

# ffmpeg Documentation

## Table of Contents

- 1 Synopsis
- 2 Description
- 3 Detailed description
  - 3.1 Filtering
    - 3.1.1 Simple filtergraphs
    - 3.1.2 Complex filtergraphs
  - 3.2 Stream copy
- 4 Stream selection
- 5 Options
  - 5.1 Stream specifiers
  - 5.2 Generic options
  - 5.3 AVOptions
  - 5.4 Main options
  - 5.5 Video Options
  - 5.6 Advanced Video options
  - 5.7 Audio Options
  - 5.8 Advanced Audio options
  - 5.9 Subtitle options
  - 5.10 Advanced Subtitle options
  - 5.11 Advanced options
  - 5.12 Preset files
    - 5.12.1 ffpreset files
    - 5.12.2 avpreset files
- 6 Examples
  - 6.1 Video and Audio grabbing
  - 6.2 X11 grabbing
  - 6.3 Video and Audio file format conversion
- 7 Syntax
  - 7.1 Quoting and escaping
    - 7.1.1 Examples
  - 7.2 Date
  - 7.3 Time duration
    - 7.3.1 Examples
  - 7.4 Video size
  - 7.5 Video rate
  - 7.6 Ratio
  - 7.7 Color
  - 7.8 Channel Layout
- 8 Expression Evaluation

- 9 OpenCL Options
- 10 Codec Options
- 11 Decoders
- 12 Video Decoders
  - 12.1 hevc
  - 12.2 rawvideo
    - 12.2.1 Options
- 13 Audio Decoders
  - 13.1 ac3
    - 13.1.1 AC-3 Decoder Options
  - 13.2 flac
    - 13.2.1 FLAC Decoder options
  - 13.3 ffwavesynth
  - 13.4 libcelt
  - 13.5 libgsm
  - 13.6 libilbc
    - 13.6.1 Options
  - 13.7 libopencore-amrnb
  - 13.8 libopencore-amrwb
  - 13.9 libopus
- 14 Subtitles Decoders
  - 14.1 dvbsub
    - 14.1.1 Options
  - 14.2 dvdsup
    - 14.2.1 Options
  - 14.3 libzybi-teletext
    - 14.3.1 Options
- 15 Encoders
- 16 Audio Encoders
  - 16.1 aac
    - 16.1.1 Options
  - 16.2 ac3 and ac3\_fixed
    - 16.2.1 AC-3 Metadata
      - 16.2.1.1 Metadata Control Options
      - 16.2.1.2 Downmix Levels
      - 16.2.1.3 Audio Production Information
      - 16.2.1.4 Other Metadata Options
    - 16.2.2 Extended Bitstream Information
      - 16.2.2.1 Extended Bitstream Information - Part 1
      - 16.2.2.2 Extended Bitstream Information - Part 2
    - 16.2.3 Other AC-3 Encoding Options
    - 16.2.4 Floating-Point-Only AC-3 Encoding Options
  - 16.3 flac

- 16.3.1 Options
- 16.4 libfaac
  - 16.4.1 Options
  - 16.4.2 Examples
- 16.5 libfdk\_aac
  - 16.5.1 Options
  - 16.5.2 Examples
- 16.6 libmp3lame
  - 16.6.1 Options
- 16.7 libopencore-amrnb
  - 16.7.1 Options
- 16.8 libshine
  - 16.8.1 Options
- 16.9 libtwolame
  - 16.9.1 Options
- 16.10 libvo-aacenc
  - 16.10.1 Options
- 16.11 libvo-amrwbenc
  - 16.11.1 Options
- 16.12 libopus
  - 16.12.1 Option Mapping
- 16.13 libvorbis
  - 16.13.1 Options
- 16.14 libwavpack
  - 16.14.1 Options
- 16.15 wavpack
  - 16.15.1 Options
    - 16.15.1.1 Shared options
    - 16.15.1.2 Private options
- 17 Video Encoders
  - 17.1 jpeg2000
    - 17.1.1 Options
  - 17.2 snow
    - 17.2.1 Options
  - 17.3 libtheora
    - 17.3.1 Options
    - 17.3.2 Examples
  - 17.4 libvpx
    - 17.4.1 Options
  - 17.5 libwebp
    - 17.5.1 Pixel Format
    - 17.5.2 Options
  - 17.6 libx264, libx264rgb

- 17.6.1 Supported Pixel Formats
  - 17.6.2 Options
- 17.7 libx265
  - 17.7.1 Options
- 17.8 libxvid
  - 17.8.1 Options
- 17.9 mpeg2
  - 17.9.1 Options
- 17.10 png
  - 17.10.1 Private options
- 17.11 ProRes
  - 17.11.1 Private Options for prores-ks
  - 17.11.2 Speed considerations
- 17.12 libkvazaar
  - 17.12.1 Options
- 18 Subtitles Encoders
  - 18.1 dvdsub
    - 18.1.1 Options
- 19 Bitstream Filters
  - 19.1 aac\_adtstoasc
  - 19.2 chomp
  - 19.3 dump\_extra
  - 19.4 h264\_mp4toannexb
  - 19.5 imxdump
  - 19.6 mjpeg2jpeg
  - 19.7 mjpega\_dump\_header
  - 19.8 movsub
  - 19.9 mp3\_header\_decompress
  - 19.10 mpeg4\_unpack\_bframes
  - 19.11 noise
  - 19.12 remove\_extra
- 20 Format Options
  - 20.1 Format stream specifiers
- 21 Demuxers
  - 21.1 aa
  - 21.2 applehttp
  - 21.3 apng
  - 21.4 asf
  - 21.5 concat
    - 21.5.1 Syntax
    - 21.5.2 Options
  - 21.6 flv
  - 21.7 libgme

- 21.8 libquvi
- 21.9 gif
- 21.10 image2
  - 21.10.1 Examples
- 21.11 mpegts
- 21.12 rawvideo
- 21.13 sbg
- 21.14 tedcaptions
- 22 Muxers
  - 22.1 aiff
    - 22.1.1 Options
  - 22.2 crc
    - 22.2.1 Examples
  - 22.3 framecrc
    - 22.3.1 Examples
  - 22.4 framemd5
    - 22.4.1 Examples
  - 22.5 gif
  - 22.6 hls
    - 22.6.1 Options
  - 22.7 ico
  - 22.8 image2
    - 22.8.1 Examples
    - 22.8.2 Options
  - 22.9 matroska
    - 22.9.1 Metadata
    - 22.9.2 Options
  - 22.10 md5
  - 22.11 mov, mp4, ismv
    - 22.11.1 Options
    - 22.11.2 Example
    - 22.11.3 Audible AAX
  - 22.12 mp3
  - 22.13 mpegts
    - 22.13.1 Options
    - 22.13.2 Example
  - 22.14 null
  - 22.15 nut
  - 22.16 ogg
  - 22.17 segment, stream\_segment, ssegment
    - 22.17.1 Options
    - 22.17.2 Examples
  - 22.18 smoothstreaming

- 22.19 tee
  - 22.19.1 Examples
- 22.20 webm\_dash\_manifest
  - 22.20.1 Options
  - 22.20.2 Example
- 22.21 webm\_chunk
  - 22.21.1 Options
  - 22.21.2 Example
- 23 Metadata
- 24 Protocols
  - 24.1 async
  - 24.2 bluray
  - 24.3 cache
  - 24.4 concat
  - 24.5 crypto
  - 24.6 data
  - 24.7 file
  - 24.8 ftp
  - 24.9 gopher
  - 24.10 hls
  - 24.11 http
    - 24.11.1 HTTP Cookies
  - 24.12 Icecast
  - 24.13 mmst
  - 24.14 mmsh
  - 24.15 md5
  - 24.16 pipe
  - 24.17 rtmp
  - 24.18 rtmpe
  - 24.19 rtmps
  - 24.20 rtmpt
  - 24.21 rtmpte
  - 24.22 rtmpts
  - 24.23 libsmbclient
  - 24.24 libssh
  - 24.25 librtmp rtmp, rtmpe, rtmps, rtmpt, rtmpte
  - 24.26 rtp
  - 24.27 rtsp
    - 24.27.1 Examples
  - 24.28 sap
    - 24.28.1 Muxer
    - 24.28.2 Demuxer
  - 24.29 sctp

- 24.30 srtp
- 24.31 subfile
- 24.32 tcp
- 24.33 tls
- 24.34 udp
  - 24.34.1 Examples
- 24.35 unix
- 25 Device Options
- 26 Input Devices
  - 26.1 alsa
    - 26.1.1 Options
  - 26.2 avfoundation
    - 26.2.1 Options
    - 26.2.2 Examples
  - 26.3 bktr
    - 26.3.1 Options
  - 26.4 decklink
    - 26.4.1 Options
    - 26.4.2 Examples
  - 26.5 dshow
    - 26.5.1 Options
    - 26.5.2 Examples
  - 26.6 dv1394
    - 26.6.1 Options
  - 26.7 fbdev
    - 26.7.1 Options
  - 26.8 gdigrab
    - 26.8.1 Options
  - 26.9 iec61883
    - 26.9.1 Options
    - 26.9.2 Examples
  - 26.10 jack
    - 26.10.1 Options
  - 26.11 lavfi
    - 26.11.1 Options
    - 26.11.2 Examples
  - 26.12 libcdio
    - 26.12.1 Options
  - 26.13 libdc1394
  - 26.14 openal
    - 26.14.1 Options
    - 26.14.2 Examples
  - 26.15 oss

- 26.15.1 Options
- 26.16 pulse
  - 26.16.1 Options
  - 26.16.2 Examples
- 26.17 qtkit
  - 26.17.1 Options
- 26.18 sndio
  - 26.18.1 Options
- 26.19 video4linux2, v4l2
  - 26.19.1 Options
- 26.20 vfwcap
  - 26.20.1 Options
- 26.21 x11grab
  - 26.21.1 Options
  - 26.21.2 *grab\_x grab\_y* AVOption
- 27 Output Devices
  - 27.1 alsa
    - 27.1.1 Examples
  - 27.2 caca
    - 27.2.1 Options
    - 27.2.2 Examples
  - 27.3 decklink
    - 27.3.1 Options
    - 27.3.2 Examples
  - 27.4 fbdev
    - 27.4.1 Options
    - 27.4.2 Examples
  - 27.5 opengl
    - 27.5.1 Options
    - 27.5.2 Examples
  - 27.6 oss
  - 27.7 pulse
    - 27.7.1 Options
    - 27.7.2 Examples
  - 27.8 sdl
    - 27.8.1 Options
    - 27.8.2 Interactive commands
    - 27.8.3 Examples
  - 27.9 sndio
  - 27.10 xv
    - 27.10.1 Options
    - 27.10.2 Examples
- 28 Resampler Options



- 29 Scaler Options
- 30 Filtering Introduction
- 31 graph2dot
- 32 Filtergraph description
  - 32.1 Filtergraph syntax
  - 32.2 Notes on filtergraph escaping
- 33 Timeline editing
- 34 Audio Filters
  - 34.1 acrossfade
    - 34.1.1 Examples
  - 34.2 adelay
    - 34.2.1 Examples
  - 34.3 aecho
    - 34.3.1 Examples
  - 34.4 aeval
    - 34.4.1 Examples
  - 34.5 afade
    - 34.5.1 Examples
  - 34.6 aformat
  - 34.7 allpass
  - 34.8 amerge
    - 34.8.1 Examples
  - 34.9 amix
  - 34.10 anull
  - 34.11 apad
    - 34.11.1 Examples
  - 34.12 aphasor
  - 34.13 aresample
    - 34.13.1 Examples
  - 34.14 asetnsamples
  - 34.15 asetrate
  - 34.16 ashowinfo
  - 34.17 astats
  - 34.18 astreamsync
    - 34.18.1 Examples
  - 34.19 asyncts
  - 34.20 atempo
    - 34.20.1 Examples
  - 34.21 atrim
  - 34.22 bandpass
  - 34.23 bandreject
  - 34.24 bass
  - 34.25 biquad

- 34.26 bs2b
- 34.27 channelmap
- 34.28 channelsplit
- 34.29 chorus
  - 34.29.1 Examples
- 34.30 compand
  - 34.30.1 Examples
- 34.31 dcsift
- 34.32 dynaudnorm
- 34.33 earwax
- 34.34 equalizer
  - 34.34.1 Examples
- 34.35 flanger
- 34.36 highpass
- 34.37 join
- 34.38 ladspa
  - 34.38.1 Examples
  - 34.38.2 Commands
- 34.39 lowpass
- 34.40 pan
  - 34.40.1 Mixing examples
  - 34.40.2 Remapping examples
- 34.41 replaygain
- 34.42 resample
- 34.43 sidechaincompress
  - 34.43.1 Examples
- 34.44 silencedetect
  - 34.44.1 Examples
- 34.45 silenceremove
  - 34.45.1 Examples
- 34.46 treble
- 34.47 volume
  - 34.47.1 Commands
  - 34.47.2 Examples
- 34.48 volumedetect
  - 34.48.1 Examples
- 35 Audio Sources
  - 35.1 abuffer
    - 35.1.1 Examples
  - 35.2 aevalsrc
    - 35.2.1 Examples
  - 35.3 anullsrc
    - 35.3.1 Examples

- 35.4 flite
  - 35.4.1 Examples
- 35.5 sine
  - 35.5.1 Examples
- 36 Audio Sinks
  - 36.1 abuffersink
  - 36.2 anullsink
- 37 Video Filters
  - 37.1 alphaextract
  - 37.2 alphamerge
  - 37.3 ass
  - 37.4 atadenoise
  - 37.5 bbox
  - 37.6 blackdetect
  - 37.7 blackframe
  - 37.8 blend, tblend
    - 37.8.1 Examples
  - 37.9 boxblur
    - 37.9.1 Examples
  - 37.10 codecview
    - 37.10.1 Examples
  - 37.11 colorbalance
    - 37.11.1 Examples
  - 37.12 colorkey
    - 37.12.1 Examples
  - 37.13 colorlevels
    - 37.13.1 Examples
  - 37.14 colorchannelmixer
    - 37.14.1 Examples
  - 37.15 colormatrix
  - 37.16 copy
  - 37.17 crop
    - 37.17.1 Examples
    - 37.17.2 Commands
  - 37.18 cropdetect
  - 37.19 curves
    - 37.19.1 Examples
  - 37.20 dctdnoiz
    - 37.20.1 Examples
  - 37.21 deband
  - 37.22 decimate
  - 37.23 deflate
  - 37.24 dejudder

- 37.25 delogo
  - 37.25.1 Examples
- 37.26 deshake
- 37.27 detelecine
- 37.28 dilation
- 37.29 drawbox
  - 37.29.1 Examples
- 37.30 drawgraph, adrawgraph
- 37.31 drawgrid
  - 37.31.1 Examples
- 37.32 drawtext
  - 37.32.1 Syntax
  - 37.32.2 Text expansion
  - 37.32.3 Examples
- 37.33 edgedetect
  - 37.33.1 Examples
- 37.34 eq
  - 37.34.1 Commands
- 37.35 erosion
- 37.36 extractplanes
  - 37.36.1 Examples
- 37.37 elbg
- 37.38 fade
  - 37.38.1 Examples
- 37.39 fftfilt
  - 37.39.1 Examples
- 37.40 field
- 37.41 fieldmatch
  - 37.41.1 p/c/n/u/b meaning
    - 37.41.1.1 p/c/n
    - 37.41.1.2 u/b
  - 37.41.2 Examples
- 37.42 fieldorder
- 37.43 fifo
- 37.44 find\_rect
  - 37.44.1 Examples
- 37.45 cover\_rect
  - 37.45.1 Examples
- 37.46 format
  - 37.46.1 Examples
- 37.47 fps
  - 37.47.1 Examples
- 37.48 framepack

- 37.49 framerate
- 37.50 framestep
- 37.51 frei0r
  - 37.51.1 Examples
- 37.52 fspp
- 37.53 geq
  - 37.53.1 Examples
- 37.54 gradfun
  - 37.54.1 Examples
- 37.55 haldclut
  - 37.55.1 Workflow examples
    - 37.55.1.1 Hald CLUT video stream
    - 37.55.1.2 Hald CLUT with preview
- 37.56 hflip
- 37.57 histeq
- 37.58 histogram
  - 37.58.1 Examples
- 37.59 hqdn3d
- 37.60 hqx
- 37.61 hstack
- 37.62 hue
  - 37.62.1 Examples
  - 37.62.2 Commands
- 37.63 idet
- 37.64 il
- 37.65 inflate
- 37.66 interlace
- 37.67 kerndeint
  - 37.67.1 Examples
- 37.68 lenscorrection
  - 37.68.1 Options
- 37.69 lut3d
- 37.70 lut, lutrgb, lutyuv
  - 37.70.1 Examples
- 37.71 mergeplanes
  - 37.71.1 Examples
- 37.72 mcdeint
- 37.73 mpdecimate
- 37.74 negate
- 37.75 noformat
  - 37.75.1 Examples
- 37.76 noise
  - 37.76.1 Examples

- 37.77 null
- 37.78 ocv
  - 37.78.1 dilate
  - 37.78.2 erode
  - 37.78.3 smooth
- 37.79 overlay
  - 37.79.1 Commands
  - 37.79.2 Examples
- 37.80 owdenoise
- 37.81 pad
  - 37.81.1 Examples
- 37.82 paletteen
- 37.82.1 Examples
- 37.83 paletteuse
  - 37.83.1 Examples
- 37.84 perspective
- 37.85 phase
- 37.86 pixdesctest
- 37.87 pp
  - 37.87.1 Examples
- 37.88 pp7
- 37.89 psnr
- 37.90 pullup
- 37.91 qp
  - 37.91.1 Examples
- 37.92 random
- 37.93 removegrain
- 37.94 removelogo
- 37.95 repeatfields
- 37.96 reverse, areverse
  - 37.96.1 Examples
- 37.97 rotate
  - 37.97.1 Examples
  - 37.97.2 Commands
- 37.98 sab
- 37.99 scale
  - 37.99.1 Options
  - 37.99.2 Examples
  - 37.99.3 Commands
- 37.100 scale2ref
  - 37.100.1 Examples
- 37.101 separatefields
- 37.102 setdar, setsar

- 37.102.1 Examples
- 37.103 setfield
- 37.104 showinfo
- 37.105 showpalette
- 37.106 shuffleplanes
- 37.107 signalstats
  - 37.107.1 Examples
- 37.108 smartblur
- 37.109 ssim
- 37.110 stereo3d
  - 37.110.1 Examples
- 37.111 spp
- 37.112 subtitles
- 37.113 super2xsai
- 37.114 swapuv
- 37.115 telecine
- 37.116 thumbnail
  - 37.116.1 Examples
- 37.117 tile
  - 37.117.1 Examples
- 37.118 tinterlace
- 37.119 transpose
- 37.120 trim
- 37.121 unsharp
  - 37.121.1 Examples
- 37.122 uspp
- 37.123 vectorscope
- 37.124 vidstabdetect
  - 37.124.1 Examples
- 37.125 vidstabtransform
  - 37.125.1 Options
  - 37.125.2 Examples
- 37.126 vflip
- 37.127 vignette
  - 37.127.1 Expressions
  - 37.127.2 Examples
- 37.128 vstack
- 37.129 w3dif
- 37.130 waveform
- 37.131 xbr
- 37.132 yadif
- 37.133 zoompan
  - 37.133.1 Examples

- 38 Video Sources
  - 38.1 buffer
  - 38.2 cellauto
    - 38.2.1 Examples
  - 38.3 mandelbrot
  - 38.4 mptestsrc
  - 38.5 frei0r\_src
  - 38.6 life
    - 38.6.1 Examples
  - 38.7 allrgb, allyuv, color, haldclutsrc, nullsrc, rgbtestsrc, smptebars, smptehtbars, testsrc
    - 38.7.1 Commands
- 39 Video Sinks
  - 39.1 buffersink
  - 39.2 nullsink
- 40 Multimedia Filters
  - 40.1 aphasemeter
  - 40.2 avectorscope
    - 40.2.1 Examples
  - 40.3 concat
    - 40.3.1 Examples
  - 40.4 ebur128
    - 40.4.1 Examples
  - 40.5 interleave, ainterleave
    - 40.5.1 Examples
  - 40.6 perms, aperms
  - 40.7 select, aselect
    - 40.7.1 Examples
  - 40.8 sendcmd, asendcmd
    - 40.8.1 Commands syntax
    - 40.8.2 Examples
  - 40.9 setpts, asetpts
    - 40.9.1 Examples
  - 40.10 settb, asettb
    - 40.10.1 Examples
  - 40.11 showcqt
    - 40.11.1 Examples
  - 40.12 showfreqs
  - 40.13 showspectrum
    - 40.13.1 Examples
  - 40.14 showvolume
  - 40.15 showwaves
    - 40.15.1 Examples
  - 40.16 showwavespic



- 40.16.1 Examples
- 40.17 split, asplit
  - 40.17.1 Examples
- 40.18 zmq, azmq
  - 40.18.1 Examples
- 41 Multimedia Sources
  - 41.1 amovie
  - 41.2 movie
    - 41.2.1 Examples
- 42 See Also
- 43 Authors

## 1 Synopsis# TOC

```
ffmpeg [global_options] {[input_file_options] -i input_file} ... {[output_file_options]
output_file} ...
```

## 2 Description# TOC

`ffmpeg` is a very fast video and audio converter that can also grab from a live audio/video source. It can also convert between arbitrary sample rates and resize video on the fly with a high quality polyphase filter.

`ffmpeg` reads from an arbitrary number of input "files" (which can be regular files, pipes, network streams, grabbing devices, etc.), specified by the `-i` option, and writes to an arbitrary number of output "files", which are specified by a plain output filename. Anything found on the command line which cannot be interpreted as an option is considered to be an output filename.

Each input or output file can, in principle, contain any number of streams of different types (video/audio/subtitle/attachment/data). The allowed number and/or types of streams may be limited by the container format. Selecting which streams from which inputs will go into which output is either done automatically or with the `-map` option (see the Stream selection chapter).

To refer to input files in options, you must use their indices (0-based). E.g. the first input file is 0, the second is 1, etc. Similarly, streams within a file are referred to by their indices. E.g. `2 : 3` refers to the fourth stream in the third input file. Also see the Stream specifiers chapter.

As a general rule, options are applied to the next specified file. Therefore, order is important, and you can have the same option on the command line multiple times. Each occurrence is then applied to the next input or output file. Exceptions from this rule are the global options (e.g. verbosity level), which should be specified first.

Do not mix input and output files – first specify all input files, then all output files. Also do not mix options which belong to different files. All options apply **ONLY** to the next input or output file and are reset between files.

- To set the video bitrate of the output file to 64 kbit/s:

```
ffmpeg -i input.avi -b:v 64k -bufsize 64k output.avi
```

- To force the frame rate of the output file to 24 fps:

```
ffmpeg -i input.avi -r 24 output.avi
```

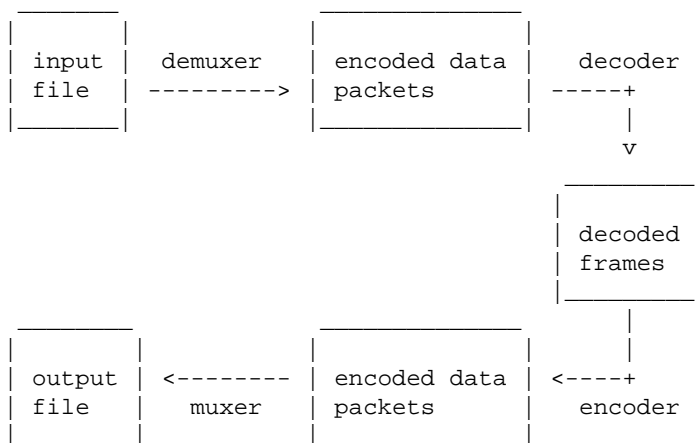
- To force the frame rate of the input file (valid for raw formats only) to 1 fps and the frame rate of the output file to 24 fps:

```
ffmpeg -r 1 -i input.m2v -r 24 output.avi
```

The format option may be needed for raw input files.

### 3 Detailed description# TOC

The transcoding process in `ffmpeg` for each output can be described by the following diagram:



`ffmpeg` calls the `libavformat` library (containing demuxers) to read input files and get packets containing encoded data from them. When there are multiple input files, `ffmpeg` tries to keep them synchronized by tracking lowest timestamp on any active input stream.

Encoded packets are then passed to the decoder (unless `streamcopy` is selected for the stream, see further for a description). The decoder produces uncompressed frames (raw video/PCM audio/...) which can be processed further by filtering (see next section). After filtering, the frames are passed to the encoder, which encodes them and outputs encoded packets. Finally those are passed to the muxer, which writes the encoded packets to the output file.

## 3.1 Filtering# TOC

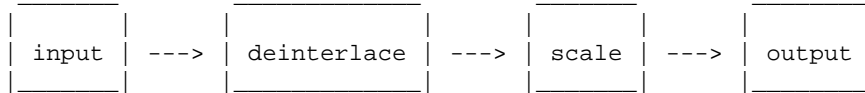
Before encoding, `ffmpeg` can process raw audio and video frames using filters from the `libavfilter` library. Several chained filters form a filter graph. `ffmpeg` distinguishes between two types of filtergraphs: simple and complex.

### 3.1.1 Simple filtergraphs# TOC

Simple filtergraphs are those that have exactly one input and output, both of the same type. In the above diagram they can be represented by simply inserting an additional step between decoding and encoding:



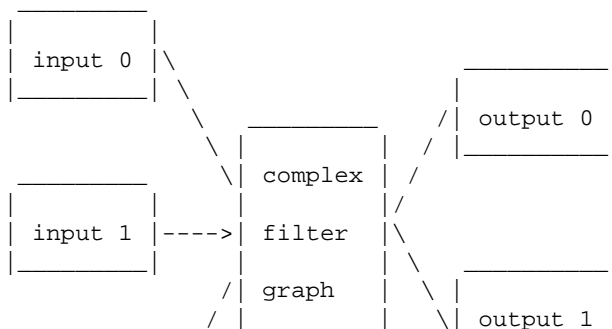
Simple filtergraphs are configured with the per-stream `-filter` option (with `-vf` and `-af` aliases for video and audio respectively). A simple filtergraph for video can look for example like this:



Note that some filters change frame properties but not frame contents. E.g. the `fps` filter in the example above changes number of frames, but does not touch the frame contents. Another example is the `setpts` filter, which only sets timestamps and otherwise passes the frames unchanged.

### 3.1.2 Complex filtergraphs# TOC

Complex filtergraphs are those which cannot be described as simply a linear processing chain applied to one stream. This is the case, for example, when the graph has more than one input and/or output, or when output stream type is different from input. They can be represented with the following diagram:





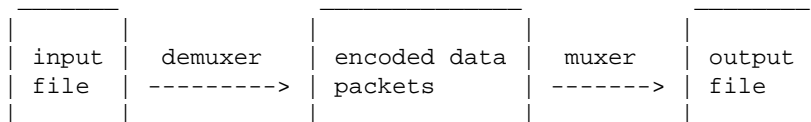
Complex filtergraphs are configured with the `-filter_complex` option. Note that this option is global, since a complex filtergraph, by its nature, cannot be unambiguously associated with a single stream or file.

The `-lavfi` option is equivalent to `-filter_complex`.

A trivial example of a complex filtergraph is the `overlay` filter, which has two video inputs and one video output, containing one video overlaid on top of the other. Its audio counterpart is the `amix` filter.

## 3.2 Stream copy# TOC

Stream copy is a mode selected by supplying the `copy` parameter to the `-codec` option. It makes `ffmpeg` omit the decoding and encoding step for the specified stream, so it does only demuxing and muxing. It is useful for changing the container format or modifying container-level metadata. The diagram above will, in this case, simplify to this:



Since there is no decoding or encoding, it is very fast and there is no quality loss. However, it might not work in some cases because of many factors. Applying filters is obviously also impossible, since filters work on uncompressed data.

## 4 Stream selection# TOC

By default, `ffmpeg` includes only one stream of each type (video, audio, subtitle) present in the input files and adds them to each output file. It picks the "best" of each based upon the following criteria: for video, it is the stream with the highest resolution, for audio, it is the stream with the most channels, for subtitles, it is the first subtitle stream. In the case where several streams of the same type rate equally, the stream with the lowest index is chosen.

You can disable some of those defaults by using the `-vn`/`-an`/`-sn` options. For full manual control, use the `-map` option, which disables the defaults just described.

## 5 Options# TOC

All the numerical options, if not specified otherwise, accept a string representing a number as input, which may be followed by one of the SI unit prefixes, for example: 'K', 'M', or 'G'.

If 'i' is appended to the SI unit prefix, the complete prefix will be interpreted as a unit prefix for binary multiples, which are based on powers of 1024 instead of powers of 1000. Appending 'B' to the SI unit prefix multiplies the value by 8. This allows using, for example: 'KB', 'MiB', 'G' and 'B' as number suffixes.

Options which do not take arguments are boolean options, and set the corresponding value to true. They can be set to false by prefixing the option name with "no". For example using "-nofoo" will set the boolean option with name "foo" to false.

## 5.1 Stream specifiers# TOC

Some options are applied per-stream, e.g. bitrate or codec. Stream specifiers are used to precisely specify which stream(s) a given option belongs to.

A stream specifier is a string generally appended to the option name and separated from it by a colon. E.g. `-codec:a:1 ac3` contains the `a:1` stream specifier, which matches the second audio stream. Therefore, it would select the ac3 codec for the second audio stream.

A stream specifier can match several streams, so that the option is applied to all of them. E.g. the stream specifier in `-b:a 128k` matches all audio streams.

An empty stream specifier matches all streams. For example, `-codec copy` or `-codec: copy` would copy all the streams without reencoding.

Possible forms of stream specifiers are:

*stream\_index*

Matches the stream with this index. E.g. `-threads:1 4` would set the thread count for the second stream to 4.

*stream\_type[:stream\_index]*

*stream\_type* is one of following: 'v' or 'V' for video, 'a' for audio, 's' for subtitle, 'd' for data, and 't' for attachments. 'v' matches all video streams, 'V' only matches video streams which are not attached pictures, video thumbnails or cover arts. If *stream\_index* is given, then it matches stream number *stream\_index* of this type. Otherwise, it matches all streams of this type.

*p:program\_id[:stream\_index]*

If *stream\_index* is given, then it matches the stream with number *stream\_index* in the program with the id *program\_id*. Otherwise, it matches all streams in the program.

*#stream\_id* or *i:stream\_id*

Match the stream by stream id (e.g. PID in MPEG-TS container).

`m:key[:value]`

Matches streams with the metadata tag *key* having the specified value. If *value* is not given, matches streams that contain the given tag with any value.

`u`

Matches streams with usable configuration, the codec must be defined and the essential information such as video dimension or audio sample rate must be present.

Note that in `ffmpeg`, matching by metadata will only work properly for input files.

## 5.2 Generic options# TOC

These options are shared amongst the `ff*` tools.

`-L`

Show license.

`-h, -?, -help, --help [arg]`

Show help. An optional parameter may be specified to print help about a specific item. If no argument is specified, only basic (non advanced) tool options are shown.

Possible values of *arg* are:

`long`

Print advanced tool options in addition to the basic tool options.

`full`

Print complete list of options, including shared and private options for encoders, decoders, demuxers, muxers, filters, etc.

`decoder=decoder_name`

Print detailed information about the decoder named *decoder\_name*. Use the `-decoders` option to get a list of all decoders.

`encoder=encoder_name`

Print detailed information about the encoder named *encoder\_name*. Use the `-encoders` option to get a list of all encoders.

`demuxer=demuxer_name`

Print detailed information about the demuxer named *demuxer\_name*. Use the `-formats` option to get a list of all demuxers and muxers.

`muxer=muxer_name`

Print detailed information about the muxer named *muxer\_name*. Use the `-formats` option to get a list of all muxers and demuxers.

`filter=filter_name`

Print detailed information about the filter name *filter\_name*. Use the `-filters` option to get a list of all filters.

`-version`

Show version.

`-formats`

Show available formats (including devices).

`-devices`

Show available devices.

`-codecs`

Show all codecs known to libavcodec.

Note that the term 'codec' is used throughout this documentation as a shortcut for what is more correctly called a media bitstream format.

`-decoders`

Show available decoders.

`-encoders`

Show all available encoders.

`-bsfs`

Show available bitstream filters.

`-protocols`

Show available protocols.

`-filters`

Show available libavfilter filters.

`-pix_fmts`

Show available pixel formats.

`-sample_fmts`

Show available sample formats.

`-layouts`

Show channel names and standard channel layouts.

`-colors`

Show recognized color names.

`-sources device[,opt1=val1[,opt2=val2]...]`

Show autodetected sources of the input device. Some devices may provide system-dependent source names that cannot be autodetected. The returned list cannot be assumed to be always complete.

```
ffmpeg -sources pulse,server=192.168.0.4
```

`-sinks device[,opt1=val1[,opt2=val2]...]`

Show autodetected sinks of the output device. Some devices may provide system-dependent sink names that cannot be autodetected. The returned list cannot be assumed to be always complete.

```
ffmpeg -sinks pulse,server=192.168.0.4
```

`-loglevel [repeat+]loglevel | -v [repeat+]loglevel`

Set the logging level used by the library. Adding "repeat+" indicates that repeated log output should not be compressed to the first line and the "Last message repeated n times" line will be omitted. "repeat" can also be used alone. If "repeat" is used alone, and with no prior loglevel set, the default loglevel will be used. If multiple loglevel parameters are given, using 'repeat' will not change the loglevel. *loglevel* is a string or a number containing one of the following values:

`'quiet, -8'`

Show nothing at all; be silent.

`'panic, 0'`



Only show fatal errors which could lead the process to crash, such as and assert failure. This is not currently used for anything.

`'fatal, 8'`

Only show fatal errors. These are errors after which the process absolutely cannot continue after.

`'error, 16'`

Show all errors, including ones which can be recovered from.

`'warning, 24'`

Show all warnings and errors. Any message related to possibly incorrect or unexpected events will be shown.

`'info, 32'`

Show informative messages during processing. This is in addition to warnings and errors. This is the default value.

`'verbose, 40'`

Same as `info`, except more verbose.

`'debug, 48'`

Show everything, including debugging information.

`'trace, 56'`

By default the program logs to `stderr`, if coloring is supported by the terminal, colors are used to mark errors and warnings. Log coloring can be disabled setting the environment variable `AV_LOG_FORCE_NOCOLOR` or `NO_COLOR`, or can be forced setting the environment variable `AV_LOG_FORCE_COLOR`. The use of the environment variable `NO_COLOR` is deprecated and will be dropped in a following FFmpeg version.

`-report`

Dump full command line and console output to a file named `program-YYYYMMDD-HHMMSS.log` in the current directory. This file can be useful for bug reports. It also implies `-loglevel verbose`.

Setting the environment variable `FFREPORT` to any value has the same effect. If the value is a `':'`-separated key=value sequence, these options will affect the report; option values must be escaped if they contain special characters or the options delimiter `':'` (see the “Quoting and escaping” section in the `ffmpeg-utils` manual).

The following options are recognized:

`file`

set the file name to use for the report; `%p` is expanded to the name of the program, `%t` is expanded to a timestamp, `%%` is expanded to a plain `%`

`level`

set the log verbosity level using a numerical value (see `-loglevel`).

For example, to output a report to a file named `ffreport.log` using a log level of 32 (alias for log level info):

```
FFREPORT=file=ffreport.log:level=32 ffmpeg -i input output
```

Errors in parsing the environment variable are not fatal, and will not appear in the report.

`-hide_banner`

Suppress printing banner.

All FFmpeg tools will normally show a copyright notice, build options and library versions. This option can be used to suppress printing this information.

`-cpuflags flags (global)`

Allows setting and clearing cpu flags. This option is intended for testing. Do not use it unless you know what you're doing.

```
ffmpeg -cpuflags -sse+mmx ...
ffmpeg -cpuflags mmx ...
ffmpeg -cpuflags 0 ...
```

Possible flags for this option are:

```
'x86'
  'mmx'
  'mmxext'
  'sse'
  'sse2'
  'sse2slow'
  'sse3'
  'sse3slow'
  'ssse3'
  'atom'
  'sse4.1'
  'sse4.2'
```

```

    'avx'
    'avx2'
    'xop'
    'fma3'
    'fma4'
    '3dnow'
    '3dnowext'
    'bmi1'
    'bmi2'
    'cmov'
'ARM'
    'armv5te'
    'armv6'
    'armv6t2'
    'vfp'
    'vfpv3'
    'neon'
    'setend'
'AArch64'
    'armv8'
    'vfp'
    'neon'
'PowerPC'
    'altivec'
'Specific Processors'
    'pentium2'
    'pentium3'
    'pentium4'
    'k6'
    'k62'
    'athlon'
    'athlonxp'
    'k8'
-opengl_bench

```

This option is used to benchmark all available OpenCL devices and print the results. This option is only available when FFmpeg has been compiled with `--enable-opengl`.

When FFmpeg is configured with `--enable-opengl`, the options for the global OpenCL context are set via `-opengl_options`. See the "OpenCL Options" section in the `ffmpeg-utils` manual for the complete list of supported options. Amongst others, these options include the ability to select a specific platform and device to run the OpenCL code on. By default, FFmpeg will run on the first device of the first platform. While the options for the global OpenCL context provide flexibility to the user in selecting the OpenCL device of their choice, most users would probably want to select the fastest OpenCL device for their system.

This option assists the selection of the most efficient configuration by identifying the appropriate device for the user's system. The built-in benchmark is run on all the OpenCL devices and the performance is measured for each device. The devices in the results list are sorted based on their performance with the fastest device listed first. The user can subsequently invoke `ffmpeg` using the device deemed most appropriate via `-opengl_options` to obtain the best performance for the OpenCL accelerated code.

Typical usage to use the fastest OpenCL device involve the following steps.

Run the command:

```
ffmpeg -opengl_bench
```

Note down the platform ID (*pidx*) and device ID (*didx*) of the first i.e. fastest device in the list. Select the platform and device using the command:

```
ffmpeg -opengl_options platform_idx=pidx:device_idx=didx ...
```

`-opengl_options options (global)`

Set OpenCL environment options. This option is only available when FFmpeg has been compiled with `--enable-opengl`.

*options* must be a list of *key=value* option pairs separated by `;`. See the “OpenCL Options” section in the `ffmpeg-utils` manual for the list of supported options.

## 5.3 AVOptions# TOC

These options are provided directly by the `libavformat`, `libavdevice` and `libavcodec` libraries. To see the list of available AVOptions, use the `-help` option. They are separated into two categories:

`generic`

These options can be set for any container, codec or device. Generic options are listed under `AVFormatContext` options for containers/devices and under `AVCodecContext` options for codecs.

`private`

These options are specific to the given container, device or codec. Private options are listed under their corresponding containers/devices/codecs.

