) audio codec.

**class** `converter.avcodecs.`**`Mpeg1Codec`**
MPEG-1 video codec.

**class** `converter.avcodecs.`**`Mpeg2Codec`**
MPEG-2 video codec.

**class** `converter.avcodecs.`**`MpegCodec`**
Base MPEG video codec.

**class** `converter.avcodecs.`**`SSA`**
SSA (SubStation Alpha) subtitle.

**class** `converter.avcodecs.`**`SubRip`**
SubRip subtitle.

**class** `converter.avcodecs.`**`SubtitleCodec`**
Base subtitle codec class handles general subtitle options. Possible parameters are:

- codec (string) - subtitle codec name (mov_text, subrib, ssa only supported currently)
- language (string) - language of subtitle stream (3 char code)
- forced (int) - force subtitles (1 true, 0 false)
- default (int) - default subtitles (1 true, 0 false)

Supported subtitle codecs are: null (no subtitle), mov_text

**class** `converter.avcodecs.`**`SubtitleCopyCodec`**
Copy subtitle stream directly from the source.

**class** `converter.avcodecs.`**`SubtitleNullCodec`**
Null video codec (no video).

**class** `converter.avcodecs.`**`TheoraCodec`**
Theora video codec. @see http://ffmpeg.org/trac/ffmpeg/wiki/TheoraVorbisEncodingGuide

**class** `converter.avcodecs.`**`VideoCodec`**
Base video codec class handles general video options. Possible parameters are:

- codec (string) - video codec name
- bitrate (string) - stream bitrate
- fps (integer) - frames per second
- width (integer) - video width
- height (integer) - video height
- **mode (string) - aspect preserval mode; one of:**
  - stretch (default) - don’t preserve aspect
  - crop - crop extra w/h
  - pad - pad with black bars
- src_width (int) - source width
- src_height (int) - source height

Aspect preserval mode is only used if both source and both destination sizes are specified. If source dimensions are not specified, aspect settings are ignored.

If source dimensions are specified, and only one of the destination dimensions is specified, the other one is calculated to preserve the aspect ratio.

Supported video codecs are: null (no video), copy (copy directly from the source), Theora, H.264/AVC, DivX, VP8, H.263, Flv, MPEG-1, MPEG-2.

**class** `converter.avcodecs.`**`VideoCopyCodec`**
Copy video stream directly from the source.

**class** `converter.avcodecs.`**`VideoNullCodec`**
Null video codec (no video).

**class** `converter.avcodecs.`**`VorbisCodec`**
Vorbis audio codec. @see http://ffmpeg.org/trac/ffmpeg/wiki/TheoraVorbisEncodingGuide

**class** `converter.avcodecs.`**`Vp8Codec`**
Google VP8 video codec.

## 4.4 Low-level ffmpeg wrapper

**class** `converter.ffmpeg.`**`FFMpeg`**(*ffmpeg_path=None*, *ffprobe_path=None*)
FFMPeg wrapper object, takes care of calling the ffmpeg binaries, passing options and parsing the output.

```
>>> f = FFMpeg()
```

**`convert`**(*infile*, *outfile*, *opts*, *timeout=10*)
Convert the source media (infile) according to specified options (a list of ffmpeg switches as strings) and save it to outfile.

Convert returns a generator that needs to be iterated to drive the conversion process. The generator will periodically yield timecode of currently processed part of the file (ie. at which second in the content is the conversion process currently).

The optional timeout argument specifies how long should the operation be blocked in case ffmpeg gets stuck and doesn't report back. See the documentation in Converter.convert() for more details about this option.

```
>>> conv = FFMpeg().convert('test.ogg', '/tmp/output.mp3',
...    ['-acodec libmp3lame', '-vn'])
>>> for timecode in conv:
...    pass # can be used to inform the user about conversion progress
```

**`probe`**(*fname*, *posters_as_video=True*)
Examine the media file and determine its format and media streams. Returns the MediaInfo object, or None if the specified file is not a valid media file.

```
>>> info = FFMpeg().probe('test1.ogg')
>>> info.format
'ogg'
>>> info.duration
33.00
>>> info.video.codec
'theora'
>>> info.video.width
720
```

```
>>> info.video.height
400
>>> info.audio.codec
'vorbis'
>>> info.audio.channels
2
:param posters_as_video: Take poster images (mainly for audio files) as
    A video stream, defaults to True
```

**thumbnail** (*fname*, *time*, *outfile*, *size=None*, *quality=4*)

Create a thumbnal of media file, and store it to outfile @param time: time point (in seconds) (float or int) @param size: Size, if specified, is WxH of the desired thumbnail.

If not specified, the video resolution is used.

**@param quality: quality of jpeg file in range 2(best)-31(worst)** recommended range: 2-6

```
>>> FFMpeg().thumbnail('test1.ogg', 5, '/tmp/shot.png', '320x240')
```

**thumbnails** (*fname*, *option_list*)

Create one or more thumbnails of video. @param option_list: a list of tuples like:

(time, outfile, size=None, quality=DEFAULT_JPEG_QUALITY) see documentation of *converter.FFMpeg.thumbnail()* for details.

```
>>> FFMpeg().thumbnails('test1.ogg', [(5, '/tmp/shot.png', '320x240'),
>>>                                   (10, '/tmp/shot2.png', None, 5)])
```

**class** converter.ffmpeg.**MediaFormatInfo**

**Describes the media container format. The attributes are:**

- format - format (short) name (eg. “ogg”)
- fullname - format full (descriptive) name
- bitrate - total bitrate (bps)
- duration - media duration in seconds
- filesize - file size

**parse_ffprobe** (*key*, *val*)

Parse raw ffprobe output (key=value).

**class** converter.ffmpeg.**MediaInfo** (*posters_as_video=True*)

Information about media object, as parsed by ffprobe. The attributes are:

- format - a MediaFormatInfo object
- streams - a list of MediaStreamInfo objects

**audio**

First audio stream, or None if there are no audio streams.

**parse_ffprobe** (*raw*)

Parse raw ffprobe output.

**video**

First video stream, or None if there are no video streams.

**class** `converter.ffmpeg.`**`MediaStreamInfo`**

Describes one stream inside a media file. The general attributes are:

- index - stream index inside the container (0-based)
- type - stream type, either 'audio' or 'video'
- codec - codec (short) name (e.g "vorbis", "theora")
- codec_desc - codec full (descriptive) name
- duration - stream duration in seconds
- metadata - optional metadata associated with a video or audio stream
- bitrate - stream bitrate in bytes/second
- attached_pic - (0, 1 or None) is stream a poster image? (e.g. in mp3)

**Video-specific attributes are:**

- video_width - width of video in pixels
- video_height - height of video in pixels
- video_fps - average frames per second

**Audio-specific attributes are:**

- audio_channels - the number of channels in the stream
- audio_samplerate - sample rate (Hz)

**`parse_ffprobe`**(*key*, *val*)

Parse raw ffprobe output (key=value).

# CHAPTER 5

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

# Index