For example to write an ID3v2.3 header instead of a default ID3v2.4 to an MP3 file, use the `id3v2_version` private option of the MP3 muxer:

```
ffmpeg -i input.flac -id3v2_version 3 out.mp3
```

All codec AVOptions are per-stream, and thus a stream specifier should be attached to them.

Note: the `-nooption` syntax cannot be used for boolean AVOptions, use `-option 0/-option 1`.

Note: the old undocumented way of specifying per-stream AVOptions by prepending `v/a/s` to the options name is now obsolete and will be removed soon.

## 5.4 Main options# TOC

`-f fmt (input/output)`

Force input or output file format. The format is normally auto detected for input files and guessed from the file extension for output files, so this option is not needed in most cases.

`-i filename (input)`

input file name

`-y (global)`

Overwrite output files without asking.

`-n (global)`

Do not overwrite output files, and exit immediately if a specified output file already exists.

`-c[:stream_specifier] codec (input/output,per-stream)`

`-codec[:stream_specifier] codec (input/output,per-stream)`

Select an encoder (when used before an output file) or a decoder (when used before an input file) for one or more streams. *codec* is the name of a decoder/encoder or a special value `copy` (output only) to indicate that the stream is not to be re-encoded.

For example

```
ffmpeg -i INPUT -map 0 -c:v libx264 -c:a copy OUTPUT
```

encodes all video streams with libx264 and copies all audio streams.

For each stream, the last matching `c` option is applied, so

```
ffmpeg -i INPUT -map 0 -c copy -c:v:1 libx264 -c:a:137 libvorbis OUTPUT
```

will copy all the streams except the second video, which will be encoded with libx264, and the 138th audio, which will be encoded with libvorbis.

`-t duration (input/output)`

When used as an input option (before `-i`), limit the *duration* of data read from the input file.

When used as an output option (before an output filename), stop writing the output after its duration reaches *duration*.

*duration* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

-to and -t are mutually exclusive and -t has priority.

-to *position* (*output*)

Stop writing the output at *position*. *position* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

-to and -t are mutually exclusive and -t has priority.

-fs *limit\_size* (*output*)

Set the file size limit, expressed in bytes.

-ss *position* (*input/output*)

When used as an input option (before -i), seeks in this input file to *position*. Note that in most formats it is not possible to seek exactly, so ffmpeg will seek to the closest seek point before *position*. When transcoding and -accurate\_seek is enabled (the default), this extra segment between the seek point and *position* will be decoded and discarded. When doing stream copy or when -noaccurate\_seek is used, it will be preserved.

When used as an output option (before an output filename), decodes but discards input until the timestamps reach *position*.

*position* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

-sseof *position* (*input/output*)

Like the -ss option but relative to the "end of file". That is negative values are earlier in the file, 0 is at EOF.

-itsoffset *offset* (*input*)

Set the input time offset.

*offset* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

The offset is added to the timestamps of the input files. Specifying a positive offset means that the corresponding streams are delayed by the time duration specified in *offset*.

`-timestamp date (output)`

Set the recording timestamp in the container.

*date* must be a date specification, see (ffmpeg-utils)the Date section in the ffmpeg-utils(1) manual.

`-metadata[:metadata_specifier] key=value (output,per-metadata)`

Set a metadata key/value pair.

An optional *metadata\_specifier* may be given to set metadata on streams or chapters. See `-map_metadata` documentation for details.

This option overrides metadata set with `-map_metadata`. It is also possible to delete metadata by using an empty value.

For example, for setting the title in the output file:

```
ffmpeg -i in.avi -metadata title="my title" out.flv
```

To set the language of the first audio stream:

```
ffmpeg -i INPUT -metadata:s:a:0 language=eng OUTPUT
```

`-target type (output)`

Specify target file type (vcd, svcd, dvd, dv, dv50). *type* may be prefixed with `pal-`, `ntsc-` or `film-` to use the corresponding standard. All the format options (bitrate, codecs, buffer sizes) are then set automatically. You can just type:

```
ffmpeg -i myfile.avi -target vcd /tmp/vcd.mpg
```

Nevertheless you can specify additional options as long as you know they do not conflict with the standard, as in:

```
ffmpeg -i myfile.avi -target vcd -bf 2 /tmp/vcd.mpg
```

`-dframes number (output)`

Set the number of data frames to output. This is an alias for `-frames:d`.

`-frames[:stream_specifier] framecount (output,per-stream)`

Stop writing to the stream after *framecount* frames.

`-q[:stream_specifier] q (output,per-stream)`

`-qscale[:stream_specifier] q (output,per-stream)`

Use fixed quality scale (VBR). The meaning of *q/qscale* is codec-dependent. If *qscale* is used without a *stream\_specifier* then it applies only to the video stream, this is to maintain compatibility with previous behavior and as specifying the same codec specific value to 2 different codecs that is audio

and video generally is not what is intended when no `stream_specifier` is used.

`-filter[:stream_specifier] filtergraph (output,per-stream)`

Create the filtergraph specified by *filtergraph* and use it to filter the stream.

*filtergraph* is a description of the filtergraph to apply to the stream, and must have a single input and a single output of the same type of the stream. In the filtergraph, the input is associated to the label `in`, and the output to the label `out`. See the `ffmpeg-filters` manual for more information about the filtergraph syntax.

See the `-filter_complex` option if you want to create filtergraphs with multiple inputs and/or outputs.

`-filter_script[:stream_specifier] filename (output,per-stream)`

This option is similar to `-filter`, the only difference is that its argument is the name of the file from which a filtergraph description is to be read.

`-pre[:stream_specifier] preset_name (output,per-stream)`

Specify the preset for matching stream(s).

`-stats (global)`

Print encoding progress/statistics. It is on by default, to explicitly disable it you need to specify `-nostats`.

`-progress url (global)`

Send program-friendly progress information to *url*.

Progress information is written approximately every second and at the end of the encoding process. It is made of "*key=value*" lines. *key* consists of only alphanumeric characters. The last key of a sequence of progress information is always "progress".

`-stdin`

Enable interaction on standard input. On by default unless standard input is used as an input. To explicitly disable interaction you need to specify `-nostdin`.

Disabling interaction on standard input is useful, for example, if `ffmpeg` is in the background process group. Roughly the same result can be achieved with `ffmpeg . . . < /dev/null` but it requires a shell.

`-debug_ts (global)`

Print timestamp information. It is off by default. This option is mostly useful for testing and debugging purposes, and the output format may change from one version to another, so it should not be employed by portable scripts.



See also the option `-fdebug ts`.

`-attach filename (output)`

Add an attachment to the output file. This is supported by a few formats like Matroska for e.g. fonts used in rendering subtitles. Attachments are implemented as a specific type of stream, so this option will add a new stream to the file. It is then possible to use per-stream options on this stream in the usual way. Attachment streams created with this option will be created after all the other streams (i.e. those created with `-map` or automatic mappings).

Note that for Matroska you also have to set the `mimetype` metadata tag:

```
ffmpeg -i INPUT -attach DejaVuSans.ttf -metadata:s:2 mimetype=application/x-truetype-font out.mkv
```

(assuming that the attachment stream will be third in the output file).

`-dump_attachment[:stream_specifier] filename (input,per-stream)`

Extract the matching attachment stream into a file named *filename*. If *filename* is empty, then the value of the `filename` metadata tag will be used.

E.g. to extract the first attachment to a file named 'out.ttf':

```
ffmpeg -dump_attachment:t:0 out.ttf -i INPUT
```

To extract all attachments to files determined by the `filename` tag:

```
ffmpeg -dump_attachment:t "" -i INPUT
```

Technical note – attachments are implemented as codec extradata, so this option can actually be used to extract extradata from any stream, not just attachments.

`-noautorotate`

Disable automatically rotating video based on file metadata.

## 5.5 Video Options# TOC

`-vframes number (output)`

Set the number of video frames to output. This is an alias for `-frames:v`.

`-r[:stream_specifier] fps (input/output,per-stream)`

Set frame rate (Hz value, fraction or abbreviation).

As an input option, ignore any timestamps stored in the file and instead generate timestamps assuming constant frame rate *fps*. This is not the same as the `-framerate` option used for some input formats like `image2` or `v4l2` (it used to be the same in older versions of FFmpeg). If in doubt use `-framerate` instead of the input option `-r`.

As an output option, duplicate or drop input frames to achieve constant output frame rate *fps*.

`-s[:stream_specifier] size (input/output,per-stream)`

Set frame size.

As an input option, this is a shortcut for the `video_size` private option, recognized by some demuxers for which the frame size is either not stored in the file or is configurable – e.g. raw video or video grabbers.

As an output option, this inserts the `scale` video filter to the *end* of the corresponding filtergraph. Please use the `scale` filter directly to insert it at the beginning or some other place.

The format is 'wxh' (default - same as source).

`-aspect[:stream_specifier] aspect (output,per-stream)`

Set the video display aspect ratio specified by *aspect*.

*aspect* can be a floating point number string, or a string of the form *num:den*, where *num* and *den* are the numerator and denominator of the aspect ratio. For example "4:3", "16:9", "1.3333", and "1.7777" are valid argument values.

If used together with `-vcodec copy`, it will affect the aspect ratio stored at container level, but not the aspect ratio stored in encoded frames, if it exists.

`-vn (output)`

Disable video recording.

`-vcodec codec (output)`

Set the video codec. This is an alias for `-codec:v`.

`-pass[:stream_specifier] n (output,per-stream)`

Select the pass number (1 or 2). It is used to do two-pass video encoding. The statistics of the video are recorded in the first pass into a log file (see also the option `-passlogfile`), and in the second pass that log file is used to generate the video at the exact requested bitrate. On pass 1, you may just deactivate audio and set output to null, examples for Windows and Unix:

```
ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y NUL
ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y /dev/null
```

`-passlogfile[:stream_specifier] prefix (output,per-stream)`

Set two-pass log file name prefix to *prefix*, the default file name prefix is "ffmpeg2pass". The complete file name will be `PREFIX-N.log`, where N is a number specific to the output stream

`-vf filtergraph (output)`

Create the filtergraph specified by *filtergraph* and use it to filter the stream.

This is an alias for `-filter:v`, see the `-filter` option.

## 5.6 Advanced Video options# TOC

`-pix_fmt[:stream_specifier] format (input/output,per-stream)`

Set pixel format. Use `-pix_fmts` to show all the supported pixel formats. If the selected pixel format can not be selected, ffmpeg will print a warning and select the best pixel format supported by the encoder. If *pix\_fmt* is prefixed by a +, ffmpeg will exit with an error if the requested pixel format can not be selected, and automatic conversions inside filtergraphs are disabled. If *pix\_fmt* is a single +, ffmpeg selects the same pixel format as the input (or graph output) and automatic conversions are disabled.

`-sws_flags flags (input/output)`

Set SwScaler flags.

`-vdt n`

Discard threshold.

`-rc_override[:stream_specifier] override (output,per-stream)`

Rate control override for specific intervals, formatted as "int,int,int" list separated with slashes. Two first values are the beginning and end frame numbers, last one is quantizer to use if positive, or quality factor if negative.

`-ilme`

Force interlacing support in encoder (MPEG-2 and MPEG-4 only). Use this option if your input file is interlaced and you want to keep the interlaced format for minimum losses. The alternative is to deinterlace the input stream with `-deinterlace`, but deinterlacing introduces losses.

`-psnr`

Calculate PSNR of compressed frames.

`-vstats`

Dump video coding statistics to `vstats_HHMMSS.log`.

`-vstats_file file`

Dump video coding statistics to *file*.

`-top[:stream_specifier] n (output,per-stream)`

top=1/bottom=0/auto=-1 field first

`-dc precision`

Intra\_dc\_precision.

`-vtag fourcc/tag (output)`

Force video tag/fourcc. This is an alias for `-tag:v`.

`-qphist (global)`

Show QP histogram

`-vbsf bitstream_filter`

Deprecated see `-bsf`

`-force_key_frames[:stream_specifier] time[,time...] (output,per-stream)`

`-force_key_frames[:stream_specifier] expr:expr (output,per-stream)`

Force key frames at the specified timestamps, more precisely at the first frames after each specified time.

If the argument is prefixed with `expr:`, the string *expr* is interpreted like an expression and is evaluated for each frame. A key frame is forced in case the evaluation is non-zero.

If one of the times is "`chapters[delta]`", it is expanded into the time of the beginning of all chapters in the file, shifted by *delta*, expressed as a time in seconds. This option can be useful to ensure that a seek point is present at a chapter mark or any other designated place in the output file.

For example, to insert a key frame at 5 minutes, plus key frames 0.1 second before the beginning of every chapter:

`-force_key_frames 0:05:00,chapters-0.1`

The expression in *expr* can contain the following constants:

`n`

the number of current processed frame, starting from 0

`n_forced`

the number of forced frames

`prev_forced_n`

the number of the previous forced frame, it is NAN when no keyframe was forced yet

`prev_forced_t`

the time of the previous forced frame, it is NAN when no keyframe was forced yet

`t`

the time of the current processed frame

For example to force a key frame every 5 seconds, you can specify:

```
-force_key_frames expr:gte(t,n_forced*5)
```

To force a key frame 5 seconds after the time of the last forced one, starting from second 13:

```
-force_key_frames expr:if(isnan(prev_forced_t),gte(t,13),gte(t,prev_forced_t+5))
```

Note that forcing too many keyframes is very harmful for the lookahead algorithms of certain encoders: using fixed-GOP options or similar would be more efficient.

```
-copyinkf[:stream_specifier] (output,per-stream)
```

When doing stream copy, copy also non-key frames found at the beginning.

```
-hwaccel[:stream_specifier] hwaccel (input,per-stream)
```

Use hardware acceleration to decode the matching stream(s). The allowed values of *hwaccel* are:

`none`

Do not use any hardware acceleration (the default).

`auto`

Automatically select the hardware acceleration method.

`vda`

Use Apple VDA hardware acceleration.

`vdpa`

Use VDPAU (Video Decode and Presentation API for Unix) hardware acceleration.

dxva2

Use DXVA2 (DirectX Video Acceleration) hardware acceleration.

This option has no effect if the selected hwaccel is not available or not supported by the chosen decoder.

Note that most acceleration methods are intended for playback and will not be faster than software decoding on modern CPUs. Additionally, `ffmpeg` will usually need to copy the decoded frames from the GPU memory into the system memory, resulting in further performance loss. This option is thus mainly useful for testing.

`-hwaccel_device[:stream_specifier] hwaccel_device (input,per-stream)`

Select a device to use for hardware acceleration.

This option only makes sense when the `-hwaccel` option is also specified. Its exact meaning depends on the specific hardware acceleration method chosen.

vdpau

For VDPAU, this option specifies the X11 display/screen to use. If this option is not specified, the value of the `DISPLAY` environment variable is used

dxva2

For DXVA2, this option should contain the number of the display adapter to use. If this option is not specified, the default adapter is used.

`-hwaccels`

List all hardware acceleration methods supported in this build of `ffmpeg`.

## 5.7 Audio Options# TOC

`-aframes number (output)`

Set the number of audio frames to output. This is an alias for `-frames:a`.

`-ar[:stream_specifier] freq (input/output,per-stream)`

Set the audio sampling frequency. For output streams it is set by default to the frequency of the corresponding input stream. For input streams this option only makes sense for audio grabbing devices and raw demuxers and is mapped to the corresponding demuxer options.

`-aq q (output)`

Set the audio quality (codec-specific, VBR). This is an alias for `-q:a`.

`-ac[:stream_specifier] channels (input/output,per-stream)`

Set the number of audio channels. For output streams it is set by default to the number of input audio channels. For input streams this option only makes sense for audio grabbing devices and raw demuxers and is mapped to the corresponding demuxer options.

`-an (output)`

Disable audio recording.

`-acodec codec (input/output)`

Set the audio codec. This is an alias for `-codec:a`.

`-sample_fmt[:stream_specifier] sample_fmt (output,per-stream)`

Set the audio sample format. Use `-sample_fmts` to get a list of supported sample formats.

`-af filtergraph (output)`

Create the filtergraph specified by *filtergraph* and use it to filter the stream.

This is an alias for `-filter:a`, see the `-filter` option.

## 5.8 Advanced Audio options# TOC

`-atag fourcc/tag (output)`

Force audio tag/fourcc. This is an alias for `-tag:a`.

`-absf bitstream_filter`

Deprecated, see `-bsf`

`-guess_layout_max channels (input,per-stream)`

If some input channel layout is not known, try to guess only if it corresponds to at most the specified number of channels. For example, 2 tells to `ffmpeg` to recognize 1 channel as mono and 2 channels as stereo but not 6 channels as 5.1. The default is to always try to guess. Use 0 to disable all guessing.

## 5.9 Subtitle options# TOC

`-scodec codec (input/output)`

Set the subtitle codec. This is an alias for `-codec:s`.

`-sn (output)`

Disable subtitle recording.

`-sbsf bitstream_filter`

Deprecated, see `-bsf`

## 5.10 Advanced Subtitle options# TOC

`-fix_sub_duration`

Fix subtitles durations. For each subtitle, wait for the next packet in the same stream and adjust the duration of the first to avoid overlap. This is necessary with some subtitles codecs, especially DVB subtitles, because the duration in the original packet is only a rough estimate and the end is actually marked by an empty subtitle frame. Failing to use this option when necessary can result in exaggerated durations or muxing failures due to non-monotonic timestamps.

Note that this option will delay the output of all data until the next subtitle packet is decoded: it may increase memory consumption and latency a lot.

`-canvas_size size`

Set the size of the canvas used to render subtitles.

## 5.11 Advanced options# TOC

`-map`

`[ - ]input_file_id[:stream_specifier][,sync_file_id[:stream_specifier]] | [linklabel] (output)`

Designate one or more input streams as a source for the output file. Each input stream is identified by the input file index *input\_file\_id* and the input stream index *input\_stream\_id* within the input file. Both indices start at 0. If specified, *sync\_file\_id:stream\_specifier* sets which input stream is used as a presentation sync reference.

The first `-map` option on the command line specifies the source for output stream 0, the second `-map` option specifies the source for output stream 1, etc.

A `-` character before the stream identifier creates a "negative" mapping. It disables matching streams from already created mappings.

An alternative *[linklabel]* form will map outputs from complex filter graphs (see the `-filter_complex` option) to the output file. *linklabel* must correspond to a defined output link label in the graph.



For example, to map ALL streams from the first input file to output

```
ffmpeg -i INPUT -map 0 output
```

For example, if you have two audio streams in the first input file, these streams are identified by "0:0" and "0:1". You can use `-map` to select which streams to place in an output file. For example:

```
ffmpeg -i INPUT -map 0:1 out.wav
```

will map the input stream in `INPUT` identified by "0:1" to the (single) output stream in `out.wav`.

For example, to select the stream with index 2 from input file `a.mov` (specified by the identifier "0:2"), and stream with index 6 from input `b.mov` (specified by the identifier "1:6"), and copy them to the output file `out.mov`:

```
ffmpeg -i a.mov -i b.mov -c copy -map 0:2 -map 1:6 out.mov
```

To select all video and the third audio stream from an input file:

```
ffmpeg -i INPUT -map 0:v -map 0:a:2 OUTPUT
```

To map all the streams except the second audio, use negative mappings

```
ffmpeg -i INPUT -map 0 -map -0:a:1 OUTPUT
```

To pick the English audio stream:

```
ffmpeg -i INPUT -map 0:m:language:eng OUTPUT
```

Note that using this option disables the default mappings for this output file.

`-ignore_unknown`

Ignore input streams with unknown type instead of failing if copying such streams is attempted.

`-copy_unknown`

Allow input streams with unknown type to be copied instead of failing if copying such streams is attempted.

`-map_channel`

`[input_file_id.stream_specifier.channel_id|-1][:output_file_id.stream_specifier]`

Map an audio channel from a given input to an output. If `output_file_id.stream_specifier` is not set, the audio channel will be mapped on all the audio streams.

Using "-1" instead of `input_file_id.stream_specifier.channel_id` will map a muted channel.

For example, assuming `INPUT` is a stereo audio file, you can switch the two audio channels with the following command:

```
ffmpeg -i INPUT -map_channel 0.0.1 -map_channel 0.0.0 OUTPUT
```

If you want to mute the first channel and keep the second:

```
ffmpeg -i INPUT -map_channel -1 -map_channel 0.0.1 OUTPUT
```

The order of the "-map\_channel" option specifies the order of the channels in the output stream. The output channel layout is guessed from the number of channels mapped (mono if one "-map\_channel", stereo if two, etc.). Using "-ac" in combination of "-map\_channel" makes the channel gain levels to be updated if input and output channel layouts don't match (for instance two "-map\_channel" options and "-ac 6").

You can also extract each channel of an input to specific outputs; the following command extracts two channels of the *INPUT* audio stream (file 0, stream 0) to the respective *OUTPUT\_CH0* and *OUTPUT\_CH1* outputs:

```
ffmpeg -i INPUT -map_channel 0.0.0 OUTPUT_CH0 -map_channel 0.0.1 OUTPUT_CH1
```

The following example splits the channels of a stereo input into two separate streams, which are put into the same output file:

```
ffmpeg -i stereo.wav -map 0:0 -map 0:0 -map_channel 0.0.0:0.0 -map_channel 0.0.1:0.1 -y out.ogg
```

Note that currently each output stream can only contain channels from a single input stream; you can't for example use "-map\_channel" to pick multiple input audio channels contained in different streams (from the same or different files) and merge them into a single output stream. It is therefore not currently possible, for example, to turn two separate mono streams into a single stereo stream. However splitting a stereo stream into two single channel mono streams is possible.

If you need this feature, a possible workaround is to use the *amerge* filter. For example, if you need to merge a media (here *input.mkv*) with 2 mono audio streams into one single stereo channel audio stream (and keep the video stream), you can use the following command:

```
ffmpeg -i input.mkv -filter_complex "[0:1] [0:2] amerge" -c:a pcm_s16le -c:v copy output.mkv
```

```
-map_metadata[:metadata_spec_out] infile[:metadata_spec_in]  
(output,per-metadata)
```

Set metadata information of the next output file from *infile*. Note that those are file indices (zero-based), not filenames. Optional *metadata\_spec\_in/out* parameters specify, which metadata to copy. A metadata specifier can have the following forms:

*g*

global metadata, i.e. metadata that applies to the whole file

*s[:stream\_spec]*

per-stream metadata. *stream\_spec* is a stream specifier as described in the Stream specifiers chapter. In an input metadata specifier, the first matching stream is copied from. In an output metadata specifier, all matching streams are copied to.

*c:chapter\_index*

per-chapter metadata. *chapter\_index* is the zero-based chapter index.

*p:program\_index*

per-program metadata. *program\_index* is the zero-based program index.

If metadata specifier is omitted, it defaults to global.

By default, global metadata is copied from the first input file, per-stream and per-chapter metadata is copied along with streams/chapters. These default mappings are disabled by creating any mapping of the relevant type. A negative file index can be used to create a dummy mapping that just disables automatic copying.

For example to copy metadata from the first stream of the input file to global metadata of the output file:

```
ffmpeg -i in.ogg -map_metadata 0:s:0 out.mp3
```

To do the reverse, i.e. copy global metadata to all audio streams:

```
ffmpeg -i in.mkv -map_metadata:s:a 0:g out.mkv
```

Note that simple 0 would work as well in this example, since global metadata is assumed by default.

`-map_chapters input_file_index (output)`

Copy chapters from input file with index *input\_file\_index* to the next output file. If no chapter mapping is specified, then chapters are copied from the first input file with at least one chapter. Use a negative file index to disable any chapter copying.

`-benchmark (global)`

Show benchmarking information at the end of an encode. Shows CPU time used and maximum memory consumption. Maximum memory consumption is not supported on all systems, it will usually display as 0 if not supported.

`-benchmark_all (global)`

Show benchmarking information during the encode. Shows CPU time used in various steps (audio/video encode/decode).

`-timelimit duration (global)`

Exit after ffmpeg has been running for *duration* seconds.

`-dump (global)`

Dump each input packet to stderr.

`-hex (global)`

When dumping packets, also dump the payload.

`-re (input)`

Read input at native frame rate. Mainly used to simulate a grab device. or live input stream (e.g. when reading from a file). Should not be used with actual grab devices or live input streams (where it can cause packet loss). By default `ffmpeg` attempts to read the input(s) as fast as possible. This option will slow down the reading of the input(s) to the native frame rate of the input(s). It is useful for real-time output (e.g. live streaming).

`-loop_input`

Loop over the input stream. Currently it works only for image streams. This option is used for automatic FFserver testing. This option is deprecated, use `-loop 1`.

`-loop_output number_of_times`

Repeatedly loop output for formats that support looping such as animated GIF (0 will loop the output infinitely). This option is deprecated, use `-loop`.

`-vsync parameter`

Video sync method. For compatibility reasons old values can be specified as numbers. Newly added values will have to be specified as strings always.

0, `passthrough`

Each frame is passed with its timestamp from the demuxer to the muxer.

1, `cfr`

Frames will be duplicated and dropped to achieve exactly the requested constant frame rate.

2, `vfr`

Frames are passed through with their timestamp or dropped so as to prevent 2 frames from having the same timestamp.

`drop`

As passthrough but destroys all timestamps, making the muxer generate fresh timestamps based on frame-rate.

`-1, auto`

Chooses between 1 and 2 depending on muxer capabilities. This is the default method.

Note that the timestamps may be further modified by the muxer, after this. For example, in the case that the format option `avoid_negative_ts` is enabled.

With `-map` you can select from which stream the timestamps should be taken. You can leave either video or audio unchanged and sync the remaining stream(s) to the unchanged one.

`-frame_drop_threshold parameter`

Frame drop threshold, which specifies how much behind video frames can be before they are dropped. In frame rate units, so 1.0 is one frame. The default is -1.1. One possible usecase is to avoid framedrops in case of noisy timestamps or to increase frame drop precision in case of exact timestamps.

`-async samples_per_second`

Audio sync method. "Stretches/squeezes" the audio stream to match the timestamps, the parameter is the maximum samples per second by which the audio is changed. `-async 1` is a special case where only the start of the audio stream is corrected without any later correction.

Note that the timestamps may be further modified by the muxer, after this. For example, in the case that the format option `avoid_negative_ts` is enabled.

This option has been deprecated. Use the `aresample` audio filter instead.

`-copyts`

Do not process input timestamps, but keep their values without trying to sanitize them. In particular, do not remove the initial start time offset value.

Note that, depending on the `vsync` option or on specific muxer processing (e.g. in case the format option `avoid_negative_ts` is enabled) the output timestamps may mismatch with the input timestamps even when this option is selected.

`-start_at_zero`

When used with `copyts`, shift input timestamps so they start at zero.

This means that using e.g. `-ss 50` will make output timestamps start at 50 seconds, regardless of what timestamp the input file started at.

`-copytb mode`

Specify how to set the encoder timebase when stream copying. *mode* is an integer numeric value, and can assume one of the following values:

1

Use the demuxer timebase.

The time base is copied to the output encoder from the corresponding input demuxer. This is sometimes required to avoid non monotonically increasing timestamps when copying video streams with variable frame rate.

0

Use the decoder timebase.

The time base is copied to the output encoder from the corresponding input decoder.

-1

Try to make the choice automatically, in order to generate a sane output.

Default value is -1.

`-shortest (output)`

Finish encoding when the shortest input stream ends.

`-dts_delta_threshold`

Timestamp discontinuity delta threshold.

`-muxdelay seconds (input)`

Set the maximum demux-decode delay.

`-muxpreload seconds (input)`

Set the initial demux-decode delay.

`-streamid output-stream-index:new-value (output)`

Assign a new stream-id value to an output stream. This option should be specified prior to the output filename to which it applies. For the situation where multiple output files exist, a streamid may be reassigned to a different value.

For example, to set the stream 0 PID to 33 and the stream 1 PID to 36 for an output mpegts file:

```
ffmpeg -i infile -streamid 0:33 -streamid 1:36 out.ts
```

`-bsf[:stream_specifier] bitstream_filters (output,per-stream)`

Set bitstream filters for matching streams. *bitstream\_filters* is a comma-separated list of bitstream filters. Use the `-bsfs` option to get the list of bitstream filters.

```
ffmpeg -i h264.mp4 -c:v copy -bsf:v h264_mp4toannexb -an out.h264
```

```
ffmpeg -i file.mov -an -vn -bsf:s mov2textsub -c:s copy -f rawvideo sub.txt
```

`-tag[:stream_specifier] codec_tag (input/output,per-stream)`

Force a tag/fourcc for matching streams.

`-timecode hh:mm:ssSEPff`

Specify Timecode for writing. *SEP* is ':' for non drop timecode and ';' (or '.') for drop.

```
ffmpeg -i input.mpg -timecode 01:02:03.04 -r 30000/1001 -s ntsc output.mpg
```

`-filter_complex filtergraph (global)`

Define a complex filtergraph, i.e. one with arbitrary number of inputs and/or outputs. For simple graphs – those with one input and one output of the same type – see the `-filter` options. *filtergraph* is a description of the filtergraph, as described in the “Filtergraph syntax” section of the `ffmpeg-filters` manual.

Input link labels must refer to input streams using the `[file_index:stream_specifier]` syntax (i.e. the same as `-map` uses). If *stream\_specifier* matches multiple streams, the first one will be used. An unlabeled input will be connected to the first unused input stream of the matching type.

Output link labels are referred to with `-map`. Unlabeled outputs are added to the first output file.

Note that with this option it is possible to use only `lavfi` sources without normal input files.

For example, to overlay an image over video

```
ffmpeg -i video.mkv -i image.png -filter_complex '[0:v][1:v]overlay[out]' -map '[out]' out.mkv
```

Here `[0:v]` refers to the first video stream in the first input file, which is linked to the first (main) input of the overlay filter. Similarly the first video stream in the second input is linked to the second (overlay) input of overlay.

Assuming there is only one video stream in each input file, we can omit input labels, so the above is equivalent to

```
ffmpeg -i video.mkv -i image.png -filter_complex 'overlay[out]' -map '[out]' out.mkv
```

Furthermore we can omit the output label and the single output from the filter graph will be added to the output file automatically, so we can simply write

```
ffmpeg -i video.mkv -i image.png -filter_complex 'overlay' out.mkv
```

To generate 5 seconds of pure red video using lavfi color source:

```
ffmpeg -filter_complex 'color=c=red' -t 5 out.mkv
```

`-lavfi filtergraph (global)`

Define a complex filtergraph, i.e. one with arbitrary number of inputs and/or outputs. Equivalent to `-filter_complex`.

`-filter_complex_script filename (global)`

This option is similar to `-filter_complex`, the only difference is that its argument is the name of the file from which a complex filtergraph description is to be read.

`-accurate_seek (input)`

This option enables or disables accurate seeking in input files with the `-ss` option. It is enabled by default, so seeking is accurate when transcoding. Use `-noaccurate_seek` to disable it, which may be useful e.g. when copying some streams and transcoding the others.

`-seek_timestamp (input)`

This option enables or disables seeking by timestamp in input files with the `-ss` option. It is disabled by default. If enabled, the argument to the `-ss` option is considered an actual timestamp, and is not offset by the start time of the file. This matters only for files which do not start from timestamp 0, such as transport streams.

`-thread_queue_size size (input)`

This option sets the maximum number of queued packets when reading from the file or device. With low latency / high rate live streams, packets may be discarded if they are not read in a timely manner; raising this value can avoid it.

`-override_ffserver (global)`

Overrides the input specifications from `ffserver`. Using this option you can map any input stream to `ffserver` and control many aspects of the encoding from `ffmpeg`. Without this option `ffmpeg` will transmit to `ffserver` what is requested by `ffserver`.

The option is intended for cases where features are needed that cannot be specified to `ffserver` but can be to `ffmpeg`.

`-sdp_file file (global)`



Print sdp information to *file*. This allows dumping sdp information when at least one output isn't an rtp stream.

`-discard (input)`

Allows discarding specific streams or frames of streams at the demuxer. Not all demuxers support this.

`none`

Discard no frame.

`default`

Default, which discards no frames.

`noref`

Discard all non-reference frames.

`bidir`

Discard all bidirectional frames.

`nokey`

Discard all frames excepts keyframes.

`all`

Discard all frames.

`-xerror (global)`

Stop and exit on error

As a special exception, you can use a bitmap subtitle stream as input: it will be converted into a video with the same size as the largest video in the file, or 720x576 if no video is present. Note that this is an experimental and temporary solution. It will be removed once libavfilter has proper support for subtitles.

For example, to hardcode subtitles on top of a DVB-T recording stored in MPEG-TS format, delaying the subtitles by 1 second:

```
ffmpeg -i input.ts -filter_complex \
'[#0x2ef] setpts=PTS+1/TB [sub] ; [#0x2d0] [sub] overlay' \
-sn -map '#0x2dc' output.mkv
```

(0x2d0, 0x2dc and 0x2ef are the MPEG-TS PIDs of respectively the video, audio and subtitles streams; 0:0, 0:3 and 0:7 would have worked too)

## 5.12 Preset files# TOC

A preset file contains a sequence of *option=value* pairs, one for each line, specifying a sequence of options which would be awkward to specify on the command line. Lines starting with the hash ('#') character are ignored and are used to provide comments. Check the `presets` directory in the FFmpeg source tree for examples.

There are two types of preset files: `ffpreset` and `avpreset` files.

### 5.12.1 ffpreset files# TOC

`ffpreset` files are specified with the `vpre`, `apre`, `spre`, and `fpre` options. The `fpre` option takes the filename of the preset instead of a preset name as input and can be used for any kind of codec. For the `vpre`, `apre`, and `spre` options, the options specified in a preset file are applied to the currently selected codec of the same type as the preset option.

The argument passed to the `vpre`, `apre`, and `spre` preset options identifies the preset file to use according to the following rules:

First `ffmpeg` searches for a file named *arg*.`ffpreset` in the directories `$FFMPEG_DATADIR` (if set), and `$HOME/.ffmpeg`, and in the `datadir` defined at configuration time (usually `PREFIX/share/ffmpeg`) or in a `ffpresets` folder along the executable on win32, in that order. For example, if the argument is `libvpx-1080p`, it will search for the file `libvpx-1080p.ffpreset`.

If no such file is found, then `ffmpeg` will search for a file named *codec\_name-arg*.`ffpreset` in the above-mentioned directories, where *codec\_name* is the name of the codec to which the preset file options will be applied. For example, if you select the video codec with `-vcodec libvpx` and use `-vpre 1080p`, then it will search for the file `libvpx-1080p.ffpreset`.

### 5.12.2 avpreset files# TOC

`avpreset` files are specified with the `pre` option. They work similar to `ffpreset` files, but they only allow encoder-specific options. Therefore, an *option=value* pair specifying an encoder cannot be used.

When the `pre` option is specified, `ffmpeg` will look for files with the suffix `.avpreset` in the directories `$AVCONV_DATADIR` (if set), and `$HOME/.avconv`, and in the `datadir` defined at configuration time (usually `PREFIX/share/ffmpeg`), in that order.

First `ffmpeg` searches for a file named *codec\_name-arg*.`avpreset` in the above-mentioned directories, where *codec\_name* is the name of the codec to which the preset file options will be applied. For example, if you select the video codec with `-vcodec libvpx` and use `-pre 1080p`, then it will search for the file `libvpx-1080p.avpreset`.

If no such file is found, then `ffmpeg` will search for a file named *arg*.`avpreset` in the same directories.

## 6 Examples# TOC

### 6.1 Video and Audio grabbing# TOC

If you specify the input format and device then ffmpeg can grab video and audio directly.

```
ffmpeg -f oss -i /dev/dsp -f video4linux2 -i /dev/video0 /tmp/out.mpg
```

Or with an ALSA audio source (mono input, card id 1) instead of OSS:

```
ffmpeg -f alsa -ac 1 -i hw:1 -f video4linux2 -i /dev/video0 /tmp/out.mpg
```

Note that you must activate the right video source and channel before launching ffmpeg with any TV viewer such as xawtv by Gerd Knorr. You also have to set the audio recording levels correctly with a standard mixer.

### 6.2 X11 grabbing# TOC

Grab the X11 display with ffmpeg via

```
ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0 /tmp/out.mpg
```

0.0 is display.screen number of your X11 server, same as the DISPLAY environment variable.

```
ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0+10,20 /tmp/out.mpg
```

0.0 is display.screen number of your X11 server, same as the DISPLAY environment variable. 10 is the x-offset and 20 the y-offset for the grabbing.

### 6.3 Video and Audio file format conversion# TOC

Any supported file format and protocol can serve as input to ffmpeg:

Examples:

- You can use YUV files as input:

```
ffmpeg -i /tmp/test%d.Y /tmp/out.mpg
```

It will use the files:

```
/tmp/test0.Y, /tmp/test0.U, /tmp/test0.V,  
/tmp/test1.Y, /tmp/test1.U, /tmp/test1.V, etc...
```

The Y files use twice the resolution of the U and V files. They are raw files, without header. They can be generated by all decent video decoders. You must specify the size of the image with the `-s` option if ffmpeg cannot guess it.

- You can input from a raw YUV420P file:

```
ffmpeg -i /tmp/test.yuv /tmp/out.avi
```

test.yuv is a file containing raw YUV planar data. Each frame is composed of the Y plane followed by the U and V planes at half vertical and horizontal resolution.

- You can output to a raw YUV420P file:

```
ffmpeg -i mydivx.avi hugefile.yuv
```

- You can set several input files and output files:

```
ffmpeg -i /tmp/a.wav -s 640x480 -i /tmp/a.yuv /tmp/a.mpg
```

Converts the audio file a.wav and the raw YUV video file a.yuv to MPEG file a.mpg.

- You can also do audio and video conversions at the same time:

```
ffmpeg -i /tmp/a.wav -ar 22050 /tmp/a.mp2
```

Converts a.wav to MPEG audio at 22050 Hz sample rate.

- You can encode to several formats at the same time and define a mapping from input stream to output streams:

```
ffmpeg -i /tmp/a.wav -map 0:a -b:a 64k /tmp/a.mp2 -map 0:a -b:a 128k /tmp/b.mp2
```

Converts a.wav to a.mp2 at 64 kbits and to b.mp2 at 128 kbits. '-map file:index' specifies which input stream is used for each output stream, in the order of the definition of output streams.

- You can transcode decrypted VOBs:

```
ffmpeg -i snatch_1.vob -f avi -c:v mpeg4 -b:v 800k -g 300 -bf 2 -c:a libmp3lame -b:a 128k snatch.avi
```

This is a typical DVD ripping example; the input is a VOB file, the output an AVI file with MPEG-4 video and MP3 audio. Note that in this command we use B-frames so the MPEG-4 stream is DivX5 compatible, and GOP size is 300 which means one intra frame every 10 seconds for 29.97fps input video. Furthermore, the audio stream is MP3-encoded so you need to enable LAME support by passing --enable-libmp3lame to configure. The mapping is particularly useful for DVD transcoding to get the desired audio language.

NOTE: To see the supported input formats, use `ffmpeg -formats`.

- You can extract images from a video, or create a video from many images:

For extracting images from a video:

```
ffmpeg -i foo.avi -r 1 -s WxH -f image2 foo-%03d.jpeg
```

This will extract one video frame per second from the video and will output them in files named `foo-001.jpeg`, `foo-002.jpeg`, etc. Images will be rescaled to fit the new WxH values.

If you want to extract just a limited number of frames, you can use the above command in combination with the `-vframes` or `-t` option, or in combination with `-ss` to start extracting from a certain point in time.

For creating a video from many images:

```
ffmpeg -f image2 -framerate 12 -i foo-%03d.jpeg -s WxH foo.avi
```

The syntax `foo-%03d.jpeg` specifies to use a decimal number composed of three digits padded with zeroes to express the sequence number. It is the same syntax supported by the C `printf` function, but only formats accepting a normal integer are suitable.

When importing an image sequence, `-i` also supports expanding shell-like wildcard patterns (globbing) internally, by selecting the image2-specific `-pattern_type glob` option.

For example, for creating a video from filenames matching the glob pattern `foo-*.jpeg`:

```
ffmpeg -f image2 -pattern_type glob -framerate 12 -i 'foo-*.jpeg' -s WxH foo.avi
```

- You can put many streams of the same type in the output:

```
ffmpeg -i test1.avi -i test2.avi -map 1:1 -map 1:0 -map 0:1 -map 0:0 -c copy -y test12.nut
```

The resulting output file `test12.nut` will contain the first four streams from the input files in reverse order.

- To force CBR video output:

```
ffmpeg -i myfile.avi -b 4000k -minrate 4000k -maxrate 4000k -bufsize 1835k out.m2v
```

- The four options `lmin`, `lmax`, `mblmin` and `mblmax` use 'lambda' units, but you may use the `QP2LAMBDA` constant to easily convert from 'q' units:

```
ffmpeg -i src.ext -lmax 21*QP2LAMBDA dst.ext
```

## 7 Syntax# TOC

This section documents the syntax and formats employed by the FFmpeg libraries and tools.

### 7.1 Quoting and escaping# TOC

FFmpeg adopts the following quoting and escaping mechanism, unless explicitly specified. The following rules are applied:

- ‘’ and ‘\’ are special characters (respectively used for quoting and escaping). In addition to them, there might be other special characters depending on the specific syntax where the escaping and quoting are employed.
- A special character is escaped by prefixing it with a ‘\’.
- All characters enclosed between ‘ ’’ are included literally in the parsed string. The quote character ‘’ itself cannot be quoted, so you may need to close the quote and escape it.
- Leading and trailing whitespaces, unless escaped or quoted, are removed from the parsed string.

Note that you may need to add a second level of escaping when using the command line or a script, which depends on the syntax of the adopted shell language.

The function `av_get_token` defined in `libavutil/avstring.h` can be used to parse a token quoted or escaped according to the rules defined above.

The tool `tools/ffescape` in the FFmpeg source tree can be used to automatically quote or escape a string in a script.

### 7.1.1 Examples# TOC

- Escape the string `Crime d'Amour` containing the ‘ ’ special character:

```
Crime d\'Amour
```

- The string above contains a quote, so the ‘ ’ needs to be escaped when quoting it:

```
'Crime d\'\'Amour'
```

- Include leading or trailing whitespaces using quoting:

```
' this string starts and ends with whitespaces '
```

- Escaping and quoting can be mixed together:

```
' The string \'string\' is a string '
```

- To include a literal ‘\’ you can use either escaping or quoting:

```
'c:\foo' can be written as c:\\foo
```

## 7.2 Date# TOC

The accepted syntax is:

```
[ (YYYY-MM-DD|YYYYMMDD) [T|t| ] ( (HH:MM:SS[.m...]) | (HHMMSS[.m...]) ) [Z]
now
```

If the value is "now" it takes the current time.

Time is local time unless Z is appended, in which case it is interpreted as UTC. If the year-month-day part is not specified it takes the current year-month-day.

## 7.3 Time duration# TOC

There are two accepted syntaxes for expressing time duration.

`[ - ][HH:]MM:SS[.m...]`

*HH* expresses the number of hours, *MM* the number of minutes for a maximum of 2 digits, and *SS* the number of seconds for a maximum of 2 digits. The *m* at the end expresses decimal value for *SS*.

*or*

`[ - ]S+[.m...]`

*S* expresses the number of seconds, with the optional decimal part *m*.

In both expressions, the optional ‘-’ indicates negative duration.

### 7.3.1 Examples# TOC

The following examples are all valid time duration:

‘55’

55 seconds

‘12:03:45’

12 hours, 03 minutes and 45 seconds

‘23.189’

23.189 seconds

## 7.4 Video size# TOC

Specify the size of the sourced video, it may be a string of the form *widthxheight*, or the name of a size abbreviation.

The following abbreviations are recognized:

‘ntsc’

720x480

‘pal’

720x576

‘qntsc’

352x240

‘qpai’

352x288

‘sntsc’

640x480

‘spai’

768x576

‘film’

352x240

‘ntsc-film’

352x240

‘sqcif’

128x96

‘qcif’

176x144

‘cif’

352x288

‘4cif’

704x576

‘16cif’

1408x1152

‘qqvga’



160x120

‘qvga’

320x240

‘vga’

640x480

‘svga’

800x600

‘xga’

1024x768

‘uxga’

1600x1200

‘qxga’

2048x1536

‘sxga’

1280x1024

‘qsxga’

2560x2048

‘hsxga’

5120x4096

‘wvga’

852x480

‘wxga’

1366x768

‘wsxga’

1600x1024

‘wuxga’

1920x1200

‘woxga’

2560x1600

‘wqsxga’

3200x2048

‘wquxga’

3840x2400

‘whsxga’

6400x4096

‘whuxga’

7680x4800

‘cga’

320x200

‘ega’

640x350

‘hd480’

852x480

‘hd720’

1280x720

‘hd1080’

1920x1080

‘2k’

2048x1080

'2kflat'

1998x1080

'2kscope'

2048x858

'4k'

4096x2160

'4kflat'

3996x2160

'4kscope'

4096x1716

'nhd'

640x360

'hqvga'

240x160

'wqvga'

400x240

'fwqvga'

432x240

'hvga'

480x320

'qhd'

960x540

'2kdc1'

2048x1080

‘4kdc1’

4096x2160

‘uhd2160’

3840x2160

‘uhd4320’

7680x4320

## 7.5 Video rate# TOC

Specify the frame rate of a video, expressed as the number of frames generated per second. It has to be a string in the format *frame\_rate\_num/frame\_rate\_den*, an integer number, a float number or a valid video frame rate abbreviation.

The following abbreviations are recognized:

‘ntsc’

30000/1001

‘pal’

25/1

‘qntsc’

30000/1001

‘qp1’

25/1

‘sntsc’

30000/1001

‘sp1’

25/1

‘film’

24/1

`'ntsc-film'`

24000/1001

## 7.6 Ratio# TOC

A ratio can be expressed as an expression, or in the form *numerator:denominator*.

Note that a ratio with infinite (1/0) or negative value is considered valid, so you should check on the returned value if you want to exclude those values.

The undefined value can be expressed using the "0:0" string.

## 7.7 Color# TOC

It can be the name of a color as defined below (case insensitive match) or a `[ 0x | # ]RRGGBB[ AA ]` sequence, possibly followed by `@` and a string representing the alpha component.

The alpha component may be a string composed by "0x" followed by an hexadecimal number or a decimal number between 0.0 and 1.0, which represents the opacity value (`'0x00'` or `'0.0'` means completely transparent, `'0xff'` or `'1.0'` completely opaque). If the alpha component is not specified then `'0xff'` is assumed.

The string `'random'` will result in a random color.

The following names of colors are recognized:

`'AliceBlue'`

`0xF0F8FF`

`'AntiqueWhite'`

`0xFAEBD7`

`'Aqua'`

`0x00FFFF`

`'Aquamarine'`

`0x7FFFD4`

`'Azure'`

0xF0FFFF

‘Beige’

0xF5F5DC

‘Bisque’

0xFFE4C4

‘Black’

0x000000

‘BlanchedAlmond’

0xFFEBCD

‘Blue’

0x0000FF

‘BlueViolet’

0x8A2BE2

‘Brown’

0xA52A2A

‘BurlyWood’

0xDEB887

‘CadetBlue’

0x5F9EA0

‘Chartreuse’

0x7FFF00

‘Chocolate’

0xD2691E

‘Coral’

0xFF7F50

‘CornflowerBlue’

0x6495ED

‘Cornsilk’

0xFFFF8DC

‘Crimson’

0xDC143C

‘Cyan’

0x00FFFF

‘DarkBlue’

0x00008B

‘DarkCyan’

0x008B8B

‘DarkGoldenRod’

0xB8860B

‘DarkGray’

0xA9A9A9

‘DarkGreen’

0x006400

‘DarkKhaki’

0xBDB76B

‘DarkMagenta’

0x8B008B

‘DarkOliveGreen’

0x556B2F

‘Darkorange’

0xFF8C00

‘DarkOrchid’

0x9932CC

‘DarkRed’

0x8B0000

‘DarkSalmon’

0xE9967A

‘DarkSeaGreen’

0x8FBC8F

‘DarkSlateBlue’

0x483D8B

‘DarkSlateGray’

0x2F4F4F

‘DarkTurquoise’

0x00CED1

‘DarkViolet’

0x9400D3

‘DeepPink’

0xFF1493

‘DeepSkyBlue’

0x00BFFF

‘DimGray’



0x696969

'DodgerBlue'

0x1E90FF

'FireBrick'

0xB22222

'FloralWhite'

0xFFFAF0

'ForestGreen'

0x228B22

'Fuchsia'

0xFF00FF

'Gainsboro'

0xDCDCDC

'GhostWhite'

0xF8F8FF

'Gold'

0xFFD700

'GoldenRod'

0xDAA520

'Gray'

0x808080

'Green'

0x008000

'GreenYellow'

0xADFF2F

‘HoneyDew’

0xF0FFF0

‘HotPink’

0xFF69B4

‘IndianRed’

0xCD5C5C

‘Indigo’

0x4B0082

‘Ivory’

0xFFFFF0

‘Khaki’

0xF0E68C

‘Lavender’

0xE6E6FA

‘LavenderBlush’

0xFFF0F5

‘LawnGreen’

0x7CFC00

‘LemonChiffon’

0xFFFACD

‘LightBlue’

0xADD8E6

‘LightCoral’

0xF08080

'LightCyan'

0xE0FFFF

'LightGoldenRodYellow'

0xFAFAD2

'LightGreen'

0x90EE90

'LightGrey'

0xD3D3D3

'LightPink'

0xFFB6C1

'LightSalmon'

0xFFA07A

'LightSeaGreen'

0x20B2AA

'LightSkyBlue'

0x87CEFA

'LightSlateGray'

0x778899

'LightSteelBlue'

0xB0C4DE

'LightYellow'

0xFFFFE0

'Lime'

0x00FF00

'LimeGreen'

0x32CD32

'Linen'

0xFAF0E6

'Magenta'

0xFF00FF

'Maroon'

0x800000

'MediumAquaMarine'

0x66CDAA

'MediumBlue'

0x0000CD

'MediumOrchid'

0xBA55D3

'MediumPurple'

0x9370D8

'MediumSeaGreen'

0x3CB371

'MediumSlateBlue'

0x7B68EE

'MediumSpringGreen'

0x00FA9A

'MediumTurquoise'

0x48D1CC

'MediumVioletRed'

0xC71585

'MidnightBlue'

0x191970

'MintCream'

0xF5FFFA

'MistyRose'

0xFFE4E1

'Moccasin'

0xFFE4B5

'NavajoWhite'

0xFFDEAD

'Navy'

0x000080

'OldLace'

0xFDF5E6

'Olive'

0x808000

'OliveDrab'

0x6B8E23

'Orange'

0xFFA500

'OrangeRed'

0xFF4500

‘Orchid’

0xDA70D6

‘PaleGoldenRod’

0xEEE8AA

‘PaleGreen’

0x98FB98

‘PaleTurquoise’

0xAFEEEE

‘PaleVioletRed’

0xD87093

‘PapayaWhip’

0xFFEFD5

‘PeachPuff’

0xFFDAB9

‘Peru’

0xCD853F

‘Pink’

0xFFC0CB

‘Plum’

0xDDA0DD

‘PowderBlue’

0xB0E0E6

‘Purple’

0x800080

‘Red’

0xFF0000

‘RosyBrown’

0xBC8F8F

‘RoyalBlue’

0x4169E1

‘SaddleBrown’

0x8B4513

‘Salmon’

0xFA8072

‘SandyBrown’

0xF4A460

‘SeaGreen’

0x2E8B57

‘SeaShell’

0xFFF5EE

‘Sienna’

0xA0522D

‘Silver’

0xC0C0C0

‘SkyBlue’

0x87CEEB

‘SlateBlue’

0x6A5ACD

'SlateGray'

0x708090

'Snow'

0xFFFAFA

'SpringGreen'

0x00FF7F

'SteelBlue'

0x4682B4

'Tan'

0xD2B48C

'Teal'

0x008080

'Thistle'

0xD8BFD8

'Tomato'

0xFF6347

'Turquoise'

0x40E0D0

'Violet'

0xEE82EE

'Wheat'

0xF5DEB3

'White'



0xFFFFFFFF

‘WhiteSmoke’

0xF5F5F5

‘Yellow’

0xFFFF00

‘YellowGreen’

0x9ACD32

## 7.8 Channel Layout# TOC

A channel layout specifies the spatial disposition of the channels in a multi-channel audio stream. To specify a channel layout, FFmpeg makes use of a special syntax.

Individual channels are identified by an id, as given by the table below:

‘FL’

front left

‘FR’

front right

‘FC’

front center

‘LFE’

low frequency

‘BL’

back left

‘BR’

back right

‘FLC’

front left-of-center

‘FRC’

front right-of-center

‘BC’

back center

‘SL’

side left

‘SR’

side right

‘TC’

top center

‘TFL’

top front left

‘TFC’

top front center

‘TFR’

top front right

‘TBL’

top back left

‘TBC’

top back center

‘TBR’

top back right

‘DL’

downmix left

‘DR’

downmix right

‘WL’

wide left

‘WR’

wide right

‘SDL’

surround direct left

‘SDR’

surround direct right

‘LFE2’

low frequency 2

Standard channel layout compositions can be specified by using the following identifiers:

‘mono’

FC

‘stereo’

FL+FR

‘2.1’

FL+FR+LFE

‘3.0’

FL+FR+FC

‘3.0(back)’

FL+FR+BC

‘4.0’

FL+FR+FC+BC

‘quad’

FL+FR+BL+BR

‘quad(side)’

FL+FR+SL+SR

‘3.1’

FL+FR+FC+LFE

‘5.0’

FL+FR+FC+BL+BR

‘5.0(side)’

FL+FR+FC+SL+SR

‘4.1’

FL+FR+FC+LFE+BC

‘5.1’

FL+FR+FC+LFE+BL+BR

‘5.1(side)’

FL+FR+FC+LFE+SL+SR

‘6.0’

FL+FR+FC+BC+SL+SR

‘6.0(front)’

FL+FR+FLC+FRC+SL+SR

‘hexagonal’

FL+FR+FC+BL+BR+BC

‘6.1’

FL+FR+FC+LFE+BC+SL+SR

‘6.1’

FL+FR+FC+LFE+BL+BR+BC

'6.1(front)'

FL+FR+LFE+FLC+FRC+SL+SR

'7.0'

FL+FR+FC+BL+BR+SL+SR

'7.0(front)'

FL+FR+FC+FLC+FRC+SL+SR

'7.1'

FL+FR+FC+LFE+BL+BR+SL+SR

'7.1(wide)'

FL+FR+FC+LFE+BL+BR+FLC+FRC

'7.1(wide-side)'

FL+FR+FC+LFE+FLC+FRC+SL+SR

'octagonal'

FL+FR+FC+BL+BR+BC+SL+SR

'downmix'

DL+DR

A custom channel layout can be specified as a sequence of terms, separated by '+' or '|'. Each term can be:

- the name of a standard channel layout (e.g. 'mono', 'stereo', '4.0', 'quad', '5.0', etc.)
- the name of a single channel (e.g. 'FL', 'FR', 'FC', 'LFE', etc.)
- a number of channels, in decimal, optionally followed by 'c', yielding the default channel layout for that number of channels (see the function `av_get_default_channel_layout`)
- a channel layout mask, in hexadecimal starting with "0x" (see the `AV_CH_*` macros in `libavutil/channel_layout.h`).

Starting from libavutil version 53 the trailing character "c" to specify a number of channels will be required, while a channel layout mask could also be specified as a decimal number (if and only if not followed by "c").

See also the function `av_get_channel_layout` defined in `libavutil/channel_layout.h`.

## 8 Expression Evaluation# TOC

When evaluating an arithmetic expression, FFmpeg uses an internal formula evaluator, implemented through the `libavutil/eval.h` interface.

An expression may contain unary, binary operators, constants, and functions.

Two expressions *expr1* and *expr2* can be combined to form another expression "*expr1;expr2*". *expr1* and *expr2* are evaluated in turn, and the new expression evaluates to the value of *expr2*.

The following binary operators are available: +, -, \*, /, ^.

The following unary operators are available: +, -.

The following functions are available:

`abs(x)`

Compute absolute value of *x*.

`acos(x)`

Compute arccosine of *x*.

`asin(x)`

Compute arcsine of *x*.

`atan(x)`

Compute arctangent of *x*.

`between(x, min, max)`

Return 1 if *x* is greater than or equal to *min* and lesser than or equal to *max*, 0 otherwise.

`bitand(x, y)`

`bitor(x, y)`

Compute bitwise and/or operation on *x* and *y*.

The results of the evaluation of *x* and *y* are converted to integers before executing the bitwise operation.

Note that both the conversion to integer and the conversion back to floating point can lose precision. Beware of unexpected results for large numbers (usually  $2^{53}$  and larger).

`ceil(expr)`

Round the value of expression *expr* upwards to the nearest integer. For example, "ceil(1.5)" is "2.0".

`clip(x, min, max)`

Return the value of *x* clipped between *min* and *max*.

`cos(x)`

Compute cosine of *x*.

`cosh(x)`

Compute hyperbolic cosine of *x*.

`eq(x, y)`

Return 1 if *x* and *y* are equivalent, 0 otherwise.

`exp(x)`

Compute exponential of *x* (with base *e*, the Euler's number).

`floor(expr)`

Round the value of expression *expr* downwards to the nearest integer. For example, "floor(-1.5)" is "-2.0".

`gauss(x)`

Compute Gauss function of *x*, corresponding to  $\exp(-x^2/2) / \sqrt{2\pi}$ .

`gcd(x, y)`

Return the greatest common divisor of *x* and *y*. If both *x* and *y* are 0 or either or both are less than zero then behavior is undefined.

`gt(x, y)`

Return 1 if *x* is greater than *y*, 0 otherwise.

`gte(x, y)`

Return 1 if *x* is greater than or equal to *y*, 0 otherwise.

`hypot(x, y)`

This function is similar to the C function with the same name; it returns " $\sqrt{x*x + y*y}$ ", the length of the hypotenuse of a right triangle with sides of length  $x$  and  $y$ , or the distance of the point  $(x, y)$  from the origin.

`if(x, y)`

Evaluate  $x$ , and if the result is non-zero return the result of the evaluation of  $y$ , return 0 otherwise.

`if(x, y, z)`

Evaluate  $x$ , and if the result is non-zero return the evaluation result of  $y$ , otherwise the evaluation result of  $z$ .

`ifnot(x, y)`

Evaluate  $x$ , and if the result is zero return the result of the evaluation of  $y$ , return 0 otherwise.

`ifnot(x, y, z)`

Evaluate  $x$ , and if the result is zero return the evaluation result of  $y$ , otherwise the evaluation result of  $z$ .

`isinf(x)`

Return 1.0 if  $x$  is +/-INFINITY, 0.0 otherwise.

`isnan(x)`

Return 1.0 if  $x$  is NAN, 0.0 otherwise.

`ld(var)`

Load the value of the internal variable with number  $var$ , which was previously stored with `st(var, expr)`. The function returns the loaded value.

`log(x)`

Compute natural logarithm of  $x$ .

`lt(x, y)`

Return 1 if  $x$  is lesser than  $y$ , 0 otherwise.

`lte(x, y)`

Return 1 if  $x$  is lesser than or equal to  $y$ , 0 otherwise.

`max(x, y)`



Return the maximum between  $x$  and  $y$ .

`min(x, y)`

Return the maximum between  $x$  and  $y$ .

`mod(x, y)`

Compute the remainder of division of  $x$  by  $y$ .

`not(expr)`

Return 1.0 if  $expr$  is zero, 0.0 otherwise.

`pow(x, y)`

Compute the power of  $x$  elevated  $y$ , it is equivalent to " $(x)^{(y)}$ ".

`print(t)`

`print(t, l)`

Print the value of expression  $t$  with loglevel  $l$ . If  $l$  is not specified then a default log level is used.  
Returns the value of the expression printed.

Prints  $t$  with loglevel  $l$

`random(x)`

Return a pseudo random value between 0.0 and 1.0.  $x$  is the index of the internal variable which will be used to save the seed/state.

`root(expr, max)`

Find an input value for which the function represented by  $expr$  with argument  $ld(0)$  is 0 in the interval  $0..max$ .

The expression in  $expr$  must denote a continuous function or the result is undefined.

$ld(0)$  is used to represent the function input value, which means that the given expression will be evaluated multiple times with various input values that the expression can access through `ld(0)`.  
When the expression evaluates to 0 then the corresponding input value will be returned.

`sin(x)`

Compute sine of  $x$ .

`sinh(x)`

Compute hyperbolic sine of  $x$ .

```
sqrt(expr)
```

Compute the square root of  $expr$ . This is equivalent to " $(expr)^{.5}$ ".

```
squish(x)
```

Compute expression  $1 / (1 + \exp(4 * x))$ .

```
st(var, expr)
```

Store the value of the expression  $expr$  in an internal variable.  $var$  specifies the number of the variable where to store the value, and it is a value ranging from 0 to 9. The function returns the value stored in the internal variable. Note, Variables are currently not shared between expressions.

```
tan(x)
```

Compute tangent of  $x$ .

```
tanh(x)
```

Compute hyperbolic tangent of  $x$ .

```
taylor(expr, x)
```

```
taylor(expr, x, id)
```

Evaluate a Taylor series at  $x$ , given an expression representing the  $ld(id)$ -th derivative of a function at 0.

When the series does not converge the result is undefined.

$ld(id)$  is used to represent the derivative order in  $expr$ , which means that the given expression will be evaluated multiple times with various input values that the expression can access through  $ld(id)$ . If  $id$  is not specified then 0 is assumed.

Note, when you have the derivatives at  $y$  instead of 0, `taylor(expr, x-y)` can be used.

```
time(0)
```

Return the current (wallclock) time in seconds.

```
trunc(expr)
```

Round the value of expression  $expr$  towards zero to the nearest integer. For example, "`trunc(-1.5)`" is "-1.0".

```
while(cond, expr)
```

Evaluate expression *expr* while the expression *cond* is non-zero, and returns the value of the last *expr* evaluation, or NAN if *cond* was always false.

The following constants are available:

PI

area of the unit disc, approximately 3.14

E

exp(1) (Euler's number), approximately 2.718

PHI

golden ratio  $(1+\sqrt{5})/2$ , approximately 1.618

Assuming that an expression is considered "true" if it has a non-zero value, note that:

\* works like AND

+ works like OR

For example the construct:

```
if (A AND B) then C
```

is equivalent to:

```
if (A*B, C)
```

In your C code, you can extend the list of unary and binary functions, and define recognized constants, so that they are available for your expressions.

The evaluator also recognizes the International System unit prefixes. If 'i' is appended after the prefix, binary prefixes are used, which are based on powers of 1024 instead of powers of 1000. The 'B' postfix multiplies the value by 8, and can be appended after a unit prefix or used alone. This allows using for example 'KB', 'MiB', 'G' and 'B' as number postfix.

The list of available International System prefixes follows, with indication of the corresponding powers of 10 and of 2.

Y

$10^{-24}$  /  $2^{-80}$

Z

$$10^{-21} / 2^{-70}$$

a

$$10^{-18} / 2^{-60}$$

f

$$10^{-15} / 2^{-50}$$

p

$$10^{-12} / 2^{-40}$$

n

$$10^{-9} / 2^{-30}$$

u

$$10^{-6} / 2^{-20}$$

m

$$10^{-3} / 2^{-10}$$

c

$$10^{-2}$$

d

$$10^{-1}$$

h

$$10^2$$

k

$$10^3 / 2^{10}$$

K

$$10^3 / 2^{10}$$

M

$10^6 / 2^{20}$

G

$10^9 / 2^{30}$

T

$10^{12} / 2^{40}$

P

$10^{15} / 2^{40}$

E

$10^{18} / 2^{50}$

Z

$10^{21} / 2^{60}$

Y

$10^{24} / 2^{70}$

## 9 OpenCL Options# TOC

When FFmpeg is configured with `--enable-opengl`, it is possible to set the options for the global OpenCL context.

The list of supported options follows:

`build_options`

Set build options used to compile the registered kernels.

See reference "OpenCL Specification Version: 1.2 chapter 5.6.4".

`platform_idx`

Select the index of the platform to run OpenCL code.

The specified index must be one of the indexes in the device list which can be obtained with `ffmpeg -opengl_bench` or `av_opengl_get_device_list()`.

`device_idx`

Select the index of the device used to run OpenCL code.

The specified index must be one of the indexes in the device list which can be obtained with `ffmpeg -opengl_bench` or `av_opengl_get_device_list()`.

## 10 Codec Options# TOC

libavcodec provides some generic global options, which can be set on all the encoders and decoders. In addition each codec may support so-called private options, which are specific for a given codec.

Sometimes, a global option may only affect a specific kind of codec, and may be nonsensical or ignored by another, so you need to be aware of the meaning of the specified options. Also some options are meant only for decoding or encoding.

Options may be set by specifying *-option value* in the FFmpeg tools, or by setting the value explicitly in the `AVCodecContext` options or using the `libavutil/opt.h` API for programmatic use.

The list of supported options follow:

*b integer (encoding, audio, video)*

Set bitrate in bits/s. Default value is 200K.

*ab integer (encoding, audio)*

Set audio bitrate (in bits/s). Default value is 128K.

*bt integer (encoding, video)*

Set video bitrate tolerance (in bits/s). In 1-pass mode, bitrate tolerance specifies how far ratecontrol is willing to deviate from the target average bitrate value. This is not related to min/max bitrate. Lowering tolerance too much has an adverse effect on quality.

*flags flags (decoding/encoding, audio, video, subtitles)*

Set generic flags.

Possible values:

`'mv4'`

Use four motion vector by macroblock (mpeg4).

`'qpel'`

Use 1/4 pel motion compensation.

`'loop'`

Use loop filter.

`'qscale'`

Use fixed qscale.

`'gmc'`

Use gmc.

`'mv0'`

Always try a mb with mv=<0,0>.

`'input_preserved'`

`'pass1'`

Use internal 2pass ratecontrol in first pass mode.

`'pass2'`

Use internal 2pass ratecontrol in second pass mode.

`'gray'`

Only decode/encode grayscale.

`'emu_edge'`

Do not draw edges.

`'psnr'`

Set error[?] variables during encoding.

`'truncated'`

`'naq'`

Normalize adaptive quantization.

`'ildct'`

Use interlaced DCT.

`'low_delay'`

Force low delay.

`'global_header'`

Place global headers in extradata instead of every keyframe.

`'bitexact'`

Only write platform-, build- and time-independent data. (except (I)DCT). This ensures that file and data checksums are reproducible and match between platforms. Its primary use is for regression testing.

`'aic'`

Apply H263 advanced intra coding / mpeg4 ac prediction.

`'cbp'`

Deprecated, use mpegvideo private options instead.

`'qprd'`

Deprecated, use mpegvideo private options instead.

`'ilme'`

Apply interlaced motion estimation.

`'cgop'`

Use closed gop.

`me_method integer (encoding, video)`

Set motion estimation method.

Possible values:

`'zero'`

zero motion estimation (fastest)

`'full'`

full motion estimation (slowest)

`'epzs'`



EPZS motion estimation (default)

‘esa’

esa motion estimation (alias for full)

‘tesa’

tesa motion estimation

‘dia’

dia motion estimation (alias for epzs)

‘log’

log motion estimation

‘phods’

phods motion estimation

‘x1’

X1 motion estimation

‘hex’

hex motion estimation

‘umh’

umh motion estimation

‘iter’

iter motion estimation

`extradata_size` *integer*

Set extradata size.

`time_base` *rational number*

Set codec time base.

It is the fundamental unit of time (in seconds) in terms of which frame timestamps are represented. For fixed-fps content, timebase should be  $1 / \text{frame\_rate}$  and timestamp increments should be identically 1.

*g integer (encoding,video)*

Set the group of picture size. Default value is 12.

*ar integer (decoding/encoding,audio)*

Set audio sampling rate (in Hz).

*ac integer (decoding/encoding,audio)*

Set number of audio channels.

*cutoff integer (encoding,audio)*

Set cutoff bandwidth.

*frame\_size integer (encoding,audio)*

Set audio frame size.

Each submitted frame except the last must contain exactly *frame\_size* samples per channel. May be 0 when the codec has CODEC\_CAP\_VARIABLE\_FRAME\_SIZE set, in that case the frame size is not restricted. It is set by some decoders to indicate constant frame size.

*frame\_number integer*

Set the frame number.

*delay integer*

*qcomp float (encoding,video)*

Set video quantizer scale compression (VBR). It is used as a constant in the ratecontrol equation. Recommended range for default rc\_eq: 0.0-1.0.

*qblur float (encoding,video)*

Set video quantizer scale blur (VBR).

*qmin integer (encoding,video)*

Set min video quantizer scale (VBR). Must be included between -1 and 69, default value is 2.

*qmax integer (encoding,video)*

Set max video quantizer scale (VBR). Must be included between -1 and 1024, default value is 31.

*qdiff integer (encoding,video)*

Set max difference between the quantizer scale (VBR).

`bf integer (encoding,video)`

Set max number of B frames between non-B-frames.

Must be an integer between -1 and 16. 0 means that B-frames are disabled. If a value of -1 is used, it will choose an automatic value depending on the encoder.

Default value is 0.

`b_qfactor float (encoding,video)`

Set qp factor between P and B frames.

`rc_strategy integer (encoding,video)`

Set ratecontrol method.

`b_strategy integer (encoding,video)`

Set strategy to choose between I/P/B-frames.

`ps integer (encoding,video)`

Set RTP payload size in bytes.

`mv_bits integer`  
`header_bits integer`  
`i_tex_bits integer`  
`p_tex_bits integer`  
`i_count integer`  
`p_count integer`  
`skip_count integer`  
`misc_bits integer`  
`frame_bits integer`  
`codec_tag integer`  
`bug flags (decoding,video)`

Workaround not auto detected encoder bugs.

Possible values:

`'autodetect'`  
`'old_msmpeg4'`

some old lavc generated msmpeg4v3 files (no autodetection)

`'xvid_ilace'`

Xvid interlacing bug (autodetected if fourcc==XVIX)

`'ump4'`

(autodetected if fourcc==UMP4)

`'no_padding'`

padding bug (autodetected)

`'amv'`

`'ac_vlc'`

illegal vlc bug (autodetected per fourcc)

`'qpel_chroma'`

`'std_qpel'`

old standard qpel (autodetected per fourcc/version)

`'qpel_chroma2'`

`'direct_blocksize'`

direct-qpel-blocksize bug (autodetected per fourcc/version)

`'edge'`

edge padding bug (autodetected per fourcc/version)

`'hpel_chroma'`

`'dc_clip'`

`'ms'`

Workaround various bugs in microsoft broken decoders.

`'trunc'`

truncated frames

`lelim integer (encoding,video)`

Set single coefficient elimination threshold for luminance (negative values also consider DC coefficient).

`celim integer (encoding,video)`

Set single coefficient elimination threshold for chrominance (negative values also consider dc coefficient)

`strict integer (decoding/encoding,audio,video)`

Specify how strictly to follow the standards.

Possible values:

‘very’

strictly conform to a older more strict version of the spec or reference software

‘strict’

strictly conform to all the things in the spec no matter what consequences

‘normal’

‘unofficial’

allow unofficial extensions

‘experimental’

allow non standardized experimental things, experimental (unfinished/work in progress/not well tested) decoders and encoders. Note: experimental decoders can pose a security risk, do not use this for decoding untrusted input.

`b_qoffset float (encoding,video)`

Set QP offset between P and B frames.

`err_detect flags (decoding,audio,video)`

Set error detection flags.

Possible values:

‘crccheck’

verify embedded CRCs

‘bitstream’

detect bitstream specification deviations

‘buffer’

detect improper bitstream length

`'explode'`

abort decoding on minor error detection

`'ignore_err'`

ignore decoding errors, and continue decoding. This is useful if you want to analyze the content of a video and thus want everything to be decoded no matter what. This option will not result in a video that is pleasing to watch in case of errors.

`'careful'`

consider things that violate the spec and have not been seen in the wild as errors

`'compliant'`

consider all spec non compliances as errors

`'aggressive'`

consider things that a sane encoder should not do as an error

`has_b_frames integer`

`block_align integer`

`mpeg_quant integer (encoding,video)`

Use MPEG quantizers instead of H.263.

`qsquish float (encoding,video)`

How to keep quantizer between qmin and qmax (0 = clip, 1 = use differentiable function).

`rc_qmod_amp float (encoding,video)`

Set experimental quantizer modulation.

`rc_qmod_freq integer (encoding,video)`

Set experimental quantizer modulation.

`rc_override_count integer`

`rc_eq string (encoding,video)`

Set rate control equation. When computing the expression, besides the standard functions defined in the section 'Expression Evaluation', the following functions are available: bits2qp(bits), qp2bits(qp). Also the following constants are available: iTex pTex tex mv fCode iCount mcVar var isI isP isB avgQP qComp avgIITex avgPITex avgPPTex avgBPTex avgTex.

`maxrate integer (encoding, audio, video)`

Set max bitrate tolerance (in bits/s). Requires bufsize to be set.

`minrate integer (encoding, audio, video)`

Set min bitrate tolerance (in bits/s). Most useful in setting up a CBR encode. It is of little use otherwise.

`bufsize integer (encoding, audio, video)`

Set ratecontrol buffer size (in bits).

`rc_buf_aggressivity float (encoding, video)`

Currently useless.

`i_qfactor float (encoding, video)`

Set QP factor between P and I frames.

`i_qoffset float (encoding, video)`

Set QP offset between P and I frames.

`rc_init_cplx float (encoding, video)`

Set initial complexity for 1-pass encoding.

`dct integer (encoding, video)`

Set DCT algorithm.

Possible values:

‘auto’

    autoselect a good one (default)

‘fastint’

    fast integer

‘int’

    accurate integer

‘mmx’

`'altivec'`  
`'faan'`

floating point AAN DCT

`lumi_mask float (encoding,video)`

Compress bright areas stronger than medium ones.

`tcplx_mask float (encoding,video)`

Set temporal complexity masking.

`scplx_mask float (encoding,video)`

Set spatial complexity masking.

`p_mask float (encoding,video)`

Set inter masking.

`dark_mask float (encoding,video)`

Compress dark areas stronger than medium ones.

`idct integer (decoding/encoding,video)`

Select IDCT implementation.

Possible values:

`'auto'`  
`'int'`  
`'simple'`  
`'simplemmx'`  
`'simpleauto'`

Automatically pick a IDCT compatible with the simple one

`'arm'`  
`'altivec'`  
`'sh4'`  
`'simplearm'`  
`'simplearmv5te'`  
`'simplearmv6'`  
`'simpleneon'`  
`'simplealpha'`



`'ipp'`  
`'xvidmmx'`  
`'faani'`

floating point AAN IDCT

`slice_count integer`  
`ec flags (decoding,video)`

Set error concealment strategy.

Possible values:

`'guess_mvs'`

iterative motion vector (MV) search (slow)

`'deblock'`

use strong deblock filter for damaged MBs

`'favor_inter'`

favor predicting from the previous frame instead of the current

`bits_per_coded_sample integer`  
`pred integer (encoding,video)`

Set prediction method.

Possible values:

`'left'`

`'plane'`

`'median'`

`aspect rational number (encoding,video)`

Set sample aspect ratio.

`debug flags (decoding/encoding,audio,video,subtitles)`

Print specific debug info.

Possible values:

`'pict'`

picture info

`'rc'`

rate control

`'bitstream'`

`'mb_type'`

macroblock (MB) type

`'qp'`

per-block quantization parameter (QP)

`'mv'`

motion vector

`'dct_coeff'`

`'green_metadata'`

display complexity metadata for the upcoming frame, GoP or for a given duration.

`'skip'`

`'startcode'`

`'pts'`

`'er'`

error recognition

`'mmco'`

memory management control operations (H.264)

`'bugs'`

`'vis_qp'`

visualize quantization parameter (QP), lower QP are tinted greener

`'vis_mb_type'`

visualize block types

`'buffers'`

picture buffer allocations

`'thread_ops'`

threading operations

‘nomc’

skip motion compensation

`vismv integer (decoding,video)`

Visualize motion vectors (MVs).

This option is deprecated, see the codecview filter instead.

Possible values:

‘pf’

forward predicted MVs of P-frames

‘bf’

forward predicted MVs of B-frames

‘bb’

backward predicted MVs of B-frames

`cmp integer (encoding,video)`

Set full pel me compare function.

Possible values:

‘sad’

sum of absolute differences, fast (default)

‘sse’

sum of squared errors

‘satd’

sum of absolute Hadamard transformed differences

‘dct’

sum of absolute DCT transformed differences

‘psnr’

sum of squared quantization errors (avoid, low quality)

‘bit’

number of bits needed for the block

‘rd’

rate distortion optimal, slow

‘zero’

0

‘vsad’

sum of absolute vertical differences

‘vsse’

sum of squared vertical differences

‘nsse’

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`subcmp integer (encoding, video)`

Set sub pel me compare function.

Possible values:

‘sad’

sum of absolute differences, fast (default)

‘sse’

sum of squared errors

‘satd’

sum of absolute Hadamard transformed differences

‘dct’

sum of absolute DCT transformed differences

‘psnr’

sum of squared quantization errors (avoid, low quality)

‘bit’

number of bits needed for the block

‘rd’

rate distortion optimal, slow

‘zero’

0

‘vsad’

sum of absolute vertical differences

‘vsse’

sum of squared vertical differences

‘nsse’

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`mbcmp integer (encoding, video)`

Set macroblock compare function.

Possible values:

`'sad'`

sum of absolute differences, fast (default)

`'sse'`

sum of squared errors

`'satd'`

sum of absolute Hadamard transformed differences

`'dct'`

sum of absolute DCT transformed differences

`'psnr'`

sum of squared quantization errors (avoid, low quality)

`'bit'`

number of bits needed for the block

`'rd'`

rate distortion optimal, slow

`'zero'`

0

`'vsad'`

sum of absolute vertical differences

`'vsse'`

sum of squared vertical differences

`'nsse'`

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`ildctcmp integer (encoding, video)`

Set interlaced dct compare function.

Possible values:

‘sad’

sum of absolute differences, fast (default)

‘sse’

sum of squared errors

‘satd’

sum of absolute Hadamard transformed differences

‘dct’

sum of absolute DCT transformed differences

‘psnr’

sum of squared quantization errors (avoid, low quality)

‘bit’

number of bits needed for the block

‘rd’

rate distortion optimal, slow

‘zero’

0

‘vsad’

sum of absolute vertical differences

‘vsse’

sum of squared vertical differences

‘nsse’

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`dia_size integer (encoding,video)`

Set diamond type & size for motion estimation.

`last_pred integer (encoding,video)`

Set amount of motion predictors from the previous frame.

`preme integer (encoding,video)`

Set pre motion estimation.

`precmp integer (encoding,video)`

Set pre motion estimation compare function.

Possible values:

‘sad’

sum of absolute differences, fast (default)

‘sse’



sum of squared errors

‘satd’

sum of absolute Hadamard transformed differences

‘dct’

sum of absolute DCT transformed differences

‘psnr’

sum of squared quantization errors (avoid, low quality)

‘bit’

number of bits needed for the block

‘rd’

rate distortion optimal, slow

‘zero’

0

‘vsad’

sum of absolute vertical differences

‘vsse’

sum of squared vertical differences

‘nsse’

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`pre_dia_size integer (encoding,video)`

Set diamond type & size for motion estimation pre-pass.

`subq integer (encoding,video)`

Set sub pel motion estimation quality.

`dtg_active_format integer`

`me_range integer (encoding,video)`

Set limit motion vectors range (1023 for DivX player).

`ibias integer (encoding,video)`

Set intra quant bias.

`pbias integer (encoding,video)`

Set inter quant bias.

`color_table_id integer`

`global_quality integer (encoding,audio,video)`

`coder integer (encoding,video)`

Possible values:

‘vlc’

variable length coder / huffman coder

‘ac’

arithmetic coder

‘raw’

raw (no encoding)

‘rle’

run-length coder

‘deflate’

deflate-based coder

`context integer (encoding,video)`

Set context model.

```
slice_flags integer  
xvmc_acceleration integer  
mbd integer (encoding,video)
```

Set macroblock decision algorithm (high quality mode).

Possible values:

‘simple’

use mbcmp (default)

‘bits’

use fewest bits

‘rd’

use best rate distortion

```
stream_codec_tag integer  
sc_threshold integer (encoding,video)
```

Set scene change threshold.

```
lmin integer (encoding,video)
```

Set min lagrange factor (VBR).

```
lmax integer (encoding,video)
```

Set max lagrange factor (VBR).

```
nr integer (encoding,video)
```

Set noise reduction.

```
rc_init_occupancy integer (encoding,video)
```

Set number of bits which should be loaded into the rc buffer before decoding starts.

```
flags2 flags (decoding/encoding,audio,video)
```

Possible values:

‘fast’

Allow non spec compliant speedup tricks.

‘sgop’

Deprecated, use mpegvideo private options instead.

‘noout’

Skip bitstream encoding.

‘ignorecrop’

Ignore cropping information from sps.

‘local\_header’

Place global headers at every keyframe instead of in extradata.

‘chunks’

Frame data might be split into multiple chunks.

‘showall’

Show all frames before the first keyframe.

‘skipprd’

Deprecated, use mpegvideo private options instead.

‘export\_mvs’

Export motion vectors into frame side-data (see AV\_FRAME\_DATA\_MOTION\_VECTORS) for codecs that support it. See also `doc/examples/export_mvs.c`.

*error integer (encoding,video)*

*qns integer (encoding,video)*

Deprecated, use mpegvideo private options instead.

*threads integer (decoding/encoding,video)*

Possible values:

‘auto’

detect a good number of threads

`me_threshold integer (encoding,video)`

Set motion estimation threshold.

`mb_threshold integer (encoding,video)`

Set macroblock threshold.

`dc integer (encoding,video)`

Set intra\_dc\_precision.

`nssew integer (encoding,video)`

Set nsse weight.

`skip_top integer (decoding,video)`

Set number of macroblock rows at the top which are skipped.

`skip_bottom integer (decoding,video)`

Set number of macroblock rows at the bottom which are skipped.

`profile integer (encoding,audio,video)`

Possible values:

‘unknown’  
‘aac\_main’  
‘aac\_low’  
‘aac\_ssr’  
‘aac\_ltp’  
‘aac\_he’  
‘aac\_he\_v2’  
‘aac\_ld’  
‘aac\_eld’  
‘mpeg2\_aac\_low’  
‘mpeg2\_aac\_he’  
‘mpeg4\_sp’  
‘mpeg4\_core’  
‘mpeg4\_main’  
‘mpeg4\_asp’  
‘dts’  
‘dts\_es’  
‘dts\_96\_24’  
‘dts\_hd\_hra’

`'dts_hd_ma'`  
`level integer (encoding, audio, video)`

Possible values:

`'unknown'`  
`lowres integer (decoding, audio, video)`

Decode at 1= 1/2, 2=1/4, 3=1/8 resolutions.

`skip_threshold integer (encoding, video)`

Set frame skip threshold.

`skip_factor integer (encoding, video)`

Set frame skip factor.

`skip_exp integer (encoding, video)`

Set frame skip exponent. Negative values behave identical to the corresponding positive ones, except that the score is normalized. Positive values exist primarily for compatibility reasons and are not so useful.

`skipcmp integer (encoding, video)`

Set frame skip compare function.

Possible values:

`'sad'`  
sum of absolute differences, fast (default)

`'sse'`  
sum of squared errors

`'satd'`  
sum of absolute Hadamard transformed differences

`'dct'`  
sum of absolute DCT transformed differences

`'psnr'`

sum of squared quantization errors (avoid, low quality)

`'bit'`

number of bits needed for the block

`'rd'`

rate distortion optimal, slow

`'zero'`

0

`'vsad'`

sum of absolute vertical differences

`'vsse'`

sum of squared vertical differences

`'nsse'`

noise preserving sum of squared differences

`'w53'`

5/3 wavelet, only used in snow

`'w97'`

9/7 wavelet, only used in snow

`'dctmax'`

`'chroma'`

`border_mask float (encoding, video)`

Increase the quantizer for macroblocks close to borders.

`mb_lmin integer (encoding, video)`

Set min macroblock lagrange factor (VBR).

`mb_lmax integer (encoding, video)`

Set max macroblock lagrange factor (VBR).

`mepc integer (encoding,video)`

Set motion estimation bitrate penalty compensation ( $1.0 = 256$ ).

`skip_loop_filter integer (decoding,video)`

`skip_idct integer (decoding,video)`

`skip_frame integer (decoding,video)`

Make decoder discard processing depending on the frame type selected by the option value.

`skip_loop_filter` skips frame loop filtering, `skip_idct` skips frame IDCT/dequantization, `skip_frame` skips decoding.

Possible values:

‘none’

Discard no frame.

‘default’

Discard useless frames like 0-sized frames.

‘noref’

Discard all non-reference frames.

‘bidir’

Discard all bidirectional frames.

‘nokey’

Discard all frames excepts keyframes.

‘all’

Discard all frames.

Default value is ‘default’.

`bidir_refine integer (encoding,video)`

Refine the two motion vectors used in bidirectional macroblocks.

`brd_scale integer (encoding,video)`

Downscale frames for dynamic B-frame decision.



`keyint_min integer (encoding,video)`

Set minimum interval between IDR-frames.

`refs integer (encoding,video)`

Set reference frames to consider for motion compensation.

`chromaoffset integer (encoding,video)`

Set chroma qp offset from luma.

`trellis integer (encoding,audio,video)`

Set rate-distortion optimal quantization.

`sc_factor integer (encoding,video)`

Set value multiplied by qscale for each frame and added to scene\_change\_score.

`mv0_threshold integer (encoding,video)`

`b_sensitivity integer (encoding,video)`

Adjust sensitivity of b\_frame\_strategy 1.

`compression_level integer (encoding,audio,video)`

`min_prediction_order integer (encoding,audio)`

`max_prediction_order integer (encoding,audio)`

`timecode_frame_start integer (encoding,video)`

Set GOP timecode frame start number, in non drop frame format.

`request_channels integer (decoding,audio)`

Set desired number of audio channels.

`bits_per_raw_sample integer`

`channel_layout integer (decoding/encoding,audio)`

Possible values:

`request_channel_layout integer (decoding,audio)`

Possible values:

`rc_max_vbv_use float (encoding,video)`

`rc_min_vbv_use float (encoding,video)`

`ticks_per_frame integer (decoding/encoding,audio,video)`

`color_primaries integer (decoding/encoding,video)`  
`color_trc integer (decoding/encoding,video)`  
`colospace integer (decoding/encoding,video)`  
`color_range integer (decoding/encoding,video)`

If used as input parameter, it serves as a hint to the decoder, which color\_range the input has.

`chroma_sample_location integer (decoding/encoding,video)`  
`log_level_offset integer`

Set the log level offset.

`slices integer (encoding,video)`

Number of slices, used in parallelized encoding.

`thread_type flags (decoding/encoding,video)`

Select which multithreading methods to use.

Use of 'frame' will increase decoding delay by one frame per thread, so clients which cannot provide future frames should not use it.

Possible values:

'slice'

Decode more than one part of a single frame at once.

Multithreading using slices works only when the video was encoded with slices.

'frame'

Decode more than one frame at once.

Default value is 'slice+frame'.

`audio_service_type integer (encoding,audio)`

Set audio service type.

Possible values:

'ma'

Main Audio Service

'ef'

Effects

‘vi’

Visually Impaired

‘hi’

Hearing Impaired

‘di’

Dialogue

‘co’

Commentary

‘em’

Emergency

‘vo’

Voice Over

‘ka’

Karaoke

`request_sample_fmt sample_fmt (decoding, audio)`

Set sample format audio decoders should prefer. Default value is none.

`pkt_timebase rational number`

`sub_charenc encoding (decoding, subtitles)`

Set the input subtitles character encoding.

`field_order field_order (video)`

Set/override the field order of the video. Possible values:

‘progressive’

Progressive video

‘tt’

Interlaced video, top field coded and displayed first

‘bb’

Interlaced video, bottom field coded and displayed first

‘tb’

Interlaced video, top coded first, bottom displayed first

‘bt’

Interlaced video, bottom coded first, top displayed first

`skip_alpha integer (decoding,video)`

Set to 1 to disable processing alpha (transparency). This works like the ‘gray’ flag in the `flags` option which skips chroma information instead of alpha. Default is 0.

`codec_whitelist list (input)`

"," separated List of allowed decoders. By default all are allowed.

`dump_separator string (input)`

Separator used to separate the fields printed on the command line about the Stream parameters. For example to separate the fields with newlines and indentation:

```
ffprobe -dump_separator "
                        " -i ~/videos/matrixbench_mpeg2.mpg
```

## 11 Decoders# TOC

Decoders are configured elements in FFmpeg which allow the decoding of multimedia streams.

When you configure your FFmpeg build, all the supported native decoders are enabled by default. Decoders requiring an external library must be enabled manually via the corresponding `--enable-lib` option. You can list all available decoders using the configure option `--list-decoders`.

You can disable all the decoders with the configure option `--disable-decoders` and selectively enable / disable single decoders with the options `--enable-decoder=DECODER` / `--disable-decoder=DECODER`.

The option `-decoders` of the ff\* tools will display the list of enabled decoders.

## 12 Video Decoders# TOC

A description of some of the currently available video decoders follows.

### 12.1 hevc# TOC

HEVC / H.265 decoder.

Note: the `skip_loop_filter` option has effect only at level `all`.

### 12.2 rawvideo# TOC

Raw video decoder.

This decoder decodes rawvideo streams.

#### 12.2.1 Options# TOC

`top top_field_first`

Specify the assumed field type of the input video.

-1

the video is assumed to be progressive (default)

0

bottom-field-first is assumed

1

top-field-first is assumed

## 13 Audio Decoders# TOC

A description of some of the currently available audio decoders follows.

### 13.1 ac3# TOC

AC-3 audio decoder.

This decoder implements part of ATSC A/52:2010 and ETSI TS 102 366, as well as the undocumented RealAudio 3 (a.k.a. dnet).

### 13.1.1 AC-3 Decoder Options# TOC

`-drc_scale value`

Dynamic Range Scale Factor. The factor to apply to dynamic range values from the AC-3 stream. This factor is applied exponentially. There are 3 notable scale factor ranges:

`drc_scale == 0`

DRC disabled. Produces full range audio.

`0 < drc_scale <= 1`

DRC enabled. Applies a fraction of the stream DRC value. Audio reproduction is between full range and full compression.

`drc_scale > 1`

DRC enabled. Applies `drc_scale` asymmetrically. Loud sounds are fully compressed. Soft sounds are enhanced.

## 13.2 flac# TOC

FLAC audio decoder.

This decoder aims to implement the complete FLAC specification from Xiph.

### 13.2.1 FLAC Decoder options# TOC

`-use_buggy_lpc`

The lavc FLAC encoder used to produce buggy streams with high lpc values (like the default value). This option makes it possible to decode such streams correctly by using lavc's old buggy lpc logic for decoding.

## 13.3 ffwavesynth# TOC

Internal wave synthesizer.

This decoder generates wave patterns according to predefined sequences. Its use is purely internal and the format of the data it accepts is not publicly documented.

## 13.4 libcelt# TOC

libcelt decoder wrapper.

libcelt allows libavcodec to decode the Xiph CELT ultra-low delay audio codec. Requires the presence of the libcelt headers and library during configuration. You need to explicitly configure the build with `--enable-libcelt`.

## 13.5 libgsm# TOC

libgsm decoder wrapper.

libgsm allows libavcodec to decode the GSM full rate audio codec. Requires the presence of the libgsm headers and library during configuration. You need to explicitly configure the build with `--enable-libgsm`.

This decoder supports both the ordinary GSM and the Microsoft variant.

## 13.6 libilbc# TOC

libilbc decoder wrapper.

libilbc allows libavcodec to decode the Internet Low Bitrate Codec (iLBC) audio codec. Requires the presence of the libilbc headers and library during configuration. You need to explicitly configure the build with `--enable-libilbc`.

### 13.6.1 Options# TOC

The following option is supported by the libilbc wrapper.

enhance

Enable the enhancement of the decoded audio when set to 1. The default value is 0 (disabled).

## 13.7 libopencore-amrnb# TOC

libopencore-amrnb decoder wrapper.

libopencore-amrnb allows libavcodec to decode the Adaptive Multi-Rate Narrowband audio codec. Using it requires the presence of the libopencore-amrnb headers and library during configuration. You need to explicitly configure the build with `--enable-libopencore-amrnb`.

An FFmpeg native decoder for AMR-NB exists, so users can decode AMR-NB without this library.

## 13.8 libopencore-amrwb# TOC

libopencore-amrwb decoder wrapper.

libopencore-amrwb allows libavcodec to decode the Adaptive Multi-Rate Wideband audio codec. Using it requires the presence of the libopencore-amrwb headers and library during configuration. You need to explicitly configure the build with `--enable-libopencore-amrwb`.

An FFmpeg native decoder for AMR-WB exists, so users can decode AMR-WB without this library.

## 13.9 libopus# TOC

libopus decoder wrapper.

libopus allows libavcodec to decode the Opus Interactive Audio Codec. Requires the presence of the libopus headers and library during configuration. You need to explicitly configure the build with `--enable-libopus`.

An FFmpeg native decoder for Opus exists, so users can decode Opus without this library.

## 14 Subtitles Decoders# TOC

### 14.1 dvbsub# TOC

#### 14.1.1 Options# TOC

`compute_clut`

-1

Compute clut if no matching CLUT is in the stream.

0

Never compute CLUT

1

Always compute CLUT and override the one provided in the stream.

`dvb_substream`

Selects the dvb substream, or all substreams if -1 which is default.

### 14.2 dvdsub# TOC

This codec decodes the bitmap subtitles used in DVDs; the same subtitles can also be found in VobSub file pairs and in some Matroska files.

#### 14.2.1 Options# TOC

`palette`

Specify the global palette used by the bitmaps. When stored in VobSub, the palette is normally specified in the index file; in Matroska, the palette is stored in the codec extra-data in the same format as in VobSub. In DVDs, the palette is stored in the IFO file, and therefore not available when reading from dumped VOB files.



The format for this option is a string containing 16 24-bits hexadecimal numbers (without 0x prefix) separated by comas, for example 0d00ee, ee450d, 101010, eaeaea, 0ce60b, ec14ed, ebff0b, 0d617a, 7b7b7b, d1d1d1, 7b2a0e, 0d950c, 0f007b, cf0dec, cfa80c, 7c127b.

ifo\_palette

Specify the IFO file from which the global palette is obtained. (experimental)

forced\_subs\_only

Only decode subtitle entries marked as forced. Some titles have forced and non-forced subtitles in the same track. Setting this flag to 1 will only keep the forced subtitles. Default value is 0.

## 14.3 libzvbi-teletext# TOC

Libzvbi allows libavcodec to decode DVB teletext pages and DVB teletext subtitles. Requires the presence of the libzvbi headers and library during configuration. You need to explicitly configure the build with `--enable-libzvbi`.

### 14.3.1 Options# TOC

txt\_page

List of teletext page numbers to decode. You may use the special \* string to match all pages. Pages that do not match the specified list are dropped. Default value is \*.

txt\_chop\_top

Discards the top teletext line. Default value is 1.

txt\_format

Specifies the format of the decoded subtitles. The teletext decoder is capable of decoding the teletext pages to bitmaps or to simple text, you should use "bitmap" for teletext pages, because certain graphics and colors cannot be expressed in simple text. You might use "text" for teletext based subtitles if your application can handle simple text based subtitles. Default value is bitmap.

txt\_left

X offset of generated bitmaps, default is 0.

txt\_top

Y offset of generated bitmaps, default is 0.

txt\_chop\_spaces

Chops leading and trailing spaces and removes empty lines from the generated text. This option is useful for teletext based subtitles where empty spaces may be present at the start or at the end of the lines or empty lines may be present between the subtitle lines because of double-sized teletext characters. Default value is 1.

`txt_duration`

Sets the display duration of the decoded teletext pages or subtitles in milliseconds. Default value is 30000 which is 30 seconds.

`txt_transparent`

Force transparent background of the generated teletext bitmaps. Default value is 0 which means an opaque (black) background.

## 15 Encoders# TOC

Encoders are configured elements in FFmpeg which allow the encoding of multimedia streams.

When you configure your FFmpeg build, all the supported native encoders are enabled by default. Encoders requiring an external library must be enabled manually via the corresponding `--enable-lib` option. You can list all available encoders using the configure option `--list-encoders`.

You can disable all the encoders with the configure option `--disable-encoders` and selectively enable / disable single encoders with the options `--enable-encoder=ENCODER` / `--disable-encoder=ENCODER`.

The option `-encoders` of the `ff*` tools will display the list of enabled encoders.

## 16 Audio Encoders# TOC

A description of some of the currently available audio encoders follows.

### 16.1 aac# TOC

Advanced Audio Coding (AAC) encoder.

This encoder is an experimental FFmpeg-native AAC encoder. Currently only the low complexity (AAC-LC) profile is supported. To use this encoder, you must set `strict` option to 'experimental' or lower.

As this encoder is experimental, unexpected behavior may exist from time to time. For a more stable AAC encoder, see `libvo-aacenc`. However, be warned that it has a worse quality reported by some users.

See also `libfdk_aac` and `libfaac`.

## 16.1.1 Options# TOC

b

Set bit rate in bits/s. Setting this automatically activates constant bit rate (CBR) mode.

q

Set quality for variable bit rate (VBR) mode. This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

stereo\_mode

Set stereo encoding mode. Possible values:

‘auto’

Automatically selected by the encoder.

‘ms\_off’

Disable middle/side encoding. This is the default.

‘ms\_force’

Force middle/side encoding.

aac\_coder

Set AAC encoder coding method. Possible values:

‘faac’

FAAC-inspired method.

This method is a simplified reimplementation of the method used in FAAC, which sets thresholds proportional to the band energies, and then decreases all the thresholds with quantizer steps to find the appropriate quantization with distortion below threshold band by band.

The quality of this method is comparable to the two loop searching method described below, but somewhat a little better and slower.

‘anmr’

Average noise to mask ratio (ANMR) trellis-based solution.

This has a theoretic best quality out of all the coding methods, but at the cost of the slowest speed.

`'twoloop'`

Two loop searching (TLS) method.

This method first sets quantizers depending on band thresholds and then tries to find an optimal combination by adding or subtracting a specific value from all quantizers and adjusting some individual quantizer a little.

This method produces similar quality with the FAAC method and is the default.

`'fast'`

Constant quantizer method.

This method sets a constant quantizer for all bands. This is the fastest of all the methods, yet produces the worst quality.

## 16.2 ac3 and ac3\_fixed# TOC

AC-3 audio encoders.

These encoders implement part of ATSC A/52:2010 and ETSI TS 102 366, as well as the undocumented RealAudio 3 (a.k.a. dnet).

The *ac3* encoder uses floating-point math, while the *ac3\_fixed* encoder only uses fixed-point integer math. This does not mean that one is always faster, just that one or the other may be better suited to a particular system. The floating-point encoder will generally produce better quality audio for a given bitrate. The *ac3\_fixed* encoder is not the default codec for any of the output formats, so it must be specified explicitly using the option `-acodec ac3_fixed` in order to use it.

### 16.2.1 AC-3 Metadata# TOC

The AC-3 metadata options are used to set parameters that describe the audio, but in most cases do not affect the audio encoding itself. Some of the options do directly affect or influence the decoding and playback of the resulting bitstream, while others are just for informational purposes. A few of the options will add bits to the output stream that could otherwise be used for audio data, and will thus affect the quality of the output. Those will be indicated accordingly with a note in the option list below.

These parameters are described in detail in several publicly-available documents.

- A/52:2010 - Digital Audio Compression (AC-3) (E-AC-3) Standard
- A/54 - Guide to the Use of the ATSC Digital Television Standard
- Dolby Metadata Guide
- Dolby Digital Professional Encoding Guidelines

### 16.2.1.1 Metadata Control Options# TOC

`-per_frame_metadata` *boolean*

Allow Per-Frame Metadata. Specifies if the encoder should check for changing metadata for each frame.

0

The metadata values set at initialization will be used for every frame in the stream. (default)

1

Metadata values can be changed before encoding each frame.

### 16.2.1.2 Downmix Levels# TOC

`-center_mixlev` *level*

Center Mix Level. The amount of gain the decoder should apply to the center channel when downmixing to stereo. This field will only be written to the bitstream if a center channel is present. The value is specified as a scale factor. There are 3 valid values:

0.707

Apply -3dB gain

0.595

Apply -4.5dB gain (default)

0.500

Apply -6dB gain

`-surround_mixlev` *level*

Surround Mix Level. The amount of gain the decoder should apply to the surround channel(s) when downmixing to stereo. This field will only be written to the bitstream if one or more surround channels are present. The value is specified as a scale factor. There are 3 valid values:

0.707

Apply -3dB gain

0.500

Apply -6dB gain (default)

0.000

Silence Surround Channel(s)

### 16.2.1.3 Audio Production Information# TOC

Audio Production Information is optional information describing the mixing environment. Either none or both of the fields are written to the bitstream.

`-mixing_level number`

Mixing Level. Specifies peak sound pressure level (SPL) in the production environment when the mix was mastered. Valid values are 80 to 111, or -1 for unknown or not indicated. The default value is -1, but that value cannot be used if the Audio Production Information is written to the bitstream. Therefore, if the `room_type` option is not the default value, the `mixing_level` option must not be -1.

`-room_type type`

Room Type. Describes the equalization used during the final mixing session at the studio or on the dubbing stage. A large room is a dubbing stage with the industry standard X-curve equalization; a small room has flat equalization. This field will not be written to the bitstream if both the `mixing_level` option and the `room_type` option have the default values.

0

notindicated

Not Indicated (default)

1

large

Large Room

2

small

Small Room

### 16.2.1.4 Other Metadata Options# TOC

`-copyright boolean`

Copyright Indicator. Specifies whether a copyright exists for this audio.

0  
off

No Copyright Exists (default)

1  
on

Copyright Exists

`-dialnorm value`

Dialogue Normalization. Indicates how far the average dialogue level of the program is below digital 100% full scale (0 dBFS). This parameter determines a level shift during audio reproduction that sets the average volume of the dialogue to a preset level. The goal is to match volume level between program sources. A value of -31dB will result in no volume level change, relative to the source volume, during audio reproduction. Valid values are whole numbers in the range -31 to -1, with -31 being the default.

`-dsur_mode mode`

Dolby Surround Mode. Specifies whether the stereo signal uses Dolby Surround (Pro Logic). This field will only be written to the bitstream if the audio stream is stereo. Using this option does **NOT** mean the encoder will actually apply Dolby Surround processing.

0  
notindicated

Not Indicated (default)

1  
off

Not Dolby Surround Encoded

2  
on

Dolby Surround Encoded

`-original boolean`

Original Bit Stream Indicator. Specifies whether this audio is from the original source and not a copy.

0  
off

Not Original Source

1  
on

Original Source (default)

## 16.2.2 Extended Bitstream Information# TOC

The extended bitstream options are part of the Alternate Bit Stream Syntax as specified in Annex D of the A/52:2010 standard. It is grouped into 2 parts. If any one parameter in a group is specified, all values in that group will be written to the bitstream. Default values are used for those that are written but have not been specified. If the mixing levels are written, the decoder will use these values instead of the ones specified in the `center_mixlev` and `surround_mixlev` options if it supports the Alternate Bit Stream Syntax.

### 16.2.2.1 Extended Bitstream Information - Part 1# TOC

`-dmix_mode mode`

Preferred Stereo Downmix Mode. Allows the user to select either Lt/Rt (Dolby Surround) or Lo/Ro (normal stereo) as the preferred stereo downmix mode.

0  
notindicated

Not Indicated (default)

1  
ltrt

Lt/Rt Downmix Preferred

2  
loro

Lo/Ro Downmix Preferred

`-ltrt_cmixlev level`

Lt/Rt Center Mix Level. The amount of gain the decoder should apply to the center channel when downmixing to stereo in Lt/Rt mode.

1.414

Apply +3dB gain



1.189

Apply +1.5dB gain

1.000

Apply 0dB gain

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain (default)

0.500

Apply -6.0dB gain

0.000

Silence Center Channel

`-lttrt_surmixlev level`

Lt/Rt Surround Mix Level. The amount of gain the decoder should apply to the surround channel(s) when downmixing to stereo in Lt/Rt mode.

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain

0.500

Apply -6.0dB gain (default)

0.000

Silence Surround Channel(s)

`-loro_cmixlev level`

Lo/Ro Center Mix Level. The amount of gain the decoder should apply to the center channel when downmixing to stereo in Lo/Ro mode.

1.414

Apply +3dB gain

1.189

Apply +1.5dB gain

1.000

Apply 0dB gain

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain (default)

0.500

Apply -6.0dB gain

0.000

Silence Center Channel

`-loro_surmixlev level`

Lo/Ro Surround Mix Level. The amount of gain the decoder should apply to the surround channel(s) when downmixing to stereo in Lo/Ro mode.

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain

0.500

Apply -6.0dB gain (default)

0.000

Silence Surround Channel(s)

### 16.2.2.2 Extended Bitstream Information - Part 2# TOC

`-dsurex_mode mode`

Dolby Surround EX Mode. Indicates whether the stream uses Dolby Surround EX (7.1 matrixed to 5.1). Using this option does **NOT** mean the encoder will actually apply Dolby Surround EX processing.

0

notindicated

Not Indicated (default)

1

on

Dolby Surround EX Off

2

off

Dolby Surround EX On

`-dheadphone_mode mode`

Dolby Headphone Mode. Indicates whether the stream uses Dolby Headphone encoding (multi-channel matrixed to 2.0 for use with headphones). Using this option does **NOT** mean the encoder will actually apply Dolby Headphone processing.

0

notindicated

Not Indicated (default)

1  
on

Dolby Headphone Off

2  
off

Dolby Headphone On

`-ad_conv_type` *type*

A/D Converter Type. Indicates whether the audio has passed through HDCD A/D conversion.

0  
standard

Standard A/D Converter (default)

1  
hdc

HDCD A/D Converter

### 16.2.3 Other AC-3 Encoding Options# TOC

`-stereo_rematrixing` *boolean*

Stereo Rematrixing. Enables/Disables use of rematrixing for stereo input. This is an optional AC-3 feature that increases quality by selectively encoding the left/right channels as mid/side. This option is enabled by default, and it is highly recommended that it be left as enabled except for testing purposes.

### 16.2.4 Floating-Point-Only AC-3 Encoding Options# TOC

These options are only valid for the floating-point encoder and do not exist for the fixed-point encoder due to the corresponding features not being implemented in fixed-point.

`-channel_coupling` *boolean*

Enables/Disables use of channel coupling, which is an optional AC-3 feature that increases quality by combining high frequency information from multiple channels into a single channel. The per-channel high frequency information is sent with less accuracy in both the frequency and time domains. This allows more bits to be used for lower frequencies while preserving enough information to reconstruct the high frequencies. This option is enabled by default for the floating-point encoder and should

generally be left as enabled except for testing purposes or to increase encoding speed.

-1

auto

Selected by Encoder (default)

0

off

Disable Channel Coupling

1

on

Enable Channel Coupling

`-cpl_start_band number`

Coupling Start Band. Sets the channel coupling start band, from 1 to 15. If a value higher than the bandwidth is used, it will be reduced to 1 less than the coupling end band. If *auto* is used, the start band will be determined by the encoder based on the bit rate, sample rate, and channel layout. This option has no effect if channel coupling is disabled.

-1

auto

Selected by Encoder (default)

## 16.3 flac# TOC

FLAC (Free Lossless Audio Codec) Encoder

### 16.3.1 Options# TOC

The following options are supported by FFmpeg's flac encoder.

`compression_level`

Sets the compression level, which chooses defaults for many other options if they are not set explicitly.

`frame_size`

Sets the size of the frames in samples per channel.

`lpc_coeff_precision`

Sets the LPC coefficient precision, valid values are from 1 to 15, 15 is the default.

`lpc_type`

Sets the first stage LPC algorithm

`'none'`

LPC is not used

`'fixed'`

fixed LPC coefficients

`'levinson'`

`'cholesky'`

`lpc_passes`

Number of passes to use for Cholesky factorization during LPC analysis

`min_partition_order`

The minimum partition order

`max_partition_order`

The maximum partition order

`prediction_order_method`

`'estimation'`

`'2level'`

`'4level'`

`'8level'`

`'search'`

Bruteforce search

`'log'`

`ch_mode`

Channel mode

`'auto'`

The mode is chosen automatically for each frame

`'indep'`

Channels are independently coded

```
'left_side'  
'right_side'  
'mid_side'  
exact_rice_parameters
```

Chooses if rice parameters are calculated exactly or approximately. if set to 1 then they are chosen exactly, which slows the code down slightly and improves compression slightly.

```
multi_dim_quant
```

Multi Dimensional Quantization. If set to 1 then a 2nd stage LPC algorithm is applied after the first stage to finetune the coefficients. This is quite slow and slightly improves compression.

## 16.4 libfaac# TOC

libfaac AAC (Advanced Audio Coding) encoder wrapper.

Requires the presence of the libfaac headers and library during configuration. You need to explicitly configure the build with `--enable-libfaac --enable-nonfree`.

This encoder is considered to be of higher quality with respect to the the native experimental FFmpeg AAC encoder.

For more information see the libfaac project at <http://www.audiocoding.com/faac.html/>.

### 16.4.1 Options# TOC

The following shared FFmpeg codec options are recognized.

The following options are supported by the libfaac wrapper. The `faac`-equivalent of the options are listed in parentheses.

`b (-b)`

Set bit rate in bits/s for ABR (Average Bit Rate) mode. If the bit rate is not explicitly specified, it is automatically set to a suitable value depending on the selected profile. `faac` bitrate is expressed in kilobits/s.

Note that libfaac does not support CBR (Constant Bit Rate) but only ABR (Average Bit Rate).

If VBR mode is enabled this option is ignored.

`ar (-R)`

Set audio sampling rate (in Hz).

`ac (-c)`

Set the number of audio channels.

`cutoff (-C)`

Set cutoff frequency. If not specified (or explicitly set to 0) it will use a value automatically computed by the library. Default value is 0.

`profile`

Set audio profile.

The following profiles are recognized:

`'aac_main'`

Main AAC (Main)

`'aac_low'`

Low Complexity AAC (LC)

`'aac_ssr'`

Scalable Sample Rate (SSR)

`'aac_ltp'`

Long Term Prediction (LTP)

If not specified it is set to `'aac_low'`.

`flags +qscale`

Set constant quality VBR (Variable Bit Rate) mode.

`global_quality`

Set quality in VBR mode as an integer number of lambda units.

Only relevant when VBR mode is enabled with `flags +qscale`. The value is converted to QP units by dividing it by `FF_QP2LAMBDA`, and used to set the quality value used by libfaac. A reasonable range for the option value in QP units is [10-500], the higher the value the higher the quality.

`q (-q)`



Enable VBR mode when set to a non-negative value, and set constant quality value as a double floating point value in QP units.

The value sets the quality value used by libfaac. A reasonable range for the option value is [10-500], the higher the value the higher the quality.

This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

## 16.4.2 Examples# TOC

- Use `ffmpeg` to convert an audio file to ABR 128 kbps AAC in an M4A (MP4) container:

```
ffmpeg -i input.wav -codec:a libfaac -b:a 128k -output.m4a
```

- Use `ffmpeg` to convert an audio file to VBR AAC, using the LTP AAC profile:

```
ffmpeg -i input.wav -c:a libfaac -profile:a aac_ltp -q:a 100 output.m4a
```

## 16.5 libfdk\_aac# TOC

libfdk-aac AAC (Advanced Audio Coding) encoder wrapper.

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

Requires the presence of the libfdk-aac headers and library during configuration. You need to explicitly configure the build with `--enable-libfdk-aac`. The library is also incompatible with GPL, so if you allow the use of GPL, you should configure with `--enable-gpl --enable-nonfree --enable-libfdk-aac`.

This encoder is considered to be of higher quality with respect to both the native experimental FFmpeg AAC encoder and libfaac.

VBR encoding, enabled through the `vbr` or `flags +qscale` options, is experimental and only works with some combinations of parameters.

Support for encoding 7.1 audio is only available with libfdk-aac 0.1.3 or higher.

For more information see the fdk-aac project at <http://sourceforge.net/p/opencore-amr/fdk-aac/>.

### 16.5.1 Options# TOC

The following options are mapped on the shared FFmpeg codec options.

b

Set bit rate in bits/s. If the bitrate is not explicitly specified, it is automatically set to a suitable value depending on the selected profile.

In case VBR mode is enabled the option is ignored.

ar

Set audio sampling rate (in Hz).

channels

Set the number of audio channels.

flags +qscale

Enable fixed quality, VBR (Variable Bit Rate) mode. Note that VBR is implicitly enabled when the vbr value is positive.

cutoff

Set cutoff frequency. If not specified (or explicitly set to 0) it will use a value automatically computed by the library. Default value is 0.

profile

Set audio profile.

The following profiles are recognized:

‘aac\_low’

Low Complexity AAC (LC)

‘aac\_he’

High Efficiency AAC (HE-AAC)

‘aac\_he\_v2’

High Efficiency AAC version 2 (HE-AACv2)

‘aac\_ld’

Low Delay AAC (LD)

‘aac\_eld’

Enhanced Low Delay AAC (ELD)

If not specified it is set to ‘aac\_low’.

The following are private options of the libfdk\_aac encoder.

#### afterburner

Enable afterburner feature if set to 1, disabled if set to 0. This improves the quality but also the required processing power.

Default value is 1.

#### eld\_sbr

Enable SBR (Spectral Band Replication) for ELD if set to 1, disabled if set to 0.

Default value is 0.

#### signaling

Set SBR/PS signaling style.

It can assume one of the following values:

‘default’

choose signaling implicitly (explicit hierarchical by default, implicit if global header is disabled)

‘implicit’

implicit backwards compatible signaling

‘explicit\_sbr’

explicit SBR, implicit PS signaling

‘explicit\_hierarchical’

explicit hierarchical signaling

Default value is ‘default’.

#### latm

Output LATM/LOAS encapsulated data if set to 1, disabled if set to 0.

Default value is 0.

#### header\_period

Set StreamMuxConfig and PCE repetition period (in frames) for sending in-band configuration buffers within LATM/LOAS transport layer.

Must be a 16-bits non-negative integer.

Default value is 0.

vbr

Set VBR mode, from 1 to 5. 1 is lowest quality (though still pretty good) and 5 is highest quality. A value of 0 will disable VBR, and CBR (Constant Bit Rate) is enabled.

Currently only the 'aac\_low' profile supports VBR encoding.

VBR modes 1-5 correspond to roughly the following average bit rates:

'1'

32 kbps/channel

'2'

40 kbps/channel

'3'

48-56 kbps/channel

'4'

64 kbps/channel

'5'

about 80-96 kbps/channel

Default value is 0.

## 16.5.2 Examples# TOC

- Use `ffmpeg` to convert an audio file to VBR AAC in an M4A (MP4) container:

```
ffmpeg -i input.wav -codec:a libfdk_aac -vbr 3 output.m4a
```

- Use `ffmpeg` to convert an audio file to CBR 64k kbps AAC, using the High-Efficiency AAC profile:

```
ffmpeg -i input.wav -c:a libfdk_aac -profile:a aac_he -b:a 64k output.m4a
```

## 16.6 libmp3lame# TOC

LAME (Lame Ain't an MP3 Encoder) MP3 encoder wrapper.

Requires the presence of the libmp3lame headers and library during configuration. You need to explicitly configure the build with `--enable-libmp3lame`.

See libshine for a fixed-point MP3 encoder, although with a lower quality.

### 16.6.1 Options# TOC

The following options are supported by the libmp3lame wrapper. The lame-equivalent of the options are listed in parentheses.

`b (-b)`

Set bitrate expressed in bits/s for CBR or ABR. LAME `bitrate` is expressed in kilobits/s.

`q (-V)`

Set constant quality setting for VBR. This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

`compression_level (-q)`

Set algorithm quality. Valid arguments are integers in the 0-9 range, with 0 meaning highest quality but slowest, and 9 meaning fastest while producing the worst quality.

`reservoir`

Enable use of bit reservoir when set to 1. Default value is 1. LAME has this enabled by default, but can be overridden by use `--nores` option.

`joint_stereo (-m j)`

Enable the encoder to use (on a frame by frame basis) either L/R stereo or mid/side stereo. Default value is 1.

`abr (--abr)`

Enable the encoder to use ABR when set to 1. The lame `--abr` sets the target bitrate, while this options only tells FFmpeg to use ABR still relies on `b` to set bitrate.

## 16.7 libopencore-amrnb# TOC

OpenCORE Adaptive Multi-Rate Narrowband encoder.

Requires the presence of the libopencore-amrnb headers and library during configuration. You need to explicitly configure the build with `--enable-libopencore-amrnb --enable-version3`.

This is a mono-only encoder. Officially it only supports 8000Hz sample rate, but you can override it by setting `strict` to 'unofficial' or lower.

### 16.7.1 Options# TOC

b

Set bitrate in bits per second. Only the following bitrates are supported, otherwise libavcodec will round to the nearest valid bitrate.

4750  
5150  
5900  
6700  
7400  
7950  
10200  
12200

dtx

Allow discontinuous transmission (generate comfort noise) when set to 1. The default value is 0 (disabled).

### 16.8 libshine# TOC

Shine Fixed-Point MP3 encoder wrapper.

Shine is a fixed-point MP3 encoder. It has a far better performance on platforms without an FPU, e.g. armel CPUs, and some phones and tablets. However, as it is more targeted on performance than quality, it is not on par with LAME and other production-grade encoders quality-wise. Also, according to the project's homepage, this encoder may not be free of bugs as the code was written a long time ago and the project was dead for at least 5 years.

This encoder only supports stereo and mono input. This is also CBR-only.

The original project (last updated in early 2007) is at <http://sourceforge.net/projects/libshine-fxp/>. We only support the updated fork by the Savonet/Liquidsoap project at <https://github.com/savonet/shine>.

Requires the presence of the libshine headers and library during configuration. You need to explicitly configure the build with `--enable-libshine`.

See also libmp3lame.

## 16.8.1 Options# TOC

The following options are supported by the libshine wrapper. The `shineenc`-equivalent of the options are listed in parentheses.

`b (-b)`

Set bitrate expressed in bits/s for CBR. `shineenc -b` option is expressed in kilobits/s.

## 16.9 libtwolame# TOC

TwoLAME MP2 encoder wrapper.

Requires the presence of the libtwolame headers and library during configuration. You need to explicitly configure the build with `--enable-libtwolame`.

### 16.9.1 Options# TOC

The following options are supported by the libtwolame wrapper. The `twolame`-equivalent options follow the FFmpeg ones and are in parentheses.

`b (-b)`

Set bitrate expressed in bits/s for CBR. `twolame b` option is expressed in kilobits/s. Default value is 128k.

`q (-V)`

Set quality for experimental VBR support. Maximum value range is from -50 to 50, useful range is from -10 to 10. The higher the value, the better the quality. This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

`mode (--mode)`

Set the mode of the resulting audio. Possible values:

`'auto'`

Choose mode automatically based on the input. This is the default.

`'stereo'`

Stereo

`'joint_stereo'`

Joint stereo

`'dual_channel'`

Dual channel

`'mono'`

Mono

`psymodel (--psyc-mode)`

Set psychoacoustic model to use in encoding. The argument must be an integer between -1 and 4, inclusive. The higher the value, the better the quality. The default value is 3.

`energy_levels (--energy)`

Enable energy levels extensions when set to 1. The default value is 0 (disabled).

`error_protection (--protect)`

Enable CRC error protection when set to 1. The default value is 0 (disabled).

`copyright (--copyright)`

Set MPEG audio copyright flag when set to 1. The default value is 0 (disabled).

`original (--original)`

Set MPEG audio original flag when set to 1. The default value is 0 (disabled).

## 16.10 libvo-aacenc# TOC

VisualOn AAC encoder.

Requires the presence of the libvo-aacenc headers and library during configuration. You need to explicitly configure the build with `--enable-libvo-aacenc --enable-version3`.

This encoder is considered to be worse than the native experimental FFmpeg AAC encoder, according to multiple sources.

### 16.10.1 Options# TOC

The VisualOn AAC encoder only support encoding AAC-LC and up to 2 channels. It is also CBR-only.

b

Set bit rate in bits/s.



## 16.11 libvo-amrwbenc# TOC

VisualOn Adaptive Multi-Rate Wideband encoder.

Requires the presence of the libvo-amrwbenc headers and library during configuration. You need to explicitly configure the build with `--enable-libvo-amrwbenc --enable-version3`.

This is a mono-only encoder. Officially it only supports 16000Hz sample rate, but you can override it by setting `strict` to 'unofficial' or lower.

### 16.11.1 Options# TOC

`b`

Set bitrate in bits/s. Only the following bitrates are supported, otherwise libavcodec will round to the nearest valid bitrate.

'6600'  
'8850'  
'12650'  
'14250'  
'15850'  
'18250'  
'19850'  
'23050'  
'23850'

`dtx`

Allow discontinuous transmission (generate comfort noise) when set to 1. The default value is 0 (disabled).

## 16.12 libopus# TOC

libopus Opus Interactive Audio Codec encoder wrapper.

Requires the presence of the libopus headers and library during configuration. You need to explicitly configure the build with `--enable-libopus`.

### 16.12.1 Option Mapping# TOC

Most libopus options are modelled after the `opusenc` utility from `opus-tools`. The following is an option mapping chart describing options supported by the libopus wrapper, and their `opusenc`-equivalent in parentheses.

`b` (*bitrate*)

Set the bit rate in bits/s. FFmpeg's `b` option is expressed in bits/s, while `opusenc`'s `bitrate` in kilobits/s.

`vbr` (*vbr*, *hard-cbr*, and *cvbr*)

Set VBR mode. The FFmpeg `vbr` option has the following valid arguments, with the their `opusenc` equivalent options in parentheses:

`'off (hard-cbr)'`

Use constant bit rate encoding.

`'on (vbr)'`

Use variable bit rate encoding (the default).

`'constrained (cvbr)'`

Use constrained variable bit rate encoding.

`compression_level` (*comp*)

Set encoding algorithm complexity. Valid options are integers in the 0-10 range. 0 gives the fastest encodes but lower quality, while 10 gives the highest quality but slowest encoding. The default is 10.

`frame_duration` (*framesize*)

Set maximum frame size, or duration of a frame in milliseconds. The argument must be exactly the following: 2.5, 5, 10, 20, 40, 60. Smaller frame sizes achieve lower latency but less quality at a given bitrate. Sizes greater than 20ms are only interesting at fairly low bitrates. The default is 20ms.

`packet_loss` (*expect-loss*)

Set expected packet loss percentage. The default is 0.

`application` (N.A.)

Set intended application type. Valid options are listed below:

`'voip'`

Favor improved speech intelligibility.

`'audio'`

Favor faithfulness to the input (the default).

`'lowdelay'`

Restrict to only the lowest delay modes.

`cutoff` (N.A.)

Set cutoff bandwidth in Hz. The argument must be exactly one of the following: 4000, 6000, 8000, 12000, or 20000, corresponding to narrowband, mediumband, wideband, super wideband, and fullband respectively. The default is 0 (cutoff disabled).

## 16.13 libvorbis# TOC

libvorbis encoder wrapper.

Requires the presence of the libvorbisenc headers and library during configuration. You need to explicitly configure the build with `--enable-libvorbis`.

### 16.13.1 Options# TOC

The following options are supported by the libvorbis wrapper. The `oggenc`-equivalent of the options are listed in parentheses.

To get a more accurate and extensive documentation of the libvorbis options, consult the libvorbisenc's and oggenc's documentations. See <http://xiph.org/vorbis/>, <http://wiki.xiph.org/Vorbis-tools>, and `oggenc(1)`.

`b` (`-b`)

Set bitrate expressed in bits/s for ABR. `oggenc -b` is expressed in kilobits/s.

`q` (`-q`)

Set constant quality setting for VBR. The value should be a float number in the range of -1.0 to 10.0. The higher the value, the better the quality. The default value is '3.0'.

This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

`cutoff` (`--advanced-encode-option lowpass_frequency=N`)

Set cutoff bandwidth in Hz, a value of 0 disables cutoff. `oggenc`'s related option is expressed in kHz. The default value is '0' (cutoff disabled).

`minrate` (`-m`)

Set minimum bitrate expressed in bits/s. `oggenc -m` is expressed in kilobits/s.

`maxrate` (`-M`)

Set maximum bitrate expressed in bits/s. `oggenc -M` is expressed in kilobits/s. This only has effect on ABR mode.

`iblock (--advanced-encode-option impulse_noisetune=N)`

Set noise floor bias for impulse blocks. The value is a float number from -15.0 to 0.0. A negative bias instructs the encoder to pay special attention to the crispness of transients in the encoded audio. The tradeoff for better transient response is a higher bitrate.

## 16.14 libwavpack# TOC

A wrapper providing WavPack encoding through libwavpack.

Only lossless mode using 32-bit integer samples is supported currently.

Requires the presence of the libwavpack headers and library during configuration. You need to explicitly configure the build with `--enable-libwavpack`.

Note that a libavcodec-native encoder for the WavPack codec exists so users can encode audios with this codec without using this encoder. See `wavpackenc`.

### 16.14.1 Options# TOC

wavpack command line utility's corresponding options are listed in parentheses, if any.

`frame_size (--blocksize)`

Default is 32768.

`compression_level`

Set speed vs. compression tradeoff. Acceptable arguments are listed below:

`'0 (-f)'`

Fast mode.

`'1'`

Normal (default) settings.

`'2 (-h)'`

High quality.

`'3 (-hh)'`

Very high quality.

`'4-8 (-hh -xEXTRAPROC)'`

Same as '3', but with extra processing enabled.

'4' is the same as -x2 and '8' is the same as -x6.

## 16.15 wavpack# TOC

WavPack lossless audio encoder.

This is a libavcodec-native WavPack encoder. There is also an encoder based on libwavpack, but there is virtually no reason to use that encoder.

See also libwavpack.

### 16.15.1 Options# TOC

The equivalent options for wavpack command line utility are listed in parentheses.

#### 16.15.1.1 Shared options# TOC

The following shared options are effective for this encoder. Only special notes about this particular encoder will be documented here. For the general meaning of the options, see the Codec Options chapter.

`frame_size (--blocksize)`

For this encoder, the range for this option is between 128 and 131072. Default is automatically decided based on sample rate and number of channel.

For the complete formula of calculating default, see `libavcodec/wavpackenc.c`.

`compression_level (-f, -h, -hh, and -x)`

This option's syntax is consistent with libwavpack's.

#### 16.15.1.2 Private options# TOC

`joint_stereo (-j)`

Set whether to enable joint stereo. Valid values are:

`'on (1)'`

Force mid/side audio encoding.

`'off (0)'`

Force left/right audio encoding.

`'auto'`

Let the encoder decide automatically.

`optimize_mono`

Set whether to enable optimization for mono. This option is only effective for non-mono streams.  
Available values:

`'on'`

enabled

`'off'`

disabled

## 17 Video Encoders# TOC

A description of some of the currently available video encoders follows.

### 17.1 jpeg2000# TOC

The native jpeg 2000 encoder is lossy by default, the `-q:v` option can be used to set the encoding quality. Lossless encoding can be selected with `-pred 1`.

#### 17.1.1 Options# TOC

`format`

Can be set to either `j2k` or `jp2` (the default) that makes it possible to store non-rgb `pix_fmts`.

### 17.2 snow# TOC

#### 17.2.1 Options# TOC

`iterative_dia_size`

dia size for the iterative motion estimation

## 17.3 libtheora# TOC

libtheora Theora encoder wrapper.

Requires the presence of the libtheora headers and library during configuration. You need to explicitly configure the build with `--enable-libtheora`.

For more information about the libtheora project see <http://www.theora.org/>.

### 17.3.1 Options# TOC

The following global options are mapped to internal libtheora options which affect the quality and the bitrate of the encoded stream.

`b`

Set the video bitrate in bit/s for CBR (Constant Bit Rate) mode. In case VBR (Variable Bit Rate) mode is enabled this option is ignored.

`flags`

Used to enable constant quality mode (VBR) encoding through the `qscale` flag, and to enable the `pass1` and `pass2` modes.

`g`

Set the GOP size.

`global_quality`

Set the global quality as an integer in lambda units.

Only relevant when VBR mode is enabled with `flags +qscale`. The value is converted to QP units by dividing it by `FF_QP2LAMBDA`, clipped in the [0 - 10] range, and then multiplied by 6.3 to get a value in the native libtheora range [0-63]. A higher value corresponds to a higher quality.

`q`

Enable VBR mode when set to a non-negative value, and set constant quality value as a double floating point value in QP units.

The value is clipped in the [0-10] range, and then multiplied by 6.3 to get a value in the native libtheora range [0-63].

This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

## 17.3.2 Examples# TOC

- Set maximum constant quality (VBR) encoding with `ffmpeg`:

```
ffmpeg -i INPUT -codec:v libtheora -q:v 10 OUTPUT.ogg
```

- Use `ffmpeg` to convert a CBR 1000 kbps Theora video stream:

```
ffmpeg -i INPUT -codec:v libtheora -b:v 1000k OUTPUT.ogg
```

## 17.4 libvpx# TOC

VP8/VP9 format supported through `libvpx`.

Requires the presence of the `libvpx` headers and library during configuration. You need to explicitly configure the build with `--enable-libvpx`.

### 17.4.1 Options# TOC

The following options are supported by the `libvpx` wrapper. The `vpxenc`-equivalent options or values are listed in parentheses for easy migration.

To reduce the duplication of documentation, only the private options and some others requiring special attention are documented here. For the documentation of the undocumented generic options, see the `Codec Options` chapter.

To get more documentation of the `libvpx` options, invoke the command `ffmpeg -h encoder=libvpx`, `ffmpeg -h encoder=libvpx-vp9` or `vpxenc --help`. Further information is available in the `libvpx` API documentation.

`b` (*target-bitrate*)

Set bitrate in bits/s. Note that `FFmpeg`'s `b` option is expressed in bits/s, while `vpxenc`'s `target-bitrate` is in kilobits/s.

`g` (*kf-max-dist*)

`keyint_min` (*kf-min-dist*)

`qmin` (*min-q*)

`qmax` (*max-q*)

`bufsize` (*buf-sz, buf-optimal-sz*)

Set `ratecontrol` buffer size (in bits). Note `vpxenc`'s options are specified in milliseconds, the `libvpx` wrapper converts this value as follows: `buf-sz` = `bufsize` \* 1000 / `bitrate`,  
`buf-optimal-sz` = `bufsize` \* 1000 / `bitrate` \* 5 / 6.

`rc_init_occupancy` (*buf-initial-sz*)



Set number of bits which should be loaded into the rc buffer before decoding starts. Note vpxenc's option is specified in milliseconds, the libvpx wrapper converts this value as follows:

$rc\_init\_occupancy * 1000 / \text{bitrate}$ .

`undershoot-pct`

Set datarate undershoot (min) percentage of the target bitrate.

`overshoot-pct`

Set datarate overshoot (max) percentage of the target bitrate.

`skip_threshold (drop-frame)`

`qcomp (bias-pct)`

`maxrate (maxsection-pct)`

Set GOP max bitrate in bits/s. Note vpxenc's option is specified as a percentage of the target bitrate, the libvpx wrapper converts this value as follows:  $(\text{maxrate} * 100 / \text{bitrate})$ .

`minrate (minsection-pct)`

Set GOP min bitrate in bits/s. Note vpxenc's option is specified as a percentage of the target bitrate, the libvpx wrapper converts this value as follows:  $(\text{minrate} * 100 / \text{bitrate})$ .

`minrate, maxrate, b end-usage=cbr`

$(\text{minrate} == \text{maxrate} == \text{bitrate})$ .

`crf (end-usage=cq, cq-level)`

`quality, deadline (deadline)`

`'best'`

Use best quality deadline. Poorly named and quite slow, this option should be avoided as it may give worse quality output than good.

`'good'`

Use good quality deadline. This is a good trade-off between speed and quality when used with the `cpu-used` option.

`'realtime'`

Use realtime quality deadline.

`speed, cpu-used (cpu-used)`

Set quality/speed ratio modifier. Higher values speed up the encode at the cost of quality.

`nr` (*noise-sensitivity*)  
`static-thresh`

Set a change threshold on blocks below which they will be skipped by the encoder.

`slices` (*token-parts*)

Note that FFmpeg's `slices` option gives the total number of partitions, while vpxenc's `token-parts` is given as  $\log_2(\text{partitions})$ .

`max-intra-rate`

Set maximum I-frame bitrate as a percentage of the target bitrate. A value of 0 means unlimited.

`force_key_frames`

`VPX_EFLAG_FORCE_KF`

Alternate reference frame related  
`auto-alt-ref`

Enable use of alternate reference frames (2-pass only).

`arnr-max-frames`

Set altref noise reduction max frame count.

`arnr-type`

Set altref noise reduction filter type: backward, forward, centered.

`arnr-strength`

Set altref noise reduction filter strength.

`rc-lookahead, lag-in-frames` (*lag-in-frames*)

Set number of frames to look ahead for frametype and ratecontrol.

`error-resilient`

Enable error resiliency features.

VP9-specific options  
`lossless`

Enable lossless mode.

`tile-columns`

Set number of tile columns to use. Note this is given as `log2(tile_columns)`. For example, 8 tile columns would be requested by setting the `tile-columns` option to 3.

`tile-rows`

Set number of tile rows to use. Note this is given as `log2(tile_rows)`. For example, 4 tile rows would be requested by setting the `tile-rows` option to 2.

`frame-parallel`

Enable frame parallel decodability features.

`aq-mode`

Set adaptive quantization mode (0: off (default), 1: variance 2: complexity, 3: cyclic refresh).

`colorspace` *color-space*

Set input color space. The VP9 bitstream supports signaling the following colorspace:

```
'rgb' sRGB
'bt709' bt709
'unspecified' unknown
'bt470bg' bt601
'smpte170m' smpte170
'smpte240m' smpte240
'bt2020_ncl' bt2020
```

For more information about libvpx see: <http://www.webmproject.org/>

## 17.5 libwebp# TOC

libwebp WebP Image encoder wrapper

libwebp is Google's official encoder for WebP images. It can encode in either lossy or lossless mode. Lossy images are essentially a wrapper around a VP8 frame. Lossless images are a separate codec developed by Google.

### 17.5.1 Pixel Format# TOC

Currently, libwebp only supports YUV420 for lossy and RGB for lossless due to limitations of the format and libwebp. Alpha is supported for either mode. Because of API limitations, if RGB is passed in when encoding lossy or YUV is passed in for encoding lossless, the pixel format will automatically be converted using functions from libwebp. This is not ideal and is done only for convenience.

## 17.5.2 Options# TOC

`-lossless boolean`

Enables/Disables use of lossless mode. Default is 0.

`-compression_level integer`

For lossy, this is a quality/speed tradeoff. Higher values give better quality for a given size at the cost of increased encoding time. For lossless, this is a size/speed tradeoff. Higher values give smaller size at the cost of increased encoding time. More specifically, it controls the number of extra algorithms and compression tools used, and varies the combination of these tools. This maps to the *method* option in libwebp. The valid range is 0 to 6. Default is 4.

`-qscale float`

For lossy encoding, this controls image quality, 0 to 100. For lossless encoding, this controls the effort and time spent at compressing more. The default value is 75. Note that for usage via libavcodec, this option is called *global\_quality* and must be multiplied by *FF\_QP2LAMBDA*.

`-preset type`

Configuration preset. This does some automatic settings based on the general type of the image.

`none`

Do not use a preset.

`default`

Use the encoder default.

`picture`

Digital picture, like portrait, inner shot

`photo`

Outdoor photograph, with natural lighting

`drawing`

Hand or line drawing, with high-contrast details

`icon`

Small-sized colorful images

text

Text-like

## 17.6 libx264, libx264rgb# TOC

x264 H.264/MPEG-4 AVC encoder wrapper.

This encoder requires the presence of the libx264 headers and library during configuration. You need to explicitly configure the build with `--enable-libx264`.

libx264 supports an impressive number of features, including 8x8 and 4x4 adaptive spatial transform, adaptive B-frame placement, CAVLC/CABAC entropy coding, interlacing (MBAFF), lossless mode, psy optimizations for detail retention (adaptive quantization, psy-RD, psy-trellis).

Many libx264 encoder options are mapped to FFmpeg global codec options, while unique encoder options are provided through private options. Additionally the `x264opts` and `x264-params` private options allows one to pass a list of key=value tuples as accepted by the libx264 `x264_param_parse` function.

The x264 project website is at <http://www.videolan.org/developers/x264.html>.

The libx264rgb encoder is the same as libx264, except it accepts packed RGB pixel formats as input instead of YUV.

### 17.6.1 Supported Pixel Formats# TOC

x264 supports 8- to 10-bit color spaces. The exact bit depth is controlled at x264's configure time. FFmpeg only supports one bit depth in one particular build. In other words, it is not possible to build one FFmpeg with multiple versions of x264 with different bit depths.

### 17.6.2 Options# TOC

The following options are supported by the libx264 wrapper. The x264-equivalent options or values are listed in parentheses for easy migration.

To reduce the duplication of documentation, only the private options and some others requiring special attention are documented here. For the documentation of the undocumented generic options, see the Codec Options chapter.

To get a more accurate and extensive documentation of the libx264 options, invoke the command `x264 --full-help` or consult the libx264 documentation.

`b` (*bitrate*)

Set bitrate in bits/s. Note that FFmpeg's `b` option is expressed in bits/s, while x264's `bitrate` is in kilobits/s.

`bf (bframes)`  
`g (keyint)`  
`qmin (qpmin)`

Minimum quantizer scale.

`qmax (qpmax)`

Maximum quantizer scale.

`qdiff (qpstep)`

Maximum difference between quantizer scales.

`qblur (qblur)`

Quantizer curve blur

`qcomp (qcomp)`

Quantizer curve compression factor

`refs (ref)`

Number of reference frames each P-frame can use. The range is from 0-16.

`sc_threshold (scenecut)`

Sets the threshold for the scene change detection.

`trellis (trellis)`

Performs Trellis quantization to increase efficiency. Enabled by default.

`nr (nr)`

`me_range (merange)`

Maximum range of the motion search in pixels.

`me_method (me)`

Set motion estimation method. Possible values in the decreasing order of speed:

`'dia (dia)'`  
`'epzs (dia)'`

Diamond search with radius 1 (fastest). `'epzs'` is an alias for `'dia'`.

`'hex (hex)'`

Hexagonal search with radius 2.

`'umh (umh)'`

Uneven multi-hexagon search.

`'esa (esa)'`

Exhaustive search.

`'tesa (tesa)'`

Hadamard exhaustive search (slowest).

`subq (subme)`

Sub-pixel motion estimation method.

`b_strategy (b-adapt)`

Adaptive B-frame placement decision algorithm. Use only on first-pass.

`keyint_min (min-keyint)`

Minimum GOP size.

`coder`

Set entropy encoder. Possible values:

`'ac'`

Enable CABAC.

`'vlc'`

Enable CAVLC and disable CABAC. It generates the same effect as x264's `--no-cabac` option.

`cmp`

Set full pixel motion estimation comparison algorithm. Possible values:

`'chroma'`

Enable chroma in motion estimation.

`'sad'`

Ignore chroma in motion estimation. It generates the same effect as x264's `--no-chroma-me` option.

`threads` (*threads*)

Number of encoding threads.

`thread_type`

Set multithreading technique. Possible values:

`'slice'`

Slice-based multithreading. It generates the same effect as x264's `--sliced-threads` option.

`'frame'`

Frame-based multithreading.

`flags`

Set encoding flags. It can be used to disable closed GOP and enable open GOP by setting it to `-cgop`. The result is similar to the behavior of x264's `--open-gop` option.

`rc_init_occupancy` (*vbv-init*)

`preset` (*preset*)

Set the encoding preset.

`tune` (*tune*)

Set tuning of the encoding params.

`profile` (*profile*)

Set profile restrictions.

`fastfirstpass`

Enable fast settings when encoding first pass, when set to 1. When set to 0, it has the same effect of x264's `--slow-firstpass` option.

`crf` (*crf*)

Set the quality for constant quality mode.



`crf_max` (*crf-max*)

In CRF mode, prevents VBV from lowering quality beyond this point.

`qp` (*qp*)

Set constant quantization rate control method parameter.

`aq-mode` (*aq-mode*)

Set AQ method. Possible values:

‘none (0)’

Disabled.

‘variance (1)’

Variance AQ (complexity mask).

‘autovariance (2)’

Auto-variance AQ (experimental).

`aq-strength` (*aq-strength*)

Set AQ strength, reduce blocking and blurring in flat and textured areas.

`psy`

Use psychovisual optimizations when set to 1. When set to 0, it has the same effect as x264’s `--no-psy` option.

`psy-rd` (*psy-rd*)

Set strength of psychovisual optimization, in *psy-rd:psy-trellis* format.

`rc-lookahead` (*rc-lookahead*)

Set number of frames to look ahead for frametype and ratecontrol.

`weightb`

Enable weighted prediction for B-frames when set to 1. When set to 0, it has the same effect as x264’s `--no-weightb` option.

`weightp` (*weightp*)

Set weighted prediction method for P-frames. Possible values:

`'none (0)'`

Disabled

`'simple (1)'`

Enable only weighted refs

`'smart (2)'`

Enable both weighted refs and duplicates

`ssim (ssim)`

Enable calculation and printing SSIM stats after the encoding.

`intra-refresh (intra-refresh)`

Enable the use of Periodic Intra Refresh instead of IDR frames when set to 1.

`avcintra-class (class)`

Configure the encoder to generate AVC-Intra. Valid values are 50,100 and 200

`bluray-compat (bluray-compat)`

Configure the encoder to be compatible with the bluray standard. It is a shorthand for setting "bluray-compat=1 force-cfr=1".

`b-bias (b-bias)`

Set the influence on how often B-frames are used.

`b-pyramid (b-pyramid)`

Set method for keeping of some B-frames as references. Possible values:

`'none (none)'`

Disabled.

`'strict (strict)'`

Strictly hierarchical pyramid.

`'normal (normal)'`

Non-strict (not Blu-ray compatible).

`mixed-refs`

Enable the use of one reference per partition, as opposed to one reference per macroblock when set to 1. When set to 0, it has the same effect as x264's `--no-mixed-refs` option.

`8x8dct`

Enable adaptive spatial transform (high profile 8x8 transform) when set to 1. When set to 0, it has the same effect as x264's `--no-8x8dct` option.

`fast-pskip`

Enable early SKIP detection on P-frames when set to 1. When set to 0, it has the same effect as x264's `--no-fast-pskip` option.

`aud` (*aud*)

Enable use of access unit delimiters when set to 1.

`mbtree`

Enable use macroblock tree ratecontrol when set to 1. When set to 0, it has the same effect as x264's `--no-mbtree` option.

`deblock` (*deblock*)

Set loop filter parameters, in *alpha:beta* form.

`cplxblur` (*cplxblur*)

Set fluctuations reduction in QP (before curve compression).

`partitions` (*partitions*)

Set partitions to consider as a comma-separated list of. Possible values in the list:

`'p8x8'`

8x8 P-frame partition.

`'p4x4'`

4x4 P-frame partition.

`'b8x8'`

4x4 B-frame partition.

`'i8x8'`

8x8 I-frame partition.

`'i4x4'`

4x4 I-frame partition. (Enabling `'p4x4'` requires `'p8x8'` to be enabled. Enabling `'i8x8'` requires adaptive spatial transform (`8x8dct` option) to be enabled.)

`'none (none)'`

Do not consider any partitions.

`'all (all)'`

Consider every partition.

`direct-pred (direct)`

Set direct MV prediction mode. Possible values:

`'none (none)'`

Disable MV prediction.

`'spatial (spatial)'`

Enable spatial predicting.

`'temporal (temporal)'`

Enable temporal predicting.

`'auto (auto)'`

Automatically decided.

`slice-max-size (slice-max-size)`

Set the limit of the size of each slice in bytes. If not specified but RTP payload size (`ps`) is specified, that is used.

`stats (stats)`

Set the file name for multi-pass stats.

`nal-hrd` (*nal-hrd*)

Set signal HRD information (requires `vbv-bufsize` to be set). Possible values:

`'none (none)'`

Disable HRD information signaling.

`'vbr (vbr)'`

Variable bit rate.

`'cbr (cbr)'`

Constant bit rate (not allowed in MP4 container).

`x264opts` (*N.A.*)

Set any x264 option, see `x264 --fullhelp` for a list.

Argument is a list of *key=value* couples separated by ":". In *filter* and *psy-rd* options that use ":" as a separator themselves, use "," instead. They accept it as well since long ago but this is kept undocumented for some reason.

For example to specify libx264 encoding options with `ffmpeg`:

```
ffmpeg -i foo.mpg -vcodec libx264 -x264opts keyint=123:min-keyint=20 -an out.mkv
```

`x264-params` (*N.A.*)

Override the x264 configuration using a `:`-separated list of *key=value* parameters.

This option is functionally the same as the `x264opts`, but is duplicated for compatibility with the Libav fork.

For example to specify libx264 encoding options with `ffmpeg`:

```
ffmpeg -i INPUT -c:v libx264 -x264-params level=30:bframes=0:weightp=0:\
cabac=0:ref=1:vbv-maxrate=768:vbv-bufsize=2000:analyse=all:me=umh:\
no-fast-pskip=1:subq=6:8x8dct=0:trellis=0 OUTPUT
```

Encoding `ffpresets` for common usages are provided so they can be used with the general presets system (e.g. passing the `pre` option).

## 17.7 libx265# TOC

x265 H.265/HEVC encoder wrapper.

This encoder requires the presence of the libx265 headers and library during configuration. You need to explicitly configure the build with `--enable-libx265`.

### 17.7.1 Options# TOC

preset

Set the x265 preset.

tune

Set the x265 tune parameter.

x265-params

Set x265 options using a list of *key=value* couples separated by ":". See `x265 --help` for a list of options.

For example to specify libx265 encoding options with `-x265-params`:

```
ffmpeg -i input -c:v libx265 -x265-params crf=26:psy-rd=1 output.mp4
```

### 17.8 libxvid# TOC

Xvid MPEG-4 Part 2 encoder wrapper.

This encoder requires the presence of the libxvidcore headers and library during configuration. You need to explicitly configure the build with `--enable-libxvid --enable-gpl`.

The native `mpeg4` encoder supports the MPEG-4 Part 2 format, so users can encode to this format without this library.

### 17.8.1 Options# TOC

The following options are supported by the libxvid wrapper. Some of the following options are listed but are not documented, and correspond to shared codec options. See the Codec Options chapter for their documentation. The other shared options which are not listed have no effect for the libxvid encoder.

b

g

qmin

qmax

mpeg\_quant

threads

bf

b\_qfactor

b\_qoffset

flags

Set specific encoding flags. Possible values:

‘mv4’

Use four motion vector by macroblock.

‘aic’

Enable high quality AC prediction.

‘gray’

Only encode grayscale.

‘gmc’

Enable the use of global motion compensation (GMC).

‘qpel’

Enable quarter-pixel motion compensation.

‘cgop’

Enable closed GOP.

‘global\_header’

Place global headers in extradata instead of every keyframe.

trellis

me\_method

Set motion estimation method. Possible values in decreasing order of speed and increasing order of quality:

‘zero’

Use no motion estimation (default).

‘phods’

‘x1’

‘log’

Enable advanced diamond zonal search for 16x16 blocks and half-pixel refinement for 16x16 blocks. ‘x1’ and ‘log’ are aliases for ‘phods’.

`'epzs'`

Enable all of the things described above, plus advanced diamond zonal search for 8x8 blocks, half-pixel refinement for 8x8 blocks, and motion estimation on chroma planes.

`'full'`

Enable all of the things described above, plus extended 16x16 and 8x8 blocks search.

`mbd`

Set macroblock decision algorithm. Possible values in the increasing order of quality:

`'simple'`

Use macroblock comparing function algorithm (default).

`'bits'`

Enable rate distortion-based half pixel and quarter pixel refinement for 16x16 blocks.

`'rd'`

Enable all of the things described above, plus rate distortion-based half pixel and quarter pixel refinement for 8x8 blocks, and rate distortion-based search using square pattern.

`lumi_aq`

Enable lumi masking adaptive quantization when set to 1. Default is 0 (disabled).

`variance_aq`

Enable variance adaptive quantization when set to 1. Default is 0 (disabled).

When combined with `lumi_aq`, the resulting quality will not be better than any of the two specified individually. In other words, the resulting quality will be the worse one of the two effects.

`ssim`

Set structural similarity (SSIM) displaying method. Possible values:

`'off'`

Disable displaying of SSIM information.

`'avg'`

Output average SSIM at the end of encoding to stdout. The format of showing the average SSIM is:



Average SSIM: %f

For users who are not familiar with C, %f means a float number, or a decimal (e.g. 0.939232).

‘frame’

Output both per-frame SSIM data during encoding and average SSIM at the end of encoding to stdout. The format of per-frame information is:

SSIM: avg: %1.3f min: %1.3f max: %1.3f

For users who are not familiar with C, %1.3f means a float number rounded to 3 digits after the dot (e.g. 0.932).

ssim\_acc

Set SSIM accuracy. Valid options are integers within the range of 0-4, while 0 gives the most accurate result and 4 computes the fastest.

## 17.9 mpeg2# TOC

MPEG-2 video encoder.

### 17.9.1 Options# TOC

seq\_disp\_ext *integer*

Specifies if the encoder should write a sequence\_display\_extension to the output.

-1

auto

Decide automatically to write it or not (this is the default) by checking if the data to be written is different from the default or unspecified values.

0

never

Never write it.

1

always

Always write it.

## 17.10 png# TOC

PNG image encoder.

### 17.10.1 Private options# TOC

`dpi integer`

Set physical density of pixels, in dots per inch, unset by default

`dpm integer`

Set physical density of pixels, in dots per meter, unset by default

## 17.11 ProRes# TOC

Apple ProRes encoder.

FFmpeg contains 2 ProRes encoders, the `prores-aw` and `prores-ks` encoder. The used encoder can be chosen with the `-vcodec` option.

### 17.11.1 Private Options for prores-ks# TOC

`profile integer`

Select the ProRes profile to encode

`'proxy'`  
`'lt'`  
`'standard'`  
`'hq'`  
`'4444'`

`quant_mat integer`

Select quantization matrix.

`'auto'`  
`'default'`  
`'proxy'`  
`'lt'`  
`'standard'`  
`'hq'`

If set to *auto*, the matrix matching the profile will be picked. If not set, the matrix providing the highest quality, *default*, will be picked.

`bits_per_mb integer`

How many bits to allot for coding one macroblock. Different profiles use between 200 and 2400 bits per macroblock, the maximum is 8000.

`mbs_per_slice integer`

Number of macroblocks in each slice (1-8); the default value (8) should be good in almost all situations.

`vendor string`

Override the 4-byte vendor ID. A custom vendor ID like *apl0* would claim the stream was produced by the Apple encoder.

`alpha_bits integer`

Specify number of bits for alpha component. Possible values are 0, 8 and 16. Use 0 to disable alpha plane coding.

### 17.11.2 Speed considerations# TOC

In the default mode of operation the encoder has to honor frame constraints (i.e. not produce frames with size bigger than requested) while still making output picture as good as possible. A frame containing a lot of small details is harder to compress and the encoder would spend more time searching for appropriate quantizers for each slice.

Setting a higher `bits_per_mb` limit will improve the speed.

For the fastest encoding speed set the `qscale` parameter (4 is the recommended value) and do not set a size constraint.

### 17.12 libkvazaar# TOC

Kvazaar H.265/HEVC encoder.

Requires the presence of the libkvazaar headers and library during configuration. You need to explicitly configure the build with `--enable-libkvazaar`.

#### 17.12.1 Options# TOC

`b`

Set target video bitrate in bit/s and enable rate control.

`threads`

Set number of encoding threads.

`kvazaar-params`

Set kvazaar parameters as a list of *name=value* pairs separated by commas (.). See kvazaar documentation for a list of options.

## 18 Subtitles Encoders# TOC

### 18.1 dvdsub# TOC

This codec encodes the bitmap subtitle format that is used in DVDs. Typically they are stored in VOBSUB file pairs (\*.idx + \*.sub), and they can also be used in Matroska files.

#### 18.1.1 Options# TOC

`even_rows_fix`

When set to 1, enable a work-around that makes the number of pixel rows even in all subtitles. This fixes a problem with some players that cut off the bottom row if the number is odd. The work-around just adds a fully transparent row if needed. The overhead is low, typically one byte per subtitle on average.

By default, this work-around is disabled.

## 19 Bitstream Filters# TOC

When you configure your FFmpeg build, all the supported bitstream filters are enabled by default. You can list all available ones using the configure option `--list-bsfs`.

You can disable all the bitstream filters using the configure option `--disable-bsfs`, and selectively enable any bitstream filter using the option `--enable-bsf=BSF`, or you can disable a particular bitstream filter using the option `--disable-bsf=BSF`.

The option `-bsfs` of the ff\* tools will display the list of all the supported bitstream filters included in your build.

The ff\* tools have a `-bsf` option applied per stream, taking a comma-separated list of filters, whose parameters follow the filter name after a '='.

```
ffmpeg -i INPUT -c:v copy -bsf:v filter1[=opt1=str1/opt2=str2][,filter2] OUTPUT
```

Below is a description of the currently available bitstream filters, with their parameters, if any.

## 19.1 aac\_adtstoasc# TOC

Convert MPEG-2/4 AAC ADTS to MPEG-4 Audio Specific Configuration bitstream filter.

This filter creates an MPEG-4 AudioSpecificConfig from an MPEG-2/4 ADTS header and removes the ADTS header.

This is required for example when copying an AAC stream from a raw ADTS AAC container to a FLV or a MOV/MP4 file.

## 19.2 chomp# TOC

Remove zero padding at the end of a packet.

## 19.3 dump\_extra# TOC

Add extradata to the beginning of the filtered packets.

The additional argument specifies which packets should be filtered. It accepts the values:

‘a’

add extradata to all key packets, but only if *local\_header* is set in the *flags2* codec context field

‘k’

add extradata to all key packets

‘e’

add extradata to all packets

If not specified it is assumed ‘k’.

For example the following `ffmpeg` command forces a global header (thus disabling individual packet headers) in the H.264 packets generated by the `libx264` encoder, but corrects them by adding the header stored in extradata to the key packets:

```
ffmpeg -i INPUT -map 0 -flags:v +global_header -c:v libx264 -bsf:v dump_extra out.ts
```

## 19.4 h264\_mp4toannexb# TOC

Convert an H.264 bitstream from length prefixed mode to start code prefixed mode (as defined in the Annex B of the ITU-T H.264 specification).

This is required by some streaming formats, typically the MPEG-2 transport stream format ("mpegts").

For example to remux an MP4 file containing an H.264 stream to mpegts format with `ffmpeg`, you can use the command:

```
ffmpeg -i INPUT.mp4 -codec copy -bsf:v h264_mp4toannexb OUTPUT.ts
```

## 19.5 imxdump# TOC

Modifies the bitstream to fit in MOV and to be usable by the Final Cut Pro decoder. This filter only applies to the `mpeg2video` codec, and is likely not needed for Final Cut Pro 7 and newer with the appropriate `-tag:v`.

For example, to remux 30 MB/sec NTSC IMX to MOV:

```
ffmpeg -i input.mxf -c copy -bsf:v imxdump -tag:v mx3n output.mov
```

## 19.6 mjpeg2jpeg# TOC

Convert MJPEG/AVI1 packets to full JPEG/JFIF packets.

MJPEG is a video codec wherein each video frame is essentially a JPEG image. The individual frames can be extracted without loss, e.g. by

```
ffmpeg -i ../some_mjpeg.avi -c:v copy frames_%d.jpg
```

Unfortunately, these chunks are incomplete JPEG images, because they lack the DHT segment required for decoding. Quoting from <http://www.digitalpreservation.gov/formats/fdd/fdd000063.shtml>:

Avery Lee, writing in the `rec.video.desktop` newsgroup in 2001, commented that "MJPEG, or at least the MJPEG in AVIs having the MJPG fourcc, is restricted JPEG with a fixed – and \*omitted\* – Huffman table. The JPEG must be YCbCr colorspace, it must be 4:2:2, and it must use basic Huffman encoding, not arithmetic or progressive. . . . You can indeed extract the MJPEG frames and decode them with a regular JPEG decoder, but you have to prepend the DHT segment to them, or else the decoder won't have any idea how to decompress the data. The exact table necessary is given in the OpenDML spec."

This bitstream filter patches the header of frames extracted from an MJPEG stream (carrying the AVI1 header ID and lacking a DHT segment) to produce fully qualified JPEG images.

```
ffmpeg -i mjpeg-movie.avi -c:v copy -bsf:v mjpeg2jpeg frame_%d.jpg
exiftran -i -9 frame*.jpg
ffmpeg -i frame_%d.jpg -c:v copy rotated.avi
```

## 19.7 mjpega\_dump\_header# TOC

## 19.8 movsub# TOC

## 19.9 mp3\_header\_decompress# TOC

## 19.10 mpeg4\_unpack\_bframes# TOC

Unpack DivX-style packed B-frames.

DivX-style packed B-frames are not valid MPEG-4 and were only a workaround for the broken Video for Windows subsystem. They use more space, can cause minor AV sync issues, require more CPU power to decode (unless the player has some decoded picture queue to compensate the 2,0,2,0 frame per packet style) and cause trouble if copied into a standard container like mp4 or mpeg-ps/ts, because MPEG-4 decoders may not be able to decode them, since they are not valid MPEG-4.

For example to fix an AVI file containing an MPEG-4 stream with DivX-style packed B-frames using `ffmpeg`, you can use the command:

```
ffmpeg -i INPUT.avi -codec copy -bsf:v mpeg4_unpack_bframes OUTPUT.avi
```

## 19.11 noise# TOC

Damages the contents of packets without damaging the container. Can be used for fuzzing or testing error resilience/concealment.

Parameters: A numeral string, whose value is related to how often output bytes will be modified. Therefore, values below or equal to 0 are forbidden, and the lower the more frequent bytes will be modified, with 1 meaning every byte is modified.

```
ffmpeg -i INPUT -c copy -bsf noise[=1] output.mkv
```

applies the modification to every byte.

## 19.12 remove\_extra# TOC

## 20 Format Options# TOC

The `libavformat` library provides some generic global options, which can be set on all the muxers and demuxers. In addition each muxer or demuxer may support so-called private options, which are specific for that component.

Options may be set by specifying *-option value* in the `FFmpeg` tools, or by setting the value explicitly in the `AVFormatContext` options or using the `libavutil/opt.h` API for programmatic use.

The list of supported options follows:

```
avioflags flags (input/output)
```

Possible values:

‘direct’

Reduce buffering.

`probesize integer (input)`

Set probing size in bytes, i.e. the size of the data to analyze to get stream information. A higher value will enable detecting more information in case it is dispersed into the stream, but will increase latency. Must be an integer not lesser than 32. It is 5000000 by default.

`packetsize integer (output)`

Set packet size.

`fflags flags (input/output)`

Set format flags.

Possible values:

‘ignidx’

Ignore index.

‘fastseek’

Enable fast, but inaccurate seeks for some formats.

‘genpts’

Generate PTS.

‘nofillin’

Do not fill in missing values that can be exactly calculated.

‘noparse’

Disable AVParsers, this needs +nofillin too.

‘igndts’

Ignore DTS.

‘discardcorrupt’



Discard corrupted frames.

`'sortdts'`

Try to interleave output packets by DTS.

`'keepside'`

Do not merge side data.

`'latm'`

Enable RTP MP4A-LATM payload.

`'nobuffer'`

Reduce the latency introduced by optional buffering

`'bitexact'`

Only write platform-, build- and time-independent data. This ensures that file and data checksums are reproducible and match between platforms. Its primary use is for regression testing.

`seek2any integer (input)`

Allow seeking to non-keyframes on demuxer level when supported if set to 1. Default is 0.

`analyzeduration integer (input)`

Specify how many microseconds are analyzed to probe the input. A higher value will enable detecting more accurate information, but will increase latency. It defaults to 5,000,000 microseconds = 5 seconds.

`cryptokey hexadecimal string (input)`

Set decryption key.

`indexmem integer (input)`

Set max memory used for timestamp index (per stream).

`rtbufsize integer (input)`

Set max memory used for buffering real-time frames.

`fdebug flags (input/output)`

Print specific debug info.

Possible values:

‘ts’

`max_delay integer (input/output)`

Set maximum muxing or demuxing delay in microseconds.

`fpsprobesize integer (input)`

Set number of frames used to probe fps.

`audio_preload integer (output)`

Set microseconds by which audio packets should be interleaved earlier.

`chunk_duration integer (output)`

Set microseconds for each chunk.

`chunk_size integer (output)`

Set size in bytes for each chunk.

`err_detect, f_err_detect flags (input)`

Set error detection flags. `f_err_detect` is deprecated and should be used only via the `ffmpeg` tool.

Possible values:

‘crccheck’

Verify embedded CRCs.

‘bitstream’

Detect bitstream specification deviations.

‘buffer’

Detect improper bitstream length.

‘explode’

Abort decoding on minor error detection.

‘careful’

Consider things that violate the spec and have not been seen in the wild as errors.

‘compliant’

Consider all spec non compliances as errors.

‘aggressive’

Consider things that a sane encoder should not do as an error.

`max_interleave_delta integer (output)`

Set maximum buffering duration for interleaving. The duration is expressed in microseconds, and defaults to 1000000 (1 second).

To ensure all the streams are interleaved correctly, libavformat will wait until it has at least one packet for each stream before actually writing any packets to the output file. When some streams are "sparse" (i.e. there are large gaps between successive packets), this can result in excessive buffering.

This field specifies the maximum difference between the timestamps of the first and the last packet in the muxing queue, above which libavformat will output a packet regardless of whether it has queued a packet for all the streams.

If set to 0, libavformat will continue buffering packets until it has a packet for each stream, regardless of the maximum timestamp difference between the buffered packets.

`use_wallclock_as_timestamps integer (input)`

Use wallclock as timestamps.

`avoid_negative_ts integer (output)`

Possible values:

‘make\_non\_negative’

Shift timestamps to make them non-negative. Also note that this affects only leading negative timestamps, and not non-monotonic negative timestamps.

‘make\_zero’

Shift timestamps so that the first timestamp is 0.

‘auto (default)’

Enables shifting when required by the target format.

‘disabled’

Disables shifting of timestamp.

When shifting is enabled, all output timestamps are shifted by the same amount. Audio, video, and subtitles desyncing and relative timestamp differences are preserved compared to how they would have been without shifting.

```
skip_initial_bytes integer (input)
```

Set number of bytes to skip before reading header and frames if set to 1. Default is 0.

```
correct_ts_overflow integer (input)
```

Correct single timestamp overflows if set to 1. Default is 1.

flush\_packets *integer* (output)

Flush the underlying I/O stream after each packet. Default 1 enables it, and has the effect of reducing the latency; 0 disables it and may slightly increase performance in some cases.

```
output_ts_offset offset (output)
```

Set the output time offset.

*offset* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

The offset is added by the muxer to the output timestamps.

Specifying a positive offset means that the corresponding streams are delayed by the time duration specified in *offset*. Default value is 0 (meaning that no offset is applied).

```
format_whitelist list (input)
```

"," separated List of allowed demuxers. By default all are allowed.

```
dump_separator string (input)
```

Separator used to separate the fields printed on the command line about the Stream parameters. For example to separate the fields with newlines and indentation:

```
ffprobe -dump_separator " " -i ~/videos/matrixbench mpeg2.mpg
```

## 20.1 Format stream specifiers# TOC

Format stream specifiers allow selection of one or more streams that match specific properties.

Possible forms of stream specifiers are:

*stream\_index*

Matches the stream with this index.

*stream\_type*[ :*stream\_index*]

*stream\_type* is one of following: 'v' for video, 'a' for audio, 's' for subtitle, 'd' for data, and 't' for attachments. If *stream\_index* is given, then it matches the stream number *stream\_index* of this type. Otherwise, it matches all streams of this type.

*p*:*program\_id*[ :*stream\_index*]

If *stream\_index* is given, then it matches the stream with number *stream\_index* in the program with the id *program\_id*. Otherwise, it matches all streams in the program.

#*stream\_id*

Matches the stream by a format-specific ID.

The exact semantics of stream specifiers is defined by the `avformat_match_stream_specifier()` function declared in the `libavformat/avformat.h` header.

## 21 Demuxers# TOC

Demuxers are configured elements in FFmpeg that can read the multimedia streams from a particular type of file.

When you configure your FFmpeg build, all the supported demuxers are enabled by default. You can list all available ones using the configure option `--list-demuxers`.

You can disable all the demuxers using the configure option `--disable-demuxers`, and selectively enable a single demuxer with the option `--enable-demuxer=DEMUXER`, or disable it with the option `--disable-demuxer=DEMUXER`.

The option `-formats` of the `ff*` tools will display the list of enabled demuxers.

The description of some of the currently available demuxers follows.

## 21.1 aa# TOC

Audible Format 2, 3, and 4 demuxer.

This demuxer is used to demux Audible Format 2, 3, and 4 (.aa) files.

## 21.2 applehttp# TOC

Apple HTTP Live Streaming demuxer.

This demuxer presents all AVStreams from all variant streams. The id field is set to the bitrate variant index number. By setting the discard flags on AVStreams (by pressing 'a' or 'v' in ffplay), the caller can decide which variant streams to actually receive. The total bitrate of the variant that the stream belongs to is available in a metadata key named "variant\_bitrate".

## 21.3 apng# TOC

Animated Portable Network Graphics demuxer.

This demuxer is used to demux APNG files. All headers, but the PNG signature, up to (but not including) the first fcTL chunk are transmitted as extradata. Frames are then split as being all the chunks between two fcTL ones, or between the last fcTL and IEND chunks.

`-ignore_loop bool`

Ignore the loop variable in the file if set.

`-max_fps int`

Maximum framerate in frames per second (0 for no limit).

`-default_fps int`

Default framerate in frames per second when none is specified in the file (0 meaning as fast as possible).

## 21.4 asf# TOC

Advanced Systems Format demuxer.

This demuxer is used to demux ASF files and MMS network streams.

`-no_resync_search bool`

Do not try to resynchronize by looking for a certain optional start code.

## 21.5 concat# TOC

Virtual concatenation script demuxer.

This demuxer reads a list of files and other directives from a text file and demuxes them one after the other, as if all their packet had been muxed together.

The timestamps in the files are adjusted so that the first file starts at 0 and each next file starts where the previous one finishes. Note that it is done globally and may cause gaps if all streams do not have exactly the same length.

All files must have the same streams (same codecs, same time base, etc.).

The duration of each file is used to adjust the timestamps of the next file: if the duration is incorrect (because it was computed using the bit-rate or because the file is truncated, for example), it can cause artifacts. The `duration` directive can be used to override the duration stored in each file.

### 21.5.1 Syntax# TOC

The script is a text file in extended-ASCII, with one directive per line. Empty lines, leading spaces and lines starting with '#' are ignored. The following directive is recognized:

`file path`

Path to a file to read; special characters and spaces must be escaped with backslash or single quotes.

All subsequent file-related directives apply to that file.

`ffconcat version 1.0`

Identify the script type and version. It also sets the `safe` option to 1 if it was to its default -1.

To make FFmpeg recognize the format automatically, this directive must appears exactly as is (no extra space or byte-order-mark) on the very first line of the script.

`duration dur`

Duration of the file. This information can be specified from the file; specifying it here may be more efficient or help if the information from the file is not available or accurate.

If the duration is set for all files, then it is possible to seek in the whole concatenated video.

`inpoint timestamp`

In point of the file. When the demuxer opens the file it instantly seeks to the specified timestamp. Seeking is done so that all streams can be presented successfully at In point.

This directive works best with intra frame codecs, because for non-intra frame ones you will usually get extra packets before the actual In point and the decoded content will most likely contain frames before In point too.

For each file, packets before the file In point will have timestamps less than the calculated start timestamp of the file (negative in case of the first file), and the duration of the files (if not specified by the `duration` directive) will be reduced based on their specified In point.

Because of potential packets before the specified In point, packet timestamps may overlap between two concatenated files.

`outpoint timestamp`

Out point of the file. When the demuxer reaches the specified decoding timestamp in any of the streams, it handles it as an end of file condition and skips the current and all the remaining packets from all streams.

Out point is exclusive, which means that the demuxer will not output packets with a decoding timestamp greater or equal to Out point.

This directive works best with intra frame codecs and formats where all streams are tightly interleaved. For non-intra frame codecs you will usually get additional packets with presentation timestamp after Out point therefore the decoded content will most likely contain frames after Out point too. If your streams are not tightly interleaved you may not get all the packets from all streams before Out point and you may only will be able to decode the earliest stream until Out point.

The duration of the files (if not specified by the `duration` directive) will be reduced based on their specified Out point.

`file_packet_metadata key=value`

Metadata of the packets of the file. The specified metadata will be set for each file packet. You can specify this directive multiple times to add multiple metadata entries.

`stream`

Introduce a stream in the virtual file. All subsequent stream-related directives apply to the last introduced stream. Some streams properties must be set in order to allow identifying the matching streams in the subfiles. If no streams are defined in the script, the streams from the first file are copied.

`exact_stream_id id`

Set the id of the stream. If this directive is given, the string with the corresponding id in the subfiles will be used. This is especially useful for MPEG-PS (VOB) files, where the order of the streams is not reliable.



## 21.5.2 Options# TOC

This demuxer accepts the following option:

`safe`

If set to 1, reject unsafe file paths. A file path is considered safe if it does not contain a protocol specification and is relative and all components only contain characters from the portable character set (letters, digits, period, underscore and hyphen) and have no period at the beginning of a component.

If set to 0, any file name is accepted.

The default is -1, it is equivalent to 1 if the format was automatically probed and 0 otherwise.

`auto_convert`

If set to 1, try to perform automatic conversions on packet data to make the streams concatenable. The default is 1.

Currently, the only conversion is adding the `h264_mp4toannexb` bitstream filter to H.264 streams in MP4 format. This is necessary in particular if there are resolution changes.

## 21.6 flv# TOC

Adobe Flash Video Format demuxer.

This demuxer is used to demux FLV files and RTMP network streams.

`-flv_metadata bool`

Allocate the streams according to the `onMetaData` array content.

## 21.7 libgme# TOC

The Game Music Emu library is a collection of video game music file emulators.

See <http://code.google.com/p/game-music-emu/> for more information.

Some files have multiple tracks. The demuxer will pick the first track by default. The `track_index` option can be used to select a different track. Track indexes start at 0. The demuxer exports the number of tracks as *tracks* meta data entry.

For very large files, the `max_size` option may have to be adjusted.

## 21.8 libquvi# TOC

Play media from Internet services using the quvi project.

The demuxer accepts a `format` option to request a specific quality. It is by default set to *best*.

See <http://quvi.sourceforge.net/> for more information.

FFmpeg needs to be built with `--enable-libquvi` for this demuxer to be enabled.

## 21.9 gif# TOC

Animated GIF demuxer.

It accepts the following options:

`min_delay`

Set the minimum valid delay between frames in hundredths of seconds. Range is 0 to 6000. Default value is 2.

`max_gif_delay`

Set the maximum valid delay between frames in hundredth of seconds. Range is 0 to 65535. Default value is 65535 (nearly eleven minutes), the maximum value allowed by the specification.

`default_delay`

Set the default delay between frames in hundredths of seconds. Range is 0 to 6000. Default value is 10.

`ignore_loop`

GIF files can contain information to loop a certain number of times (or infinitely). If `ignore_loop` is set to 1, then the loop setting from the input will be ignored and looping will not occur. If set to 0, then looping will occur and will cycle the number of times according to the GIF. Default value is 1.

For example, with the overlay filter, place an infinitely looping GIF over another video:

```
ffmpeg -i input.mp4 -ignore_loop 0 -i input.gif -filter_complex overlay=shortest=1 out.mkv
```

Note that in the above example the `shortest` option for overlay filter is used to end the output video at the length of the shortest input file, which in this case is `input.mp4` as the GIF in this example loops infinitely.

## 21.10 image2# TOC

Image file demuxer.

This demuxer reads from a list of image files specified by a pattern. The syntax and meaning of the pattern is specified by the option *pattern\_type*.

The pattern may contain a suffix which is used to automatically determine the format of the images contained in the files.

The size, the pixel format, and the format of each image must be the same for all the files in the sequence.

This demuxer accepts the following options:

`framerate`

Set the frame rate for the video stream. It defaults to 25.

`loop`

If set to 1, loop over the input. Default value is 0.

`pattern_type`

Select the pattern type used to interpret the provided filename.

*pattern\_type* accepts one of the following values.

`none`

Disable pattern matching, therefore the video will only contain the specified image. You should use this option if you do not want to create sequences from multiple images and your filenames may contain special pattern characters.

`sequence`

Select a sequence pattern type, used to specify a sequence of files indexed by sequential numbers.

A sequence pattern may contain the string "%d" or "%0Nd", which specifies the position of the characters representing a sequential number in each filename matched by the pattern. If the form "%d0Nd" is used, the string representing the number in each filename is 0-padded and *N* is the total number of 0-padded digits representing the number. The literal character '%' can be specified in the pattern with the string "%%".

If the sequence pattern contains "%d" or "%0Nd", the first filename of the file list specified by the pattern must contain a number inclusively contained between *start\_number* and *start\_number+start\_number\_range-1*, and all the following numbers must be sequential.

For example the pattern "img-%03d.bmp" will match a sequence of filenames of the form `img-001.bmp`, `img-002.bmp`, ..., `img-010.bmp`, etc.; the pattern "i%%m%%g-%d.jpg" will match a sequence of filenames of the form `i%m%g-1.jpg`, `i%m%g-2.jpg`, ..., `i%m%g-10.jpg`, etc.

Note that the pattern must not necessarily contain "%d" or "%Nd", for example to convert a single image file `img.jpeg` you can employ the command:

```
ffmpeg -i img.jpeg img.png
```

`glob`

Select a glob wildcard pattern type.

The pattern is interpreted like a `glob()` pattern. This is only selectable if libavformat was compiled with globbing support.

`glob_sequence` (*deprecated, will be removed*)

Select a mixed glob wildcard/sequence pattern.

If your version of libavformat was compiled with globbing support, and the provided pattern contains at least one glob meta character among `%*?[]{}` that is preceded by an unescaped "%", the pattern is interpreted like a `glob()` pattern, otherwise it is interpreted like a sequence pattern.

All glob special characters `%*?[]{}` must be prefixed with "%". To escape a literal "%" you shall use "%%".

For example the pattern `foo-%*.jpeg` will match all the filenames prefixed by "foo-" and terminating with ".jpeg", and `foo-%?%?%.jpeg` will match all the filenames prefixed with "foo-", followed by a sequence of three characters, and terminating with ".jpeg".

This pattern type is deprecated in favor of *glob* and *sequence*.

Default value is *glob\_sequence*.

`pixel_format`

Set the pixel format of the images to read. If not specified the pixel format is guessed from the first image file in the sequence.

`start_number`

Set the index of the file matched by the image file pattern to start to read from. Default value is 0.

`start_number_range`

Set the index interval range to check when looking for the first image file in the sequence, starting from *start\_number*. Default value is 5.

`ts_from_file`

If set to 1, will set frame timestamp to modification time of image file. Note that monotonicity of timestamps is not provided: images go in the same order as without this option. Default value is 0. If set to 2, will set frame timestamp to the modification time of the image file in nanosecond precision.

`video_size`

Set the video size of the images to read. If not specified the video size is guessed from the first image file in the sequence.

### 21.10.1 Examples# TOC

- Use `ffmpeg` for creating a video from the images in the file sequence `img-001.jpeg`, `img-002.jpeg`, ..., assuming an input frame rate of 10 frames per second:

```
ffmpeg -framerate 10 -i 'img-%03d.jpeg' out.mkv
```

- As above, but start by reading from a file with index 100 in the sequence:

```
ffmpeg -framerate 10 -start_number 100 -i 'img-%03d.jpeg' out.mkv
```

- Read images matching the `"*.png"` glob pattern, that is all the files terminating with the `".png"` suffix:

```
ffmpeg -framerate 10 -pattern_type glob -i "*.png" out.mkv
```

### 21.11 mpegts# TOC

MPEG-2 transport stream demuxer.

This demuxer accepts the following options:

`resync_size`

Set size limit for looking up a new synchronization. Default value is 65536.

`fix_teletext_pts`

Override teletext packet PTS and DTS values with the timestamps calculated from the PCR of the first program which the teletext stream is part of and is not discarded. Default value is 1, set this option to 0 if you want your teletext packet PTS and DTS values untouched.

`ts_packet_size`

Output option carrying the raw packet size in bytes. Show the detected raw packet size, cannot be set by the user.

`scan_all_pmts`

Scan and combine all PMTs. The value is an integer with value from -1 to 1 (-1 means automatic setting, 1 means enabled, 0 means disabled). Default value is -1.

## 21.12 rawvideo# TOC

Raw video demuxer.

This demuxer allows one to read raw video data. Since there is no header specifying the assumed video parameters, the user must specify them in order to be able to decode the data correctly.

This demuxer accepts the following options:

`framerate`

Set input video frame rate. Default value is 25.

`pixel_format`

Set the input video pixel format. Default value is yuv420p.

`video_size`

Set the input video size. This value must be specified explicitly.

For example to read a rawvideo file `input.raw` with `ffplay`, assuming a pixel format of `rgb24`, a video size of `320x240`, and a frame rate of 10 images per second, use the command:

```
ffplay -f rawvideo -pixel_format rgb24 -video_size 320x240 -framerate 10 input.raw
```

## 21.13 sbg# TOC

SBaGen script demuxer.

This demuxer reads the script language used by SBaGen <http://uazu.net/sbagen/> to generate binaural beats sessions. A SBG script looks like that:

```
-SE
a: 300-2.5/3 440+4.5/0
b: 300-2.5/0 440+4.5/3
off: -
NOW      == a
+0:07:00 == b
+0:14:00 == a
+0:21:00 == b
+0:30:00  off
```

A SBG script can mix absolute and relative timestamps. If the script uses either only absolute timestamps (including the script start time) or only relative ones, then its layout is fixed, and the conversion is straightforward. On the other hand, if the script mixes both kind of timestamps, then the *NOW* reference for relative timestamps will be taken from the current time of day at the time the script is read, and the script layout will be frozen according to that reference. That means that if the script is directly played, the actual times will match the absolute timestamps up to the sound controller's clock accuracy, but if the user somehow pauses the playback or seeks, all times will be shifted accordingly.

## 21.14 tedcaptions# TOC

JSON captions used for TED Talks.

TED does not provide links to the captions, but they can be guessed from the page. The file `tools/bookmarklets.html` from the FFmpeg source tree contains a bookmarklet to expose them.

This demuxer accepts the following option:

`start_time`

Set the start time of the TED talk, in milliseconds. The default is 15000 (15s). It is used to sync the captions with the downloadable videos, because they include a 15s intro.

Example: convert the captions to a format most players understand:

```
ffmpeg -i http://www.ted.com/talks/subtitles/id/1/lang/en talk1-en.srt
```

## 22 Muxers# TOC

Muxers are configured elements in FFmpeg which allow writing multimedia streams to a particular type of file.

When you configure your FFmpeg build, all the supported muxers are enabled by default. You can list all available muxers using the configure option `--list-muxers`.

You can disable all the muxers with the configure option `--disable-muxers` and selectively enable / disable single muxers with the options `--enable-muxer=MUXER` / `--disable-muxer=MUXER`.

The option `-formats` of the ff\* tools will display the list of enabled muxers.

A description of some of the currently available muxers follows.

### 22.1 aiff# TOC

Audio Interchange File Format muxer.

### 22.1.1 Options# TOC

It accepts the following options:

`write_id3v2`

Enable ID3v2 tags writing when set to 1. Default is 0 (disabled).

`id3v2_version`

Select ID3v2 version to write. Currently only version 3 and 4 (aka. ID3v2.3 and ID3v2.4) are supported. The default is version 4.

### 22.2 crc# TOC

CRC (Cyclic Redundancy Check) testing format.

This muxer computes and prints the Adler-32 CRC of all the input audio and video frames. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the CRC.

The output of the muxer consists of a single line of the form: `CRC=0xCRC`, where *CRC* is a hexadecimal number 0-padded to 8 digits containing the CRC for all the decoded input frames.

See also the `framecrc` muxer.

#### 22.2.1 Examples# TOC

For example to compute the CRC of the input, and store it in the file `out.crc`:

```
ffmpeg -i INPUT -f crc out.crc
```

You can print the CRC to stdout with the command:

```
ffmpeg -i INPUT -f crc -
```

You can select the output format of each frame with `ffmpeg` by specifying the audio and video codec and format. For example to compute the CRC of the input audio converted to PCM unsigned 8-bit and the input video converted to MPEG-2 video, use the command:

```
ffmpeg -i INPUT -c:a pcm_u8 -c:v mpeg2video -f crc -
```

### 22.3 framecrc# TOC

Per-packet CRC (Cyclic Redundancy Check) testing format.

This muxer computes and prints the Adler-32 CRC for each audio and video packet. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the CRC.



The output of the muxer consists of a line for each audio and video packet of the form:

*stream\_index, packet\_dts, packet\_pts, packet\_duration, packet\_size, 0xCRC*

*CRC* is a hexadecimal number 0-padded to 8 digits containing the CRC of the packet.

### 22.3.1 Examples# TOC

For example to compute the CRC of the audio and video frames in `INPUT`, converted to raw audio and video packets, and store it in the file `out.crc`:

```
ffmpeg -i INPUT -f framecrc out.crc
```

To print the information to stdout, use the command:

```
ffmpeg -i INPUT -f framecrc -
```

With `ffmpeg`, you can select the output format to which the audio and video frames are encoded before computing the CRC for each packet by specifying the audio and video codec. For example, to compute the CRC of each decoded input audio frame converted to PCM unsigned 8-bit and of each decoded input video frame converted to MPEG-2 video, use the command:

```
ffmpeg -i INPUT -c:a pcm_u8 -c:v mpeg2video -f framecrc -
```

See also the `crc` muxer.

## 22.4 framemd5# TOC

Per-packet MD5 testing format.

This muxer computes and prints the MD5 hash for each audio and video packet. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the hash.

The output of the muxer consists of a line for each audio and video packet of the form:

*stream\_index, packet\_dts, packet\_pts, packet\_duration, packet\_size, MD5*

*MD5* is a hexadecimal number representing the computed MD5 hash for the packet.

### 22.4.1 Examples# TOC

For example to compute the MD5 of the audio and video frames in `INPUT`, converted to raw audio and video packets, and store it in the file `out.md5`:

```
ffmpeg -i INPUT -f framemd5 out.md5
```

To print the information to stdout, use the command:

```
ffmpeg -i INPUT -f framemd5 -
```

See also the md5 muxer.

## 22.5 gif# TOC

Animated GIF muxer.

It accepts the following options:

`loop`

Set the number of times to loop the output. Use `-1` for no loop, 0 for looping indefinitely (default).

`final_delay`

Force the delay (expressed in centiseconds) after the last frame. Each frame ends with a delay until the next frame. The default is `-1`, which is a special value to tell the muxer to re-use the previous delay. In case of a loop, you might want to customize this value to mark a pause for instance.

For example, to encode a gif looping 10 times, with a 5 seconds delay between the loops:

```
ffmpeg -i INPUT -loop 10 -final_delay 500 out.gif
```

Note 1: if you wish to extract the frames in separate GIF files, you need to force the image2 muxer:

```
ffmpeg -i INPUT -c:v gif -f image2 "out%d.gif"
```

Note 2: the GIF format has a very small time base: the delay between two frames can not be smaller than one centi second.

## 22.6 hls# TOC

Apple HTTP Live Streaming muxer that segments MPEG-TS according to the HTTP Live Streaming (HLS) specification.

It creates a playlist file, and one or more segment files. The output filename specifies the playlist filename.

By default, the muxer creates a file for each segment produced. These files have the same name as the playlist, followed by a sequential number and a `.ts` extension.

For example, to convert an input file with `ffmpeg`:

```
ffmpeg -i in.nut out.m3u8
```

This example will produce the playlist, `out.m3u8`, and segment files: `out0.ts`, `out1.ts`, `out2.ts`, etc.

See also the segment muxer, which provides a more generic and flexible implementation of a segmenter, and can be used to perform HLS segmentation.

## 22.6.1 Options# TOC

This muxer supports the following options:

`hls_time` *seconds*

Set the segment length in seconds. Default value is 2.

`hls_list_size` *size*

Set the maximum number of playlist entries. If set to 0 the list file will contain all the segments. Default value is 5.

`hls_ts_options` *options\_list*

Set output format options using a `:`-separated list of key=value parameters. Values containing special characters must be escaped.

`hls_wrap` *wrap*

Set the number after which the segment filename number (the number specified in each segment file) wraps. If set to 0 the number will be never wrapped. Default value is 0.

This option is useful to avoid to fill the disk with many segment files, and limits the maximum number of segment files written to disk to *wrap*.

`start_number` *number*

Start the playlist sequence number from *number*. Default value is 0.

`hls_allow_cache` *allowcache*

Explicitly set whether the client MAY (1) or MUST NOT (0) cache media segments.

`hls_base_url` *baseurl*

Append *baseurl* to every entry in the playlist. Useful to generate playlists with absolute paths.

Note that the playlist sequence number must be unique for each segment and it is not to be confused with the segment filename sequence number which can be cyclic, for example if the `wrap` option is specified.

`hls_segment_filename` *filename*

Set the segment filename. Unless `hls_flags` `single_file` is set *filename* is used as a string format with the segment number:

```
ffmpeg in.nut -hls_segment_filename 'file%03d.ts' out.m3u8
```

This example will produce the playlist, `out.m3u8`, and segment files: `file000.ts`, `file001.ts`, `file002.ts`, etc.

```
hls_key_info_file key_info_file
```

Use the information in *key\_info\_file* for segment encryption. The first line of *key\_info\_file* specifies the key URI written to the playlist. The key URL is used to access the encryption key during playback. The second line specifies the path to the key file used to obtain the key during the encryption process. The key file is read as a single packed array of 16 octets in binary format. The optional third line specifies the initialization vector (IV) as a hexadecimal string to be used instead of the segment sequence number (default) for encryption. Changes to *key\_info\_file* will result in segment encryption with the new key/IV and an entry in the playlist for the new key URI/IV.

Key info file format:

```
key URI  
key file path  
IV (optional)
```

Example key URIs:

```
http://server/file.key  
/path/to/file.key  
file.key
```

Example key file paths:

```
file.key  
/path/to/file.key
```

Example IV:

```
0123456789ABCDEF0123456789ABCDEF
```

Key info file example:

```
http://server/file.key  
/path/to/file.key  
0123456789ABCDEF0123456789ABCDEF
```

Example shell script:

```
#!/bin/sh
BASE_URL=${1:-'.'}
openssl rand 16 > file.key
echo $BASE_URL/file.key > file.keyinfo
echo file.key >> file.keyinfo
echo $(openssl rand -hex 16) >> file.keyinfo
ffmpeg -f lavfi -re -i testsrc -c:v h264 -hls_flags delete_segments \
    -hls_key_info_file file.keyinfo out.m3u8
```

`hls_flags single_file`

If this flag is set, the muxer will store all segments in a single MPEG-TS file, and will use byte ranges in the playlist. HLS playlists generated with this way will have the version number 4. For example:

```
ffmpeg -i in.nut -hls_flags single_file out.m3u8
```

Will produce the playlist, `out.m3u8`, and a single segment file, `out.ts`.

`hls_flags delete_segments`

Segment files removed from the playlist are deleted after a period of time equal to the duration of the segment plus the duration of the playlist.

## 22.7 ico# TOC

ICO file muxer.

Microsoft's icon file format (ICO) has some strict limitations that should be noted:

- Size cannot exceed 256 pixels in any dimension
- Only BMP and PNG images can be stored
- If a BMP image is used, it must be one of the following pixel formats:

BMP Bit Depth	FFmpeg Pixel Format
1bit	pal8
4bit	pal8
8bit	pal8
16bit	rgb555le
24bit	bgr24
32bit	bgra

- If a BMP image is used, it must use the BITMAPINFOHEADER DIB header
- If a PNG image is used, it must use the rgba pixel format

## 22.8 image2# TOC

Image file muxer.

The image file muxer writes video frames to image files.

The output filenames are specified by a pattern, which can be used to produce sequentially numbered series of files. The pattern may contain the string "%d" or "%0Nd", this string specifies the position of the characters representing a numbering in the filenames. If the form "%0Nd" is used, the string representing the number in each filename is 0-padded to *N* digits. The literal character '%' can be specified in the pattern with the string "%%".

If the pattern contains "%d" or "%0Nd", the first filename of the file list specified will contain the number 1, all the following numbers will be sequential.

The pattern may contain a suffix which is used to automatically determine the format of the image files to write.

For example the pattern "img-%03d.bmp" will specify a sequence of filenames of the form img-001.bmp, img-002.bmp, ..., img-010.bmp, etc. The pattern "img%%-%d.jpg" will specify a sequence of filenames of the form img%-1.jpg, img%-2.jpg, ..., img%-10.jpg, etc.

### 22.8.1 Examples# TOC

The following example shows how to use `ffmpeg` for creating a sequence of files `img-001.jpeg`, `img-002.jpeg`, ..., taking one image every second from the input video:

```
ffmpeg -i in.avi -vsync 1 -r 1 -f image2 'img-%03d.jpeg'
```

Note that with `ffmpeg`, if the format is not specified with the `-f` option and the output filename specifies an image file format, the `image2` muxer is automatically selected, so the previous command can be written as:

```
ffmpeg -i in.avi -vsync 1 -r 1 'img-%03d.jpeg'
```

Note also that the pattern must not necessarily contain "%d" or "%0Nd", for example to create a single image file `img.jpeg` from the input video you can employ the command:

```
ffmpeg -i in.avi -f image2 -frames:v 1 img.jpeg
```

The `strftime` option allows you to expand the filename with date and time information. Check the documentation of the `strftime()` function for the syntax.

For example to generate image files from the `strftime()` "%Y-%m-%d\_%H-%M-%S" pattern, the following `ffmpeg` command can be used:

```
ffmpeg -f v4l2 -r 1 -i /dev/video0 -f image2 -strftime 1 "%Y-%m-%d_%H-%M-%S.jpg"
```

### 22.8.2 Options# TOC

`start_number`

Start the sequence from the specified number. Default value is 0.

`update`

If set to 1, the filename will always be interpreted as just a filename, not a pattern, and the corresponding file will be continuously overwritten with new images. Default value is 0.

`strftime`

If set to 1, expand the filename with date and time information from `strftime()`. Default value is 0.

The image muxer supports the .Y.U.V image file format. This format is special in that each image frame consists of three files, for each of the YUV420P components. To read or write this image file format, specify the name of the '.Y' file. The muxer will automatically open the '.U' and '.V' files as required.

## 22.9 matroska# TOC

Matroska container muxer.

This muxer implements the matroska and webm container specs.

### 22.9.1 Metadata# TOC

The recognized metadata settings in this muxer are:

`title`

Set title name provided to a single track.

`language`

Specify the language of the track in the Matroska languages form.

The language can be either the 3 letters bibliographic ISO-639-2 (ISO 639-2/B) form (like "fre" for French), or a language code mixed with a country code for specialities in languages (like "fre-ca" for Canadian French).

`stereo_mode`

Set stereo 3D video layout of two views in a single video track.

The following values are recognized:

'mono'

video is not stereo

`'left_right'`

Both views are arranged side by side, Left-eye view is on the left

`'bottom_top'`

Both views are arranged in top-bottom orientation, Left-eye view is at bottom

`'top_bottom'`

Both views are arranged in top-bottom orientation, Left-eye view is on top

`'checkerboard_rl'`

Each view is arranged in a checkerboard interleaved pattern, Left-eye view being first

`'checkerboard_lr'`

Each view is arranged in a checkerboard interleaved pattern, Right-eye view being first

`'row_interleaved_rl'`

Each view is constituted by a row based interleaving, Right-eye view is first row

`'row_interleaved_lr'`

Each view is constituted by a row based interleaving, Left-eye view is first row

`'col_interleaved_rl'`

Both views are arranged in a column based interleaving manner, Right-eye view is first column

`'col_interleaved_lr'`

Both views are arranged in a column based interleaving manner, Left-eye view is first column

`'anaglyph_cyan_red'`

All frames are in anaglyph format viewable through red-cyan filters

`'right_left'`

Both views are arranged side by side, Right-eye view is on the left

`'anaglyph_green_magenta'`



All frames are in anaglyph format viewable through green-magenta filters

`'block_lr'`

Both eyes laced in one Block, Left-eye view is first

`'block_rl'`

Both eyes laced in one Block, Right-eye view is first

For example a 3D WebM clip can be created using the following command line:

```
ffmpeg -i sample_left_right_clip.mpg -an -c:v libvpx -metadata stereo_mode=left_right -y stereo_clip.webm
```

## 22.9.2 Options# TOC

This muxer supports the following options:

`reserve_index_space`

By default, this muxer writes the index for seeking (called cues in Matroska terms) at the end of the file, because it cannot know in advance how much space to leave for the index at the beginning of the file. However for some use cases – e.g. streaming where seeking is possible but slow – it is useful to put the index at the beginning of the file.

If this option is set to a non-zero value, the muxer will reserve a given amount of space in the file header and then try to write the cues there when the muxing finishes. If the available space does not suffice, muxing will fail. A safe size for most use cases should be about 50kB per hour of video.

Note that cues are only written if the output is seekable and this option will have no effect if it is not.

## 22.10 md5# TOC

MD5 testing format.

This muxer computes and prints the MD5 hash of all the input audio and video frames. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the hash.

The output of the muxer consists of a single line of the form: `MD5=MD5`, where *MD5* is a hexadecimal number representing the computed MD5 hash.

For example to compute the MD5 hash of the input converted to raw audio and video, and store it in the file `out.md5`:

```
ffmpeg -i INPUT -f md5 out.md5
```

You can print the MD5 to stdout with the command:

```
ffmpeg -i INPUT -f md5 -
```

See also the framemd5 muxer.

## 22.11 mov, mp4, ismv# TOC

MOV/MP4/ISMV (Smooth Streaming) muxer.

The mov/mp4/ismv muxer supports fragmentation. Normally, a MOV/MP4 file has all the metadata about all packets stored in one location (written at the end of the file, it can be moved to the start for better playback by adding *faststart* to the *movflags*, or using the *qt-faststart* tool). A fragmented file consists of a number of fragments, where packets and metadata about these packets are stored together. Writing a fragmented file has the advantage that the file is decodable even if the writing is interrupted (while a normal MOV/MP4 is undecodable if it is not properly finished), and it requires less memory when writing very long files (since writing normal MOV/MP4 files stores info about every single packet in memory until the file is closed). The downside is that it is less compatible with other applications.

### 22.11.1 Options# TOC

Fragmentation is enabled by setting one of the AVOptions that define how to cut the file into fragments:

`-moov_size bytes`

Reserves space for the moov atom at the beginning of the file instead of placing the moov atom at the end. If the space reserved is insufficient, muxing will fail.

`-movflags frag_keyframe`

Start a new fragment at each video keyframe.

`-frag_duration duration`

Create fragments that are *duration* microseconds long.

`-frag_size size`

Create fragments that contain up to *size* bytes of payload data.

`-movflags frag_custom`

Allow the caller to manually choose when to cut fragments, by calling `av_write_frame(ctx, NULL)` to write a fragment with the packets written so far. (This is only useful with other applications integrating libavformat, not from `ffmpeg`.)

`-min_frag_duration duration`

Don't create fragments that are shorter than *duration* microseconds long.

If more than one condition is specified, fragments are cut when one of the specified conditions is fulfilled. The exception to this is `-min_frag_duration`, which has to be fulfilled for any of the other conditions to apply.

Additionally, the way the output file is written can be adjusted through a few other options:

`-movflags empty_moov`

Write an initial moov atom directly at the start of the file, without describing any samples in it. Generally, an mdat/moov pair is written at the start of the file, as a normal MOV/MP4 file, containing only a short portion of the file. With this option set, there is no initial mdat atom, and the moov atom only describes the tracks but has a zero duration.

This option is implicitly set when writing ismv (Smooth Streaming) files.

`-movflags separate_moof`

Write a separate moof (movie fragment) atom for each track. Normally, packets for all tracks are written in a moof atom (which is slightly more efficient), but with this option set, the muxer writes one moof/mdat pair for each track, making it easier to separate tracks.

This option is implicitly set when writing ismv (Smooth Streaming) files.

`-movflags faststart`

Run a second pass moving the index (moov atom) to the beginning of the file. This operation can take a while, and will not work in various situations such as fragmented output, thus it is not enabled by default.

`-movflags rtphint`

Add RTP hinting tracks to the output file.

`-movflags disable_chpl`

Disable Nero chapter markers (chpl atom). Normally, both Nero chapters and a QuickTime chapter track are written to the file. With this option set, only the QuickTime chapter track will be written. Nero chapters can cause failures when the file is reprocessed with certain tagging programs, like mp3Tag 2.61a and iTunes 11.3, most likely other versions are affected as well.

`-movflags omit_tfhd_offset`

Do not write any absolute base\_data\_offset in tfhd atoms. This avoids tying fragments to absolute byte positions in the file/streams.

`-movflags default_base_moof`

Similarly to the `omit_tfhd_offset`, this flag avoids writing the absolute `base_data_offset` field in `tfhd` atoms, but does so by using the new `default-base-is-moof` flag instead. This flag is new from 14496-12:2012. This may make the fragments easier to parse in certain circumstances (avoiding basing track fragment location calculations on the implicit end of the previous track fragment).

### 22.11.2 Example# TOC

Smooth Streaming content can be pushed in real time to a publishing point on IIS with this muxer.

Example:

```
ffmpeg -re <normal input/transcoding options> -movflags isml+frag_keyframe -f ismv http://server/publishingpoint.isml/Streams(Encoder1)
```

### 22.11.3 Audible AAX# TOC

Audible AAX files are encrypted M4B files, and they can be decrypted by specifying a 4 byte activation secret.

```
ffmpeg -activation_bytes 1CEB00DA -i test.aax -vn -c:a copy output.mp4
```

## 22.12 mp3# TOC

The MP3 muxer writes a raw MP3 stream with the following optional features:

- An ID3v2 metadata header at the beginning (enabled by default). Versions 2.3 and 2.4 are supported, the `id3v2_version` private option controls which one is used (3 or 4). Setting `id3v2_version` to 0 disables the ID3v2 header completely.

The muxer supports writing attached pictures (APIC frames) to the ID3v2 header. The pictures are supplied to the muxer in form of a video stream with a single packet. There can be any number of those streams, each will correspond to a single APIC frame. The stream metadata tags *title* and *comment* map to APIC *description* and *picture type* respectively. See <http://id3.org/id3v2.4.0-frames> for allowed picture types.

Note that the APIC frames must be written at the beginning, so the muxer will buffer the audio frames until it gets all the pictures. It is therefore advised to provide the pictures as soon as possible to avoid excessive buffering.

- A Xing/LAME frame right after the ID3v2 header (if present). It is enabled by default, but will be written only if the output is seekable. The `write_xing` private option can be used to disable it. The frame contains various information that may be useful to the decoder, like the audio duration or encoder delay.
- A legacy ID3v1 tag at the end of the file (disabled by default). It may be enabled with the `write_id3v1` private option, but as its capabilities are very limited, its usage is not recommended.

Examples:

Write an mp3 with an ID3v2.3 header and an ID3v1 footer:

```
ffmpeg -i INPUT -id3v2_version 3 -write_id3v1 1 out.mp3
```

To attach a picture to an mp3 file select both the audio and the picture stream with map:

```
ffmpeg -i input.mp3 -i cover.png -c copy -map 0 -map 1  
-metadata:s:v title="Album cover" -metadata:s:v comment="Cover (Front)" out.mp3
```

Write a "clean" MP3 without any extra features:

```
ffmpeg -i input.wav -write_xing 0 -id3v2_version 0 out.mp3
```

## 22.13 mpegts# TOC

MPEG transport stream muxer.

This muxer implements ISO 13818-1 and part of ETSI EN 300 468.

The recognized metadata settings in mpegts muxer are `service_provider` and `service_name`. If they are not set the default for `service_provider` is "FFmpeg" and the default for `service_name` is "Service01".

### 22.13.1 Options# TOC

The muxer options are:

`-mpegts_original_network_id number`

Set the `original_network_id` (default 0x0001). This is unique identifier of a network in DVB. Its main use is in the unique identification of a service through the path `Original_Network_ID`, `Transport_Stream_ID`.

`-mpegts_transport_stream_id number`

Set the `transport_stream_id` (default 0x0001). This identifies a transponder in DVB.

`-mpegts_service_id number`

Set the `service_id` (default 0x0001) also known as program in DVB.

`-mpegts_service_type number`

Set the program `service_type` (default *digital\_tv*), see below a list of pre defined values.

`-mpegts_pmt_start_pid number`

Set the first PID for PMT (default 0x1000, max 0x1f00).

`-mpegts_start_pid number`

Set the first PID for data packets (default 0x0100, max 0x0f00).

`-mpegts_m2ts_mode number`

Enable m2ts mode if set to 1. Default value is -1 which disables m2ts mode.

`-muxrate number`

Set a constant muxrate (default VBR).

`-pcr_period number`

Override the default PCR retransmission time (default 20ms), ignored if variable muxrate is selected.

`pat_period number`

Maximal time in seconds between PAT/PMT tables.

`sdt_period number`

Maximal time in seconds between SDT tables.

`-pes_payload_size number`

Set minimum PES packet payload in bytes.

`-mpegts_flags flags`

Set flags (see below).

`-mpegts_copyts number`

Preserve original timestamps, if value is set to 1. Default value is -1, which results in shifting timestamps so that they start from 0.

`-tables_version number`

Set PAT, PMT and SDT version (default 0, valid values are from 0 to 31, inclusively). This option allows updating stream structure so that standard consumer may detect the change. To do so, reopen output AVFormatContext (in case of API usage) or restart ffmpeg instance, cyclically changing `tables_version` value:

```
ffmpeg -i source1.ts -codec copy -f mpegts -tables_version 0 udp://1.1.1.1:1111
ffmpeg -i source2.ts -codec copy -f mpegts -tables_version 1 udp://1.1.1.1:1111
...
ffmpeg -i source3.ts -codec copy -f mpegts -tables_version 31 udp://1.1.1.1:1111
ffmpeg -i source1.ts -codec copy -f mpegts -tables_version 0 udp://1.1.1.1:1111
ffmpeg -i source2.ts -codec copy -f mpegts -tables_version 1 udp://1.1.1.1:1111
...
```

Option `mpegts_service_type` accepts the following values:

`hex_value`

Any hexadecimal value between 0x01 to 0xff as defined in ETSI 300 468.

`digital_tv`

Digital TV service.

`digital_radio`

Digital Radio service.

`teletext`

Teletext service.

`advanced_codec_digital_radio`

Advanced Codec Digital Radio service.

`mpeg2_digital_hdtv`

MPEG2 Digital HDTV service.

`advanced_codec_digital_sdtv`

Advanced Codec Digital SDTV service.

`advanced_codec_digital_hdtv`

Advanced Codec Digital HDTV service.

Option `mpegts_flags` may take a set of such flags:

`resend_headers`

Reemit PAT/PMT before writing the next packet.

`latm`

Use LATM packetization for AAC.

`pat_pmt_at_frames`

Reemit PAT and PMT at each video frame.

### 22.13.2 Example# TOC

```
ffmpeg -i file.mpg -c copy \
-mpegts_original_network_id 0x1122 \
-mpegts_transport_stream_id 0x3344 \
-mpegts_service_id 0x5566 \
-mpegts_pmt_start_pid 0x1500 \
-mpegts_start_pid 0x150 \
-metadata service_provider="Some provider" \
-metadata service_name="Some Channel" \
-y out.ts
```

### 22.14 null# TOC

Null muxer.

This muxer does not generate any output file, it is mainly useful for testing or benchmarking purposes.

For example to benchmark decoding with `ffmpeg` you can use the command:

```
ffmpeg -benchmark -i INPUT -f null out.null
```

Note that the above command does not read or write the `out.null` file, but specifying the output file is required by the `ffmpeg` syntax.

Alternatively you can write the command as:

```
ffmpeg -benchmark -i INPUT -f null -
```

### 22.15 nut# TOC

`-syncpoints flags`

Change the syncpoint usage in nut:

*default* use the normal low-overhead seeking aids.

*none* do not use the syncpoints at all, reducing the overhead but making the stream non-seekable;

Use of this option is not recommended, as the resulting files are very damage sensitive and seeking is not possible. Also in general the overhead from syncpoints is negligible. Note, `-write_index 0` can be used to disable all growing data tables, allowing to mux endless streams with limited memory and without these disadvantages.

*timestamped* extend the syncpoint with a wallclock field.

The *none* and *timestamped* flags are experimental.

`-write_index bool`



Write index at the end, the default is to write an index.

```
ffmpeg -i INPUT -f_strict experimental -syncpoints none - | processor
```

## 22.16 ogg# TOC

Ogg container muxer.

`-page_duration duration`

Preferred page duration, in microseconds. The muxer will attempt to create pages that are approximately *duration* microseconds long. This allows the user to compromise between seek granularity and container overhead. The default is 1 second. A value of 0 will fill all segments, making pages as large as possible. A value of 1 will effectively use 1 packet-per-page in most situations, giving a small seek granularity at the cost of additional container overhead.

`-serial_offset value`

Serial value from which to set the streams serial number. Setting it to different and sufficiently large values ensures that the produced ogg files can be safely chained.

## 22.17 segment, stream\_segment, ssegment# TOC

Basic stream segmenter.

This muxer outputs streams to a number of separate files of nearly fixed duration. Output filename pattern can be set in a fashion similar to image2, or by using a `strftime` template if the `strftime` option is enabled.

`stream_segment` is a variant of the muxer used to write to streaming output formats, i.e. which do not require global headers, and is recommended for outputting e.g. to MPEG transport stream segments. `ssegment` is a shorter alias for `stream_segment`.

Every segment starts with a keyframe of the selected reference stream, which is set through the `reference_stream` option.

Note that if you want accurate splitting for a video file, you need to make the input key frames correspond to the exact splitting times expected by the segmenter, or the segment muxer will start the new segment with the key frame found next after the specified start time.

The segment muxer works best with a single constant frame rate video.

Optionally it can generate a list of the created segments, by setting the option `segment_list`. The list type is specified by the `segment_list_type` option. The entry filenames in the segment list are set by default to the basename of the corresponding segment files.

See also the hls muxer, which provides a more specific implementation for HLS segmentation.

## 22.17.1 Options# TOC

The segment muxer supports the following options:

`reference_stream specifier`

Set the reference stream, as specified by the string *specifier*. If *specifier* is set to `auto`, the reference is chosen automatically. Otherwise it must be a stream specifier (see the “Stream specifiers” chapter in the ffmpeg manual) which specifies the reference stream. The default value is `auto`.

`segment_format format`

Override the inner container format, by default it is guessed by the filename extension.

`segment_format_options options_list`

Set output format options using a `:`-separated list of key=value parameters. Values containing the `:` special character must be escaped.

`segment_list name`

Generate also a listfile named *name*. If not specified no listfile is generated.

`segment_list_flags flags`

Set flags affecting the segment list generation.

It currently supports the following flags:

‘cache’

Allow caching (only affects M3U8 list files).

‘live’

Allow live-friendly file generation.

`segment_list_size size`

Update the list file so that it contains at most *size* segments. If 0 the list file will contain all the segments. Default value is 0.

`segment_list_entry_prefix prefix`

Prepend *prefix* to each entry. Useful to generate absolute paths. By default no prefix is applied.

`segment_list_type type`

Select the listing format.

The following values are recognized:

‘flat’

Generate a flat list for the created segments, one segment per line.

‘csv, ext’

Generate a list for the created segments, one segment per line, each line matching the format (comma-separated values):

*segment\_filename, segment\_start\_time, segment\_end\_time*

*segment\_filename* is the name of the output file generated by the muxer according to the provided pattern. CSV escaping (according to RFC4180) is applied if required.

*segment\_start\_time* and *segment\_end\_time* specify the segment start and end time expressed in seconds.

A list file with the suffix ".csv" or ".ext" will auto-select this format.

‘ext’ is deprecated in favor of ‘csv’.

‘ffconcat’

Generate an ffconcat file for the created segments. The resulting file can be read using the FFmpeg concat demuxer.

A list file with the suffix ".ffcat" or ".ffconcat" will auto-select this format.

‘m3u8’

Generate an extended M3U8 file, version 3, compliant with <http://tools.ietf.org/id/draft-pantos-http-live-streaming>.

A list file with the suffix ".m3u8" will auto-select this format.

If not specified the type is guessed from the list file name suffix.

`segment_time time`

Set segment duration to *time*, the value must be a duration specification. Default value is "2". See also the `segment_times` option.

Note that splitting may not be accurate, unless you force the reference stream key-frames at the given time. See the introductory notice and the examples below.

`segment_atclocktime 1/0`

If set to "1" split at regular clock time intervals starting from 00:00 o'clock. The *time* value specified in `segment_time` is used for setting the length of the splitting interval.

For example with `segment_time` set to "900" this makes it possible to create files at 12:00 o'clock, 12:15, 12:30, etc.

Default value is "0".

`segment_time_delta delta`

Specify the accuracy time when selecting the start time for a segment, expressed as a duration specification. Default value is "0".

When delta is specified a key-frame will start a new segment if its PTS satisfies the relation:

$$\text{PTS} \geq \text{start\_time} - \text{time\_delta}$$

This option is useful when splitting video content, which is always split at GOP boundaries, in case a key frame is found just before the specified split time.

In particular may be used in combination with the `ffmpeg` option *force\_key\_frames*. The key frame times specified by *force\_key\_frames* may not be set accurately because of rounding issues, with the consequence that a key frame time may result set just before the specified time. For constant frame rate videos a value of  $1/(2*\text{frame\_rate})$  should address the worst case mismatch between the specified time and the time set by *force\_key\_frames*.

`segment_times times`

Specify a list of split points. *times* contains a list of comma separated duration specifications, in increasing order. See also the `segment_time` option.

`segment_frames frames`

Specify a list of split video frame numbers. *frames* contains a list of comma separated integer numbers, in increasing order.

This option specifies to start a new segment whenever a reference stream key frame is found and the sequential number (starting from 0) of the frame is greater or equal to the next value in the list.

`segment_wrap limit`

Wrap around segment index once it reaches *limit*.

`segment_start_number` *number*

Set the sequence number of the first segment. Defaults to 0.

`strftime` *1/0*

Use the `strftime` function to define the name of the new segments to write. If this is selected, the output segment name must contain a `strftime` function template. Default value is 0.

`break_non_keyframes` *1/0*

If enabled, allow segments to start on frames other than keyframes. This improves behavior on some players when the time between keyframes is inconsistent, but may make things worse on others, and can cause some oddities during seeking. Defaults to 0.

`reset_timestamps` *1/0*

Reset timestamps at the begin of each segment, so that each segment will start with near-zero timestamps. It is meant to ease the playback of the generated segments. May not work with some combinations of muxers/codecs. It is set to 0 by default.

`initial_offset` *offset*

Specify timestamp offset to apply to the output packet timestamps. The argument must be a time duration specification, and defaults to 0.

## 22.17.2 Examples# TOC

- Remux the content of file `in.mkv` to a list of segments `out-000.nut`, `out-001.nut`, etc., and write the list of generated segments to `out.list`:

```
ffmpeg -i in.mkv -codec copy -map 0 -f segment -segment_list out.list out%03d.nut
```

- Segment input and set output format options for the output segments:

```
ffmpeg -i in.mkv -f segment -segment_time 10 -segment_format_options movflags=+faststart out%03d.mp4
```

- Segment the input file according to the split points specified by the `segment_times` option:

```
ffmpeg -i in.mkv -codec copy -map 0 -f segment -segment_list out.csv -segment_times 1,2,3,5,8,13,21 out%03d.nut
```

- Use the `ffmpeg force_key_frames` option to force key frames in the input at the specified location, together with the segment option `segment_time_delta` to account for possible roundings operated when setting key frame times.

```
ffmpeg -i in.mkv -force_key_frames 1,2,3,5,8,13,21 -codec:v mpeg4 -codec:a pcm_s16le -map 0 \
-f segment -segment_list out.csv -segment_times 1,2,3,5,8,13,21 -segment_time_delta 0.05 out%03d.nut
```

In order to force key frames on the input file, transcoding is required.

- Segment the input file by splitting the input file according to the frame numbers sequence specified with the `segment_frames` option:

```
ffmpeg -i in.mkv -codec copy -map 0 -f segment -segment_list out.csv -segment_frames 100,200,300,500,800 out%03d.nut
```

- Convert the `in.mkv` to TS segments using the `libx264` and `libfaac` encoders:

```
ffmpeg -i in.mkv -map 0 -codec:v libx264 -codec:a libfaac -f ssegment -segment_list out.list out%03d.ts
```

- Segment the input file, and create an M3U8 live playlist (can be used as live HLS source):

```
ffmpeg -re -i in.mkv -codec copy -map 0 -f segment -segment_list playlist.m3u8 \
-segment_list_flags +live -segment_time 10 out%03d.mkv
```

## 22.18 smoothstreaming# TOC

Smooth Streaming muxer generates a set of files (Manifest, chunks) suitable for serving with conventional web server.

`window_size`

Specify the number of fragments kept in the manifest. Default 0 (keep all).

`extra_window_size`

Specify the number of fragments kept outside of the manifest before removing from disk. Default 5.

`lookahead_count`

Specify the number of lookahead fragments. Default 2.

`min_frag_duration`

Specify the minimum fragment duration (in microseconds). Default 5000000.

`remove_at_exit`

Specify whether to remove all fragments when finished. Default 0 (do not remove).

## 22.19 tee# TOC

The tee muxer can be used to write the same data to several files or any other kind of muxer. It can be used, for example, to both stream a video to the network and save it to disk at the same time.

It is different from specifying several outputs to the `ffmpeg` command-line tool because the audio and video data will be encoded only once with the tee muxer; encoding can be a very expensive process. It is not useful when using the `libavformat` API directly because it is then possible to feed the same packets to several muxers directly.

The slave outputs are specified in the file name given to the muxer, separated by '|'. If any of the slave name contains the '|' separator, leading or trailing spaces or any special character, it must be escaped (see (ffmpeg-utils)the "Quoting and escaping" section in the ffmpeg-utils(1) manual).

Muxer options can be specified for each slave by prepending them as a list of *key=value* pairs separated by ':', between square brackets. If the options values contain a special character or the ':' separator, they must be escaped; note that this is a second level escaping.

The following special options are also recognized:

`f`

Specify the format name. Useful if it cannot be guessed from the output name suffix.

`bsfs[ /spec ]`

Specify a list of bitstream filters to apply to the specified output.

It is possible to specify to which streams a given bitstream filter applies, by appending a stream specifier to the option separated by /. *spec* must be a stream specifier (see Format stream specifiers). If the stream specifier is not specified, the bitstream filters will be applied to all streams in the output.

Several bitstream filters can be specified, separated by ",".

`select`

Select the streams that should be mapped to the slave output, specified by a stream specifier. If not specified, this defaults to all the input streams.

## 22.19.1 Examples# TOC

- Encode something and both archive it in a WebM file and stream it as MPEG-TS over UDP (the streams need to be explicitly mapped):

```
ffmpeg -i ... -c:v libx264 -c:a mp2 -f tee -map 0:v -map 0:a  
"archive-20121107.mkv|[f=mpegts]udp://10.0.1.255:1234/"
```

- Use ffmpeg to encode the input, and send the output to three different destinations. The `dump_extra` bitstream filter is used to add extradata information to all the output video keyframes packets, as requested by the MPEG-TS format. The `select` option is applied to `out.aac` in order to make it contain only audio packets.

```
ffmpeg -i ... -map 0 -flags +global_header -c:v libx264 -c:a aac -strict experimental  
-f tee "[bsfs/v=dump_extra]out.ts|[movflags=+faststart]out.mp4|[select=a]out.aac"
```

- As below, but select only stream `a:1` for the audio output. Note that a second level escaping must be performed, as `":"` is a special character used to separate options.

```
ffmpeg -i ... -map 0 -flags +global_header -c:v libx264 -c:a aac -strict experimental  
-f tee "[bsfs/v=dump_extra]out.ts|[movflags=+faststart]out.mp4|[select='a:1\']out.aac"
```

Note: some codecs may need different options depending on the output format; the auto-detection of this can not work with the tee muxer. The main example is the `global_header` flag.

## 22.20 webm\_dash\_manifest# TOC

WebM DASH Manifest muxer.

This muxer implements the WebM DASH Manifest specification to generate the DASH manifest XML. It also supports manifest generation for DASH live streams.

For more information see:

- WebM DASH Specification:  
<https://sites.google.com/a/webmproject.org/wiki/adaptive-streaming/webm-dash-specification>
- ISO DASH Specification:  
[http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274\\_ISO\\_IEC\\_23009-1\\_2014.zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274_ISO_IEC_23009-1_2014.zip)

### 22.20.1 Options# TOC

This muxer supports the following options:

`adaptation_sets`

This option has the following syntax: "id=x,streams=a,b,c id=y,streams=d,e" where x and y are the unique identifiers of the adaptation sets and a,b,c,d and e are the indices of the corresponding audio and video streams. Any number of adaptation sets can be added using this option.

`live`

Set this to 1 to create a live stream DASH Manifest. Default: 0.

`chunk_start_index`

Start index of the first chunk. This will go in the 'startNumber' attribute of the 'SegmentTemplate' element in the manifest. Default: 0.

`chunk_duration_ms`

Duration of each chunk in milliseconds. This will go in the 'duration' attribute of the 'SegmentTemplate' element in the manifest. Default: 1000.

`utc_timing_url`

URL of the page that will return the UTC timestamp in ISO format. This will go in the 'value' attribute of the 'UTCTiming' element in the manifest. Default: None.



time\_shift\_buffer\_depth

Smallest time (in seconds) shifting buffer for which any Representation is guaranteed to be available. This will go in the 'timeShiftBufferDepth' attribute of the 'MPD' element. Default: 60.

```
minimum_update_period
```

Minimum update period (in seconds) of the manifest. This will go in the 'minimumUpdatePeriod' attribute of the 'MPD' element. Default: 0.

### 22.20.2 Example# TOC

```
ffmpeg -f webm_dash_manifest -i video1.webm \
-f webm_dash_manifest -i video2.webm \
-f webm_dash_manifest -i audio1.webm \
-f webm_dash_manifest -i audio2.webm \
-map 0 -map 1 -map 2 -map 3 \
-c copy \
-f webm_dash_manifest \
-adaptation_sets "id=0,streams=0,1 id=1,streams=2,3" \
manifest.xml
```

## 22.21 webm\_chunk# TOC

## WebM Live Chunk Muxer.

This muxer writes out WebM headers and chunks as separate files which can be consumed by clients that support WebM Live streams via DASH.

### 22.21.1 Options# TOC

This muxer supports the following options:

chunk\_start\_index

Index of the first chunk (defaults to 0).

header

Filename of the header where the initialization data will be written.

audio\_chunk\_duration

Duration of each audio chunk in milliseconds (defaults to 5000).

### 22.21.2 Example# TOC

```
ffmpeg -f v4l2 -i /dev/video0 \
-f alsa -i hw:0 \
-map 0:0 \
-c:v libvpx-vp9 \
```

```

-s 640x360 -keyint_min 30 -g 30 \
-f webm_chunk \
-header webm_live_video_360.hdr \
-chunk_start_index 1 \
webm_live_video_360_%d.chk \
-map 1:0 \
-c:a libvorbis \
-b:a 128k \
-f webm_chunk \
-header webm_live_audio_128.hdr \
-chunk_start_index 1 \
-audio_chunk_duration 1000 \
webm_live_audio_128_%d.chk

```

## 23 Metadata# TOC

FFmpeg is able to dump metadata from media files into a simple UTF-8-encoded INI-like text file and then load it back using the metadata muxer/demuxer.

The file format is as follows:

1. A file consists of a header and a number of metadata tags divided into sections, each on its own line.
2. The header is a ‘;FFMETADATA1’ string, followed by a version number (now 1).
3. Metadata tags are of the form ‘key=value’
4. Immediately after header follows global metadata
5. After global metadata there may be sections with per-stream/per-chapter metadata.
6. A section starts with the section name in uppercase (i.e. STREAM or CHAPTER) in brackets (‘[’, ‘]’) and ends with next section or end of file.
7. At the beginning of a chapter section there may be an optional timebase to be used for start/end values. It must be in form ‘TIMEBASE=num/den’, where *num* and *den* are integers. If the timebase is missing then start/end times are assumed to be in milliseconds.

Next a chapter section must contain chapter start and end times in form ‘START=num’, ‘END=num’, where *num* is a positive integer.

8. Empty lines and lines starting with ‘;’ or ‘#’ are ignored.
9. Metadata keys or values containing special characters (‘=’, ‘;’, ‘#’, ‘\’ and a newline) must be escaped with a backslash ‘\’.
10. Note that whitespace in metadata (e.g. ‘foo = bar’) is considered to be a part of the tag (in the example above key is ‘foo ’, value is ‘ bar’).

A ffmadata file might look like this:

```

;FFMETADATA1
title=bike\shed
;this is a comment
artist=FFmpeg troll team

```

```

[CHAPTER]
TIMEBASE=1/1000

```

```
START=0
#chapter ends at 0:01:00
END=60000
title=chapter \#1
[STREAM]
title=multi\
line
```

By using the `ffmetadata` muxer and demuxer it is possible to extract metadata from an input file to an `ffmetadata` file, and then transcode the file into an output file with the edited `ffmetadata` file.

Extracting an `ffmetadata` file with `ffmpeg` goes as follows:

```
ffmpeg -i INPUT -f ffmetadata FFMETADATAFILE
```

Reinserting edited metadata information from the `FFMETADATAFILE` file can be done as:

```
ffmpeg -i INPUT -i FFMETADATAFILE -map_metadata 1 -codec copy OUTPUT
```

## 24 Protocols# TOC

Protocols are configured elements in `FFmpeg` that enable access to resources that require specific protocols.

When you configure your `FFmpeg` build, all the supported protocols are enabled by default. You can list all available ones using the configure option "`--list-protocols`".

You can disable all the protocols using the configure option "`--disable-protocols`", and selectively enable a protocol using the option "`--enable-protocol=PROTOCOL`", or you can disable a particular protocol using the option "`--disable-protocol=PROTOCOL`".

The option "`-protocols`" of the `ff*` tools will display the list of supported protocols.

A description of the currently available protocols follows.

### 24.1 async# TOC

Asynchronous data filling wrapper for input stream.

Fill data in a background thread, to decouple I/O operation from demux thread.

```
async:URL
async:http://host/resource
async:cache:http://host/resource
```

## 24.2 bluray# TOC

Read BluRay playlist.

The accepted options are:

angle

BluRay angle

chapter

Start chapter (1...N)

playlist

Playlist to read (BDMV/PLAYLIST/?????.mpls)

Examples:

Read longest playlist from BluRay mounted to /mnt/bluray:

```
bluray:/mnt/bluray
```

Read angle 2 of playlist 4 from BluRay mounted to /mnt/bluray, start from chapter 2:

```
-playlist 4 -angle 2 -chapter 2 bluray:/mnt/bluray
```

## 24.3 cache# TOC

Caching wrapper for input stream.

Cache the input stream to temporary file. It brings seeking capability to live streams.

cache:URL

## 24.4 concat# TOC

Physical concatenation protocol.

Read and seek from many resources in sequence as if they were a unique resource.

A URL accepted by this protocol has the syntax:

concat:URL1|URL2|...|URLN

where *URL1*, *URL2*, ..., *URLN* are the urls of the resource to be concatenated, each one possibly specifying a distinct protocol.

For example to read a sequence of files `split1.mpeg`, `split2.mpeg`, `split3.mpeg` with `ffplay` use the command:

```
ffplay concat:split1.mpeg\|split2.mpeg\|split3.mpeg
```

Note that you may need to escape the character "|" which is special for many shells.

## 24.5 crypto# TOC

AES-encrypted stream reading protocol.

The accepted options are:

`key`

Set the AES decryption key binary block from given hexadecimal representation.

`iv`

Set the AES decryption initialization vector binary block from given hexadecimal representation.

Accepted URL formats:

```
crypto:URL  
crypto+URL
```

## 24.6 data# TOC

Data in-line in the URI. See [http://en.wikipedia.org/wiki/Data\\_URI\\_scheme](http://en.wikipedia.org/wiki/Data_URI_scheme).

For example, to convert a GIF file given inline with `ffmpeg`:

```
ffmpeg -i "data:image/gif;base64,R0lGODdhCAAIAMIEAAAAAAAAA//8AAP//AP//////////////////ywAAAAACAAIAAADF0gEDLojDgdGiJdJqUX02iB4E8Q9jUMkADs=" smiley.png
```

## 24.7 file# TOC

File access protocol.

Read from or write to a file.

A file URL can have the form:

```
file:filename
```

where *filename* is the path of the file to read.

An URL that does not have a protocol prefix will be assumed to be a file URL. Depending on the build, an URL that looks like a Windows path with the drive letter at the beginning will also be assumed to be a file URL (usually not the case in builds for unix-like systems).

For example to read from a file `input.mpeg` with `ffmpeg` use the command:

```
ffmpeg -i file:input.mpeg output.mpeg
```

This protocol accepts the following options:

`truncate`

Truncate existing files on write, if set to 1. A value of 0 prevents truncating. Default value is 1.

`blocksize`

Set I/O operation maximum block size, in bytes. Default value is `INT_MAX`, which results in not limiting the requested block size. Setting this value reasonably low improves user termination request reaction time, which is valuable for files on slow medium.

## 24.8 ftp# TOC

FTP (File Transfer Protocol).

Read from or write to remote resources using FTP protocol.

Following syntax is required.

```
ftp://[user[:password]@]server[:port]/path/to/remote/resource.mpeg
```

This protocol accepts the following options.

`timeout`

Set timeout in microseconds of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

`ftp-anonymous-password`

Password used when login as anonymous user. Typically an e-mail address should be used.

`ftp-write-seekable`

Control seekability of connection during encoding. If set to 1 the resource is supposed to be seekable, if set to 0 it is assumed not to be seekable. Default value is 0.

NOTE: Protocol can be used as output, but it is recommended to not do it, unless special care is taken (tests, customized server configuration etc.). Different FTP servers behave in different way during seek operation. `ff*` tools may produce incomplete content due to server limitations.

## 24.9 gopher# TOC

Gopher protocol.

## 24.10 hls# TOC

Read Apple HTTP Live Streaming compliant segmented stream as a uniform one. The M3U8 playlists describing the segments can be remote HTTP resources or local files, accessed using the standard file protocol. The nested protocol is declared by specifying "+*proto*" after the hls URI scheme name, where *proto* is either "file" or "http".

```
hls+http://host/path/to/remote/resource.m3u8
hls+file://path/to/local/resource.m3u8
```

Using this protocol is discouraged - the hls demuxer should work just as well (if not, please report the issues) and is more complete. To use the hls demuxer instead, simply use the direct URLs to the m3u8 files.

## 24.11 http# TOC

HTTP (Hyper Text Transfer Protocol).

This protocol accepts the following options:

`seekable`

Control seekability of connection. If set to 1 the resource is supposed to be seekable, if set to 0 it is assumed not to be seekable, if set to -1 it will try to autodetect if it is seekable. Default value is -1.

`chunked_post`

If set to 1 use chunked Transfer-Encoding for posts, default is 1.

`content_type`

Set a specific content type for the POST messages.

`headers`

Set custom HTTP headers, can override built in default headers. The value must be a string encoding the headers.

`multiple_requests`

Use persistent connections if set to 1, default is 0.

`post_data`

Set custom HTTP post data.

`user-agent`

`user_agent`

Override the User-Agent header. If not specified the protocol will use a string describing the libavformat build. ("Lavf/<version>")

`timeout`

Set timeout in microseconds of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

`mime_type`

Export the MIME type.

`icy`

If set to 1 request ICY (SHOUTcast) metadata from the server. If the server supports this, the metadata has to be retrieved by the application by reading the `icy_metadata_headers` and `icy_metadata_packet` options. The default is 1.

`icy_metadata_headers`

If the server supports ICY metadata, this contains the ICY-specific HTTP reply headers, separated by newline characters.

`icy_metadata_packet`

If the server supports ICY metadata, and `icy` was set to 1, this contains the last non-empty metadata packet sent by the server. It should be polled in regular intervals by applications interested in mid-stream metadata updates.

`cookies`

Set the cookies to be sent in future requests. The format of each cookie is the same as the value of a Set-Cookie HTTP response field. Multiple cookies can be delimited by a newline character.

`offset`

Set initial byte offset.

`end_offset`

Try to limit the request to bytes preceding this offset.

`method`



When used as a client option it sets the HTTP method for the request.

When used as a server option it sets the HTTP method that is going to be expected from the client(s). If the expected and the received HTTP method do not match the client will be given a Bad Request response. When unset the HTTP method is not checked for now. This will be replaced by autodetection in the future.

## listen

If set to 1 enables experimental HTTP server. This can be used to send data when used as an output option, or read data from a client with HTTP POST when used as an input option. If set to 2 enables experimental mutli-client HTTP server. This is not yet implemented in ffmpeg.c or ffserver.c and thus must not be used as a command line option.

```
# Server side (sending):
ffmpeg -i somefile.ogg -c copy -listen 1 -f ogg http://server:port

# Client side (receiving):
ffmpeg -i http://server:port -c copy somefile.ogg

# Client can also be done with wget:
wget http://server:port -O somefile.ogg

# Server side (receiving):
ffmpeg -listen 1 -i http://server:port -c copy somefile.ogg

# Client side (sending):
ffmpeg -i somefile.ogg -chunked_post 0 -c copy -f ogg http://server:port

# Client can also be done with wget:
wget --post-file=somefile.ogg http://server:port
```

### 24.11.1 HTTP Cookies# TOC

Some HTTP requests will be denied unless cookie values are passed in with the request. The `cookies` option allows these cookies to be specified. At the very least, each cookie must specify a value along with a path and domain. HTTP requests that match both the domain and path will automatically include the cookie value in the HTTP Cookie header field. Multiple cookies can be delimited by a newline.

The required syntax to play a stream specifying a cookie is:

```
ffplay -cookies "nlqptid=nltd=tsn; path=/; domain=somedomain.com;" http://somedomain.com/somestream.m3u8
```

### 24.12 Icecast# TOC

Icecast protocol (stream to Icecast servers)

This protocol accepts the following options:

`ice_genre`

Set the stream genre.

`ice_name`

Set the stream name.

`ice_description`

Set the stream description.

`ice_url`

Set the stream website URL.

`ice_public`

Set if the stream should be public. The default is 0 (not public).

`user_agent`

Override the User-Agent header. If not specified a string of the form "Lavf/<version>" will be used.

`password`

Set the Icecast mountpoint password.

`content_type`

Set the stream content type. This must be set if it is different from audio/mpeg.

`legacy_icecast`

This enables support for Icecast versions < 2.4.0, that do not support the HTTP PUT method but the SOURCE method.

`icecast://[username[:password]@]server:port/mountpoint`

## **24.13 mmst# TOC**

MMS (Microsoft Media Server) protocol over TCP.

## **24.14 mmsh# TOC**

MMS (Microsoft Media Server) protocol over HTTP.

The required syntax is:

```
mmsch://server[:port][[/app][/playpath]]
```

## 24.15 md5# TOC

MD5 output protocol.

Computes the MD5 hash of the data to be written, and on close writes this to the designated output or stdout if none is specified. It can be used to test muxers without writing an actual file.

Some examples follow.

```
# Write the MD5 hash of the encoded AVI file to the file output.avi.md5.
ffmpeg -i input.flv -f avi -y md5:output.avi.md5
```

```
# Write the MD5 hash of the encoded AVI file to stdout.
ffmpeg -i input.flv -f avi -y md5:
```

Note that some formats (typically MOV) require the output protocol to be seekable, so they will fail with the MD5 output protocol.

## 24.16 pipe# TOC

UNIX pipe access protocol.

Read and write from UNIX pipes.

The accepted syntax is:

```
pipe:[number]
```

*number* is the number corresponding to the file descriptor of the pipe (e.g. 0 for stdin, 1 for stdout, 2 for stderr). If *number* is not specified, by default the stdout file descriptor will be used for writing, stdin for reading.

For example to read from stdin with `ffmpeg`:

```
cat test.wav | ffmpeg -i pipe:0
# ...this is the same as...
cat test.wav | ffmpeg -i pipe:
```

For writing to stdout with `ffmpeg`:

```
ffmpeg -i test.wav -f avi pipe:1 | cat > test.avi
# ...this is the same as...
ffmpeg -i test.wav -f avi pipe: | cat > test.avi
```

This protocol accepts the following options:

blocksize

Set I/O operation maximum block size, in bytes. Default value is `INT_MAX`, which results in not limiting the requested block size. Setting this value reasonably low improves user termination request reaction time, which is valuable if data transmission is slow.

Note that some formats (typically MOV), require the output protocol to be seekable, so they will fail with the pipe output protocol.

## 24.17 rtmp# TOC

Real-Time Messaging Protocol.

The Real-Time Messaging Protocol (RTMP) is used for streaming multimedia content across a TCP/IP network.

The required syntax is:

```
rtmp://[username:password@]server[:port][/app][/instance][/playpath]
```

The accepted parameters are:

username

An optional username (mostly for publishing).

password

An optional password (mostly for publishing).

server

The address of the RTMP server.

port

The number of the TCP port to use (by default is 1935).

app

It is the name of the application to access. It usually corresponds to the path where the application is installed on the RTMP server (e.g. `/ondemand/`, `/flash/live/`, etc.). You can override the value parsed from the URI through the `rtmp_app` option, too.

playpath

It is the path or name of the resource to play with reference to the application specified in *app*, may be prefixed by "mp4:". You can override the value parsed from the URI through the `rtmp_playpath` option, too.

`listen`

Act as a server, listening for an incoming connection.

`timeout`

Maximum time to wait for the incoming connection. Implies `listen`.

Additionally, the following parameters can be set via command line options (or in code via `AVOptions`):

`rtmp_app`

Name of application to connect on the RTMP server. This option overrides the parameter specified in the URI.

`rtmp_buffer`

Set the client buffer time in milliseconds. The default is 3000.

`rtmp_conn`

Extra arbitrary AMF connection parameters, parsed from a string, e.g. like `B:1 S:authMe O:1 NN:code:1.23 NS:flag:ok O:0`. Each value is prefixed by a single character denoting the type, B for Boolean, N for number, S for string, O for object, or Z for null, followed by a colon. For Booleans the data must be either 0 or 1 for FALSE or TRUE, respectively. Likewise for Objects the data must be 0 or 1 to end or begin an object, respectively. Data items in subobjects may be named, by prefixing the type with 'N' and specifying the name before the value (i.e. `NB:myFlag:1`). This option may be used multiple times to construct arbitrary AMF sequences.

`rtmp_flashver`

Version of the Flash plugin used to run the SWF player. The default is LNX 9,0,124,2. (When publishing, the default is FMLE/3.0 (compatible; <libavformat version>).)

`rtmp_flush_interval`

Number of packets flushed in the same request (RTMPT only). The default is 10.

`rtmp_live`

Specify that the media is a live stream. No resuming or seeking in live streams is possible. The default value is `any`, which means the subscriber first tries to play the live stream specified in the playpath. If a live stream of that name is not found, it plays the recorded stream. The other possible values are `live` and `recorded`.

`rtmp_pageurl`

URL of the web page in which the media was embedded. By default no value will be sent.

`rtmp_playpath`

Stream identifier to play or to publish. This option overrides the parameter specified in the URL.

`rtmp_subscribe`

Name of live stream to subscribe to. By default no value will be sent. It is only sent if the option is specified or if `rtmp_live` is set to live.

`rtmp_swfhash`

SHA256 hash of the decompressed SWF file (32 bytes).

`rtmp_swfsize`

Size of the decompressed SWF file, required for SWFVerification.

`rtmp_swfurl`

URL of the SWF player for the media. By default no value will be sent.

`rtmp_swfverify`

URL to player swf file, compute hash/size automatically.

`rtmp_tcurl`

URL of the target stream. Defaults to `proto://host[:port]/app`.

For example to read with `ffplay` a multimedia resource named "sample" from the application "vod" from an RTMP server "myserver":

```
ffplay rtmp://myserver/vod/sample
```

To publish to a password protected server, passing the playpath and app names separately:

```
ffmpeg -re -i <input> -f flv -rtmp_playpath some/long/path -rtmp_app long/app/name rtmp://username:password@myserver/
```

## 24.18 rtmpe# TOC

Encrypted Real-Time Messaging Protocol.

The Encrypted Real-Time Messaging Protocol (RTMPE) is used for streaming multimedia content within standard cryptographic primitives, consisting of Diffie-Hellman key exchange and HMACSHA256, generating a pair of RC4 keys.

## 24.19 rtmps# TOC

Real-Time Messaging Protocol over a secure SSL connection.

The Real-Time Messaging Protocol (RTMPS) is used for streaming multimedia content across an encrypted connection.

## 24.20 rtmpt# TOC

Real-Time Messaging Protocol tunneled through HTTP.

The Real-Time Messaging Protocol tunneled through HTTP (RTMPT) is used for streaming multimedia content within HTTP requests to traverse firewalls.

## 24.21 rtmpte# TOC

Encrypted Real-Time Messaging Protocol tunneled through HTTP.

The Encrypted Real-Time Messaging Protocol tunneled through HTTP (RTMPTE) is used for streaming multimedia content within HTTP requests to traverse firewalls.

## 24.22 rtmpts# TOC

Real-Time Messaging Protocol tunneled through HTTPS.

The Real-Time Messaging Protocol tunneled through HTTPS (RTMPTS) is used for streaming multimedia content within HTTPS requests to traverse firewalls.

## 24.23 libsmbclient# TOC

libsmbclient permits one to manipulate CIFS/SMB network resources.

Following syntax is required.

```
smb://[[domain:]user[:password@]]server[/share[/path[/file]]]
```

This protocol accepts the following options.

`timeout`

Set timeout in milliseconds of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

`truncate`

Truncate existing files on write, if set to 1. A value of 0 prevents truncating. Default value is 1.

workgroup

Set the workgroup used for making connections. By default workgroup is not specified.

For more information see: <http://www.samba.org/>.

## 24.24 libssh# TOC

Secure File Transfer Protocol via libssh

Read from or write to remote resources using SFTP protocol.

Following syntax is required.

```
sftp://[user[:password]@]server[:port]/path/to/remote/resource.mpeg
```

This protocol accepts the following options.

timeout

Set timeout of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

truncate

Truncate existing files on write, if set to 1. A value of 0 prevents truncating. Default value is 1.

private\_key

Specify the path of the file containing private key to use during authorization. By default libssh searches for keys in the `~/ .ssh/` directory.

Example: Play a file stored on remote server.

```
ffplay sftp://user:password@server_address:22/home/user/resource.mpeg
```

## 24.25 librtmp rtmp, rtmpe, rtmps, rtmpt, rtmppte# TOC

Real-Time Messaging Protocol and its variants supported through librtmp.

Requires the presence of the librtmp headers and library during configuration. You need to explicitly configure the build with "`--enable-librtmp`". If enabled this will replace the native RTMP protocol.

This protocol provides most client functions and a few server functions needed to support RTMP, RTMP tunneled in HTTP (RTMPT), encrypted RTMP (RTMPE), RTMP over SSL/TLS (RTMPS) and tunneled variants of these encrypted types (RTMPTE, RTMPTS).



The required syntax is:

```
rtmp_proto://server[:port][/app][/playpath] options
```

where *rtmp\_proto* is one of the strings "rtmp", "rtmpt", "rtmpe", "rtmps", "rtmpte", "rtmpts" corresponding to each RTMP variant, and *server*, *port*, *app* and *playpath* have the same meaning as specified for the RTMP native protocol. *options* contains a list of space-separated options of the form *key=val*.

See the librtmp manual page (man 3 librtmp) for more information.

For example, to stream a file in real-time to an RTMP server using ffmpeg:

```
ffmpeg -re -i myfile -f flv rtmp://myserver/live/mystream
```

To play the same stream using ffplay:

```
ffplay "rtmp://myserver/live/mystream live=1"
```

## 24.26 rtp# TOC

Real-time Transport Protocol.

The required syntax for an RTP URL is: `rtp://hostname[:port][?option=val...]`

*port* specifies the RTP port to use.

The following URL options are supported:

`ttl=n`

Set the TTL (Time-To-Live) value (for multicast only).

`rtcpport=n`

Set the remote RTCP port to *n*.

`localrtpport=n`

Set the local RTP port to *n*.

`localrtcpport=n`

Set the local RTCP port to *n*.

`pkt_size=n`

Set max packet size (in bytes) to *n*.

`connect=0 | 1`

Do a `connect ( )` on the UDP socket (if set to 1) or not (if set to 0).

`sources=ip[ , ip]`

List allowed source IP addresses.

`block=ip[ , ip]`

List disallowed (blocked) source IP addresses.

`write_to_source=0 | 1`

Send packets to the source address of the latest received packet (if set to 1) or to a default remote address (if set to 0).

`localport=n`

Set the local RTP port to *n*.

This is a deprecated option. Instead, `localrtpport` should be used.

Important notes:

1. If `rtcpport` is not set the RTCP port will be set to the RTP port value plus 1.
2. If `localrtpport` (the local RTP port) is not set any available port will be used for the local RTP and RTCP ports.
3. If `localrtcpport` (the local RTCP port) is not set it will be set to the local RTP port value plus 1.

## 24.27 rtsp# TOC

Real-Time Streaming Protocol.

RTSP is not technically a protocol handler in libavformat, it is a demuxer and muxer. The demuxer supports both normal RTSP (with data transferred over RTP; this is used by e.g. Apple and Microsoft) and Real-RTSP (with data transferred over RDT).

The muxer can be used to send a stream using RTSP ANNOUNCE to a server supporting it (currently Darwin Streaming Server and Mischa Spiegelmock's RTSP server).

The required syntax for a RTSP url is:

`rtsp://hostname[:port]/path`

Options can be set on the `ffmpeg/ffplay` command line, or set in code via `AVOptions` or in `avformat_open_input`.

The following options are supported.

`initial_pause`

Do not start playing the stream immediately if set to 1. Default value is 0.

`rtsp_transport`

Set RTSP transport protocols.

It accepts the following values:

`'udp'`

Use UDP as lower transport protocol.

`'tcp'`

Use TCP (interleaving within the RTSP control channel) as lower transport protocol.

`'udp_multicast'`

Use UDP multicast as lower transport protocol.

`'http'`

Use HTTP tunneling as lower transport protocol, which is useful for passing proxies.

Multiple lower transport protocols may be specified, in that case they are tried one at a time (if the setup of one fails, the next one is tried). For the muxer, only the `'tcp'` and `'udp'` options are supported.

`rtsp_flags`

Set RTSP flags.

The following values are accepted:

`'filter_src'`

Accept packets only from negotiated peer address and port.

`'listen'`

Act as a server, listening for an incoming connection.

`'prefer_tcp'`

Try TCP for RTP transport first, if TCP is available as RTSP RTP transport.

Default value is 'none'.

`allowed_media_types`

Set media types to accept from the server.

The following flags are accepted:

'video'  
'audio'  
'data'

By default it accepts all media types.

`min_port`

Set minimum local UDP port. Default value is 5000.

`max_port`

Set maximum local UDP port. Default value is 65000.

`timeout`

Set maximum timeout (in seconds) to wait for incoming connections.

A value of -1 means infinite (default). This option implies the `rtsp_flags` set to 'listen'.

`reorder_queue_size`

Set number of packets to buffer for handling of reordered packets.

`sttimeout`

Set socket TCP I/O timeout in microseconds.

`user-agent`

Override User-Agent header. If not specified, it defaults to the libavformat identifier string.

When receiving data over UDP, the demuxer tries to reorder received packets (since they may arrive out of order, or packets may get lost totally). This can be disabled by setting the maximum demuxing delay to zero (via the `max_delay` field of `AVFormatContext`).

When watching multi-bitrate Real-RTSP streams with `ffplay`, the streams to display can be chosen with `-vst n` and `-ast n` for video and audio respectively, and can be switched on the fly by pressing `v` and `a`.

## 24.27.1 Examples# TOC

The following examples all make use of the `ffplay` and `ffmpeg` tools.

- Watch a stream over UDP, with a max reordering delay of 0.5 seconds:

```
ffplay -max_delay 500000 -rtsp_transport udp rtsp://server/video.mp4
```

- Watch a stream tunneled over HTTP:

```
ffplay -rtsp_transport http rtsp://server/video.mp4
```

- Send a stream in realtime to a RTSP server, for others to watch:

```
ffmpeg -re -i input -f rtsp -muxdelay 0.1 rtsp://server/live.sdp
```

- Receive a stream in realtime:

```
ffmpeg -rtsp_flags listen -i rtsp://ownaddress/live.sdp output
```

## 24.28 sap# TOC

Session Announcement Protocol (RFC 2974). This is not technically a protocol handler in libavformat, it is a muxer and demuxer. It is used for signalling of RTP streams, by announcing the SDP for the streams regularly on a separate port.

### 24.28.1 Muxer# TOC

The syntax for a SAP url given to the muxer is:

```
sap://destination[:port][?options]
```

The RTP packets are sent to *destination* on port *port*, or to port 5004 if no port is specified. *options* is a &-separated list. The following options are supported:

`announce_addr=address`

Specify the destination IP address for sending the announcements to. If omitted, the announcements are sent to the commonly used SAP announcement multicast address 224.2.127.254 (sap.mcast.net), or ff0e::2:7ffe if *destination* is an IPv6 address.

`announce_port=port`

Specify the port to send the announcements on, defaults to 9875 if not specified.

`ttl=t1`

Specify the time to live value for the announcements and RTP packets, defaults to 255.

`same_port=0/1`

If set to 1, send all RTP streams on the same port pair. If zero (the default), all streams are sent on unique ports, with each stream on a port 2 numbers higher than the previous. VLC/Live555 requires this to be set to 1, to be able to receive the stream. The RTP stack in libavformat for receiving requires all streams to be sent on unique ports.

Example command lines follow.

To broadcast a stream on the local subnet, for watching in VLC:

```
ffmpeg -re -i input -f sap sap://224.0.0.255?same_port=1
```

Similarly, for watching in `ffplay`:

```
ffmpeg -re -i input -f sap sap://224.0.0.255
```

And for watching in `ffplay`, over IPv6:

```
ffmpeg -re -i input -f sap sap://[ff0e::1:2:3:4]
```

## 24.28.2 Demuxer# TOC

The syntax for a SAP url given to the demuxer is:

```
sap://[address][:port]
```

*address* is the multicast address to listen for announcements on, if omitted, the default 224.2.127.254 (sap.mcast.net) is used. *port* is the port that is listened on, 9875 if omitted.

The demuxers listens for announcements on the given address and port. Once an announcement is received, it tries to receive that particular stream.

Example command lines follow.

To play back the first stream announced on the normal SAP multicast address:

```
ffplay sap://
```

To play back the first stream announced on one the default IPv6 SAP multicast address:

```
ffplay sap://[ff0e::2:7ffe]
```

## 24.29 sctp# TOC

Stream Control Transmission Protocol.

The accepted URL syntax is:

`sctp://host:port[?options]`

The protocol accepts the following options:

`listen`

If set to any value, listen for an incoming connection. Outgoing connection is done by default.

`max_streams`

Set the maximum number of streams. By default no limit is set.

## 24.30 srtp# TOC

Secure Real-time Transport Protocol.

The accepted options are:

`srtp_in_suite`

`srtp_out_suite`

Select input and output encoding suites.

Supported values:

`'AES_CM_128_HMAC_SHA1_80'`

`'SRTP_AES128_CM_HMAC_SHA1_80'`

`'AES_CM_128_HMAC_SHA1_32'`

`'SRTP_AES128_CM_HMAC_SHA1_32'`

`srtp_in_params`

`srtp_out_params`

Set input and output encoding parameters, which are expressed by a base64-encoded representation of a binary block. The first 16 bytes of this binary block are used as master key, the following 14 bytes are used as master salt.

## 24.31 subfile# TOC

Virtually extract a segment of a file or another stream. The underlying stream must be seekable.

Accepted options:

`start`

Start offset of the extracted segment, in bytes.

`end`

End offset of the extracted segment, in bytes.

Examples:

Extract a chapter from a DVD VOB file (start and end sectors obtained externally and multiplied by 2048):

```
subfile,,start,153391104,end,268142592,,:/media/dvd/VIDEO_TS/VTS_08_1.VOB
```

Play an AVI file directly from a TAR archive:

```
subfile,,start,183241728,end,366490624,, :archive.tar
```

## 24.32 tcp# TOC

Transmission Control Protocol.

The required syntax for a TCP url is:

```
tcp://hostname:port[?options]
```

*options* contains a list of &-separated options of the form *key=val*.

The list of supported options follows.

```
listen=1/0
```

Listen for an incoming connection. Default value is 0.

```
timeout=microseconds
```

Set raise error timeout, expressed in microseconds.

This option is only relevant in read mode: if no data arrived in more than this time interval, raise error.

```
listen_timeout=milliseconds
```

Set listen timeout, expressed in milliseconds.

The following example shows how to setup a listening TCP connection with `ffmpeg`, which is then accessed with `ffplay`:

```
ffmpeg -i input -f format tcp://hostname:port?listen  
ffplay tcp://hostname:port
```



## 24.33 `tls`# TOC

Transport Layer Security (TLS) / Secure Sockets Layer (SSL)

The required syntax for a TLS/SSL url is:

```
tls://hostname:port[?options]
```

The following parameters can be set via command line options (or in code via `AVOptions`):

```
ca_file, cafile=filename
```

A file containing certificate authority (CA) root certificates to treat as trusted. If the linked TLS library contains a default this might not need to be specified for verification to work, but not all libraries and setups have defaults built in. The file must be in OpenSSL PEM format.

```
tls_verify=1/0
```

If enabled, try to verify the peer that we are communicating with. Note, if using OpenSSL, this currently only makes sure that the peer certificate is signed by one of the root certificates in the CA database, but it does not validate that the certificate actually matches the host name we are trying to connect to. (With GnuTLS, the host name is validated as well.)

This is disabled by default since it requires a CA database to be provided by the caller in many cases.

```
cert_file, cert=filename
```

A file containing a certificate to use in the handshake with the peer. (When operating as server, in listen mode, this is more often required by the peer, while client certificates only are mandated in certain setups.)

```
key_file, key=filename
```

A file containing the private key for the certificate.

```
listen=1/0
```

If enabled, listen for connections on the provided port, and assume the server role in the handshake instead of the client role.

Example command lines:

To create a TLS/SSL server that serves an input stream.

```
ffmpeg -i input -f format tls://hostname:port?listen&cert=server.crt&key=server.key
```

To play back a stream from the TLS/SSL server using `ffplay`:

```
ffplay tls://hostname:port
```

## 24.34 udp# TOC

User Datagram Protocol.

The required syntax for an UDP URL is:

```
udp://hostname:port[?options]
```

*options* contains a list of &-separated options of the form *key=val*.

In case threading is enabled on the system, a circular buffer is used to store the incoming data, which allows one to reduce loss of data due to UDP socket buffer overruns. The *fifo\_size* and *overrun\_nonfatal* options are related to this buffer.

The list of supported options follows.

```
buffer_size=size
```

Set the UDP maximum socket buffer size in bytes. This is used to set either the receive or send buffer size, depending on what the socket is used for. Default is 64KB. See also *fifo\_size*.

```
localport=port
```

Override the local UDP port to bind with.

```
localaddr=addr
```

Choose the local IP address. This is useful e.g. if sending multicast and the host has multiple interfaces, where the user can choose which interface to send on by specifying the IP address of that interface.

```
pkt_size=size
```

Set the size in bytes of UDP packets.

```
reuse=1/0
```

Explicitly allow or disallow reusing UDP sockets.

```
ttl=t1
```

Set the time to live value (for multicast only).

```
connect=1/0
```

Initialize the UDP socket with `connect ( )`. In this case, the destination address can't be changed with `ff_udp_set_remote_url` later. If the destination address isn't known at the start, this option can be specified in `ff_udp_set_remote_url`, too. This allows finding out the source address for the packets

with `getsockname`, and makes writes return with `AVERROR(ECONNREFUSED)` if "destination unreachable" is received. For receiving, this gives the benefit of only receiving packets from the specified peer address/port.

`sources=address[,address]`

Only receive packets sent to the multicast group from one of the specified sender IP addresses.

`block=address[,address]`

Ignore packets sent to the multicast group from the specified sender IP addresses.

`fifo_size=units`

Set the UDP receiving circular buffer size, expressed as a number of packets with size of 188 bytes. If not specified defaults to  $7 \times 4096$ .

`overrun_nonfatal=1/0`

Survive in case of UDP receiving circular buffer overrun. Default value is 0.

`timeout=microseconds`

Set raise error timeout, expressed in microseconds.

This option is only relevant in read mode: if no data arrived in more than this time interval, raise error.

`broadcast=1/0`

Explicitly allow or disallow UDP broadcasting.

Note that broadcasting may not work properly on networks having a broadcast storm protection.

## 24.34.1 Examples# TOC

- Use `ffmpeg` to stream over UDP to a remote endpoint:

```
ffmpeg -i input -f format udp://hostname:port
```

- Use `ffmpeg` to stream in mpegts format over UDP using 188 sized UDP packets, using a large input buffer:

```
ffmpeg -i input -f mpegts udp://hostname:port?pkt_size=188&buffer_size=65535
```

- Use `ffmpeg` to receive over UDP from a remote endpoint:

```
ffmpeg -i udp://[multicast-address]:port ...
```

## 24.35 unix# TOC

Unix local socket

The required syntax for a Unix socket URL is:

```
unix://filepath
```

The following parameters can be set via command line options (or in code via `AVOptions`):

`timeout`

Timeout in ms.

`listen`

Create the Unix socket in listening mode.

## 25 Device Options# TOC

The `libavdevice` library provides the same interface as `libavformat`. Namely, an input device is considered like a demuxer, and an output device like a muxer, and the interface and generic device options are the same provided by `libavformat` (see the `ffmpeg-formats` manual).

In addition each input or output device may support so-called private options, which are specific for that component.

Options may be set by specifying *-option value* in the FFmpeg tools, or by setting the value explicitly in the device `AVFormatContext` options or using the `libavutil/opt.h` API for programmatic use.

## 26 Input Devices# TOC

Input devices are configured elements in FFmpeg which enable accessing the data coming from a multimedia device attached to your system.

When you configure your FFmpeg build, all the supported input devices are enabled by default. You can list all available ones using the configure option `"-list-indevs"`.

You can disable all the input devices using the configure option `"-disable-indevs"`, and selectively enable an input device using the option `"-enable-indev=INDEV"`, or you can disable a particular input device using the option `"-disable-indev=INDEV"`.

The option `"-devices"` of the `ff*` tools will display the list of supported input devices.

A description of the currently available input devices follows.

## 26.1 alsa# TOC

ALSA (Advanced Linux Sound Architecture) input device.

To enable this input device during configuration you need libasound installed on your system.

This device allows capturing from an ALSA device. The name of the device to capture has to be an ALSA card identifier.

An ALSA identifier has the syntax:

```
hw: CARD[ , DEV[ , SUBDEV ] ]
```

where the *DEV* and *SUBDEV* components are optional.

The three arguments (in order: *CARD*, *DEV*, *SUBDEV*) specify card number or identifier, device number and subdevice number (-1 means any).

To see the list of cards currently recognized by your system check the files `/proc/asound/cards` and `/proc/asound/devices`.

For example to capture with `ffmpeg` from an ALSA device with card id 0, you may run the command:

```
ffmpeg -f alsa -i hw:0 alsaout.wav
```

For more information see: <http://www.alsa-project.org/alsa-doc/alsa-lib/pcm.html>

### 26.1.1 Options# TOC

`sample_rate`

Set the sample rate in Hz. Default is 48000.

`channels`

Set the number of channels. Default is 2.

## 26.2 avfoundation# TOC

AVFoundation input device.

AVFoundation is the currently recommended framework by Apple for streamgrabbing on OSX >= 10.7 as well as on iOS. The older QTKit framework has been marked deprecated since OSX version 10.7.

The input filename has to be given in the following syntax:

```
-i "[[VIDEO]:[AUDIO]]"
```

The first entry selects the video input while the latter selects the audio input. The stream has to be specified by the device name or the device index as shown by the device list. Alternatively, the video and/or audio input device can be chosen by index using the `-video_device_index <INDEX>` and/or `-audio_device_index <INDEX>`, overriding any device name or index given in the input filename.

All available devices can be enumerated by using `-list_devices true`, listing all device names and corresponding indices.

There are two device name aliases:

`default`

Select the AVFoundation default device of the corresponding type.

`none`

Do not record the corresponding media type. This is equivalent to specifying an empty device name or index.

### 26.2.1 Options# TOC

AVFoundation supports the following options:

`-list_devices <TRUE|FALSE>`

If set to true, a list of all available input devices is given showing all device names and indices.

`-video_device_index <INDEX>`

Specify the video device by its index. Overrides anything given in the input filename.

`-audio_device_index <INDEX>`

Specify the audio device by its index. Overrides anything given in the input filename.

`-pixel_format <FORMAT>`

Request the video device to use a specific pixel format. If the specified format is not supported, a list of available formats is given and the first one in this list is used instead. Available pixel formats are: `monob`, `rgb555be`, `rgb555le`, `rgb565be`, `rgb565le`, `rgb24`, `bgr24`, `0rgb`, `bgr0`, `0bgr`, `rgb0`, `bgr48be`, `uyvy422`, `yuva444p`, `yuva444p16le`, `yuv444p`, `yuv422p16`, `yuv422p10`, `yuv444p10`, `yuv420p`, `nv12`, `yuyv422`, `gray`

`-framerate`

Set the grabbing frame rate. Default is `ntsc`, corresponding to a frame rate of 30000/1001.

`-video_size`

Set the video frame size.

`-capture_cursor`

Capture the mouse pointer. Default is 0.

`-capture_mouse_clicks`

Capture the screen mouse clicks. Default is 0.

## 26.2.2 Examples# TOC

- Print the list of AVFoundation supported devices and exit:

```
$ ffmpeg -f avfoundation -list_devices true -i ""
```

- Record video from video device 0 and audio from audio device 0 into out.avi:

```
$ ffmpeg -f avfoundation -i "0:0" out.avi
```

- Record video from video device 2 and audio from audio device 1 into out.avi:

```
$ ffmpeg -f avfoundation -video_device_index 2 -i ":1" out.avi
```

- Record video from the system default video device using the pixel format bgr0 and do not record any audio into out.avi:

```
$ ffmpeg -f avfoundation -pixel_format bgr0 -i "default:none" out.avi
```

## 26.3 bktr# TOC

BSD video input device.

### 26.3.1 Options# TOC

`framerate`

Set the frame rate.

`video_size`

Set the video frame size. Default is vga.

`standard`

Available values are:

```
'pal'
'ntsc'
'secam'
'paln'
'palm'
'ntscj'
```

## 26.4 decklink# TOC

The decklink input device provides capture capabilities for Blackmagic DeckLink devices.

To enable this input device, you need the Blackmagic DeckLink SDK and you need to configure with the appropriate `--extra-cflags` and `--extra-ldflags`. On Windows, you need to run the IDL files through `widl`.

DeckLink is very picky about the formats it supports. Pixel format is `uyvy422` or `v210`, framerate and video size must be determined for your device with `-list_formats 1`. Audio sample rate is always 48 kHz and the number of channels can be 2, 8 or 16.

### 26.4.1 Options# TOC

`list_devices`

If set to `true`, print a list of devices and exit. Defaults to `false`.

`list_formats`

If set to `true`, print a list of supported formats and exit. Defaults to `false`.

`bm_v210`

If set to `'1'`, video is captured in 10 bit `v210` instead of `uyvy422`. Not all Blackmagic devices support this option.

### 26.4.2 Examples# TOC

- List input devices:

```
ffmpeg -f decklink -list_devices 1 -i dummy
```

- List supported formats:

```
ffmpeg -f decklink -list_formats 1 -i 'Intensity Pro'
```

- Capture video clip at 1080i50 (format 11):

```
ffmpeg -f decklink -i 'Intensity Pro@11' -acodec copy -vcodec copy output.avi
```

- Capture video clip at 1080i50 10 bit:



```
ffmpeg -bm_v210 1 -f decklink -i 'UltraStudio Mini Recorder@11' -acodec copy -vcodec copy output.avi
```

## 26.5 dshow# TOC

Windows DirectShow input device.

DirectShow support is enabled when FFmpeg is built with the mingw-w64 project. Currently only audio and video devices are supported.

Multiple devices may be opened as separate inputs, but they may also be opened on the same input, which should improve synchronism between them.

The input name should be in the format:

```
TYPE=NAME[ :TYPE=NAME]
```

where *TYPE* can be either *audio* or *video*, and *NAME* is the device's name or alternative name..

### 26.5.1 Options# TOC

If no options are specified, the device's defaults are used. If the device does not support the requested options, it will fail to open.

`video_size`

Set the video size in the captured video.

`framerate`

Set the frame rate in the captured video.

`sample_rate`

Set the sample rate (in Hz) of the captured audio.

`sample_size`

Set the sample size (in bits) of the captured audio.

`channels`

Set the number of channels in the captured audio.

`list_devices`

If set to true, print a list of devices and exit.

`list_options`

If set to true, print a list of selected device's options and exit.

video\_device\_number

Set video device number for devices with the same name (starts at 0, defaults to 0).

audio\_device\_number

Set audio device number for devices with the same name (starts at 0, defaults to 0).

pixel\_format

Select pixel format to be used by DirectShow. This may only be set when the video codec is not set or set to rawvideo.

audio\_buffer\_size

Set audio device buffer size in milliseconds (which can directly impact latency, depending on the device). Defaults to using the audio device's default buffer size (typically some multiple of 500ms).

Setting this value too low can degrade performance. See also

[http://msdn.microsoft.com/en-us/library/windows/desktop/dd377582\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/dd377582(v=vs.85).aspx)

video\_pin\_name

Select video capture pin to use by name or alternative name.

audio\_pin\_name

Select audio capture pin to use by name or alternative name.

crossbar\_video\_input\_pin\_number

Select video input pin number for crossbar device. This will be routed to the crossbar device's Video Decoder output pin. Note that changing this value can affect future invocations (sets a new default) until system reboot occurs.

crossbar\_audio\_input\_pin\_number

Select audio input pin number for crossbar device. This will be routed to the crossbar device's Audio Decoder output pin. Note that changing this value can affect future invocations (sets a new default) until system reboot occurs.

show\_video\_device\_dialog

If set to true, before capture starts, popup a display dialog to the end user, allowing them to change video filter properties and configurations manually. Note that for crossbar devices, adjusting values in this dialog may be needed at times to toggle between PAL (25 fps) and NTSC (29.97) input frame rates, sizes, interlacing, etc. Changing these values can enable different scan rates/frame rates and avoiding green bars at the bottom, flickering scan lines, etc. Note that with some devices, changing these properties can also affect future invocations (sets new defaults) until system reboot occurs.

`show_audio_device_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to change audio filter properties and configurations manually.

`show_video_crossbar_connection_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to manually modify crossbar pin routings, when it opens a video device.

`show_audio_crossbar_connection_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to manually modify crossbar pin routings, when it opens an audio device.

`show_analog_tv_tuner_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to manually modify TV channels and frequencies.

`show_analog_tv_tuner_audio_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to manually modify TV audio (like mono vs. stereo, Language A,B or C).

`audio_device_load`

Load an audio capture filter device from file instead of searching it by name. It may load additional parameters too, if the filter supports the serialization of its properties to. To use this an audio capture source has to be specified, but it can be anything even fake one.

`audio_device_save`

Save the currently used audio capture filter device and its parameters (if the filter supports it) to a file. If a file with the same name exists it will be overwritten.

`video_device_load`

Load a video capture filter device from file instead of searching it by name. It may load additional parameters too, if the filter supports the serialization of its properties to. To use this a video capture source has to be specified, but it can be anything even fake one.

`video_device_save`

Save the currently used video capture filter device and its parameters (if the filter supports it) to a file. If a file with the same name exists it will be overwritten.

## 26.5.2 Examples# TOC

- Print the list of DirectShow supported devices and exit:

```
$ ffmpeg -list_devices true -f dshow -i dummy
```

- Open video device *Camera*:

```
$ ffmpeg -f dshow -i video="Camera"
```

- Open second video device with name *Camera*:

```
$ ffmpeg -f dshow -video_device_number 1 -i video="Camera"
```

- Open video device *Camera* and audio device *Microphone*:

```
$ ffmpeg -f dshow -i video="Camera":audio="Microphone"
```

- Print the list of supported options in selected device and exit:

```
$ ffmpeg -list_options true -f dshow -i video="Camera"
```

- Specify pin names to capture by name or alternative name, specify alternative device name:

```
$ ffmpeg -f dshow -audio_pin_name "Audio Out" -video_pin_name 2 -i video=video="device_gmp_\\?pci#ven_1a0a&dev_62021461&rev_0184&e2c7d856&a0a0a18[65e8773d-8f56-11d0-a3b9-00a0c9223196]\\{ca465100-dab0-4d59-818f-8c477284adf6}" :audio="Microphone"
```

- Configure a crossbar device, specifying crossbar pins, allow user to adjust video capture properties at startup:

```
$ ffmpeg -f dshow -show_video_device_dialog true -crossbar_video_input_pin_number 0  
-crossbar_audio_input_pin_number 3 -i video="AVerMedia BDA Analog Capture":audio="AVerMedia BDA Analog Capture"
```

## 26.6 dv1394# TOC

Linux DV 1394 input device.

### 26.6.1 Options# TOC

framerate

Set the frame rate. Default is 25.

standard

Available values are:

‘pal’  
‘ntsc’

Default value is ntsc.

## 26.7 fbdev# TOC

Linux framebuffer input device.

The Linux framebuffer is a graphic hardware-independent abstraction layer to show graphics on a computer monitor, typically on the console. It is accessed through a file device node, usually `/dev/fb0`.

For more detailed information read the file `Documentation/fb/framebuffer.txt` included in the Linux source tree.

See also <http://linux-fbdev.sourceforge.net/>, and `fbset(1)`.

To record from the framebuffer device `/dev/fb0` with `ffmpeg`:

```
ffmpeg -f fbdev -framerate 10 -i /dev/fb0 out.avi
```

You can take a single screenshot image with the command:

```
ffmpeg -f fbdev -framerate 1 -i /dev/fb0 -frames:v 1 screenshot.jpeg
```

### 26.7.1 Options# TOC

`framerate`

Set the frame rate. Default is 25.

## 26.8 gdigrab# TOC

Win32 GDI-based screen capture device.

This device allows you to capture a region of the display on Windows.

There are two options for the input filename:

`desktop`

or

`title=window_title`

The first option will capture the entire desktop, or a fixed region of the desktop. The second option will instead capture the contents of a single window, regardless of its position on the screen.

For example, to grab the entire desktop using `ffmpeg`:

```
ffmpeg -f gdigrab -framerate 6 -i desktop out.mpg
```

Grab a 640x480 region at position 10 , 20:

```
ffmpeg -f gdigrab -framerate 6 -offset_x 10 -offset_y 20 -video_size vga -i desktop out.mpg
```

Grab the contents of the window named "Calculator"

```
ffmpeg -f gdigrab -framerate 6 -i title=Calculator out.mpg
```

## 26.8.1 Options# TOC

`draw_mouse`

Specify whether to draw the mouse pointer. Use the value 0 to not draw the pointer. Default value is 1.

`framerate`

Set the grabbing frame rate. Default value is `ntsc`, corresponding to a frame rate of  $30000/1001$ .

`show_region`

Show grabbed region on screen.

If `show_region` is specified with 1, then the grabbing region will be indicated on screen. With this option, it is easy to know what is being grabbed if only a portion of the screen is grabbed.

Note that `show_region` is incompatible with grabbing the contents of a single window.

For example:

```
ffmpeg -f gdigrab -show_region 1 -framerate 6 -video_size cif -offset_x 10 -offset_y 20 -i desktop out.mpg
```

`video_size`

Set the video frame size. The default is to capture the full screen if `desktop` is selected, or the full window size if `title=window_title` is selected.

`offset_x`

When capturing a region with `video_size`, set the distance from the left edge of the screen or desktop.

Note that the offset calculation is from the top left corner of the primary monitor on Windows. If you have a monitor positioned to the left of your primary monitor, you will need to use a negative `offset_x` value to move the region to that monitor.

`offset_y`

When capturing a region with `video_size`, set the distance from the top edge of the screen or desktop.

Note that the offset calculation is from the top left corner of the primary monitor on Windows. If you have a monitor positioned above your primary monitor, you will need to use a negative `offset_y` value to move the region to that monitor.

## 26.9 iec61883# TOC

FireWire DV/HDV input device using libiec61883.

To enable this input device, you need libiec61883, libraw1394 and libavc1394 installed on your system. Use the configure option `--enable-libiec61883` to compile with the device enabled.

The iec61883 capture device supports capturing from a video device connected via IEEE1394 (FireWire), using libiec61883 and the new Linux FireWire stack (juju). This is the default DV/HDV input method in Linux Kernel 2.6.37 and later, since the old FireWire stack was removed.

Specify the FireWire port to be used as input file, or "auto" to choose the first port connected.

### 26.9.1 Options# TOC

`dvtype`

Override autodetection of DV/HDV. This should only be used if auto detection does not work, or if usage of a different device type should be prohibited. Treating a DV device as HDV (or vice versa) will not work and result in undefined behavior. The values `auto`, `dv` and `hdv` are supported.

`dvbuffer`

Set maximum size of buffer for incoming data, in frames. For DV, this is an exact value. For HDV, it is not frame exact, since HDV does not have a fixed frame size.

`dvguid`

Select the capture device by specifying its GUID. Capturing will only be performed from the specified device and fails if no device with the given GUID is found. This is useful to select the input if multiple devices are connected at the same time. Look at `/sys/bus/firewire/devices` to find out the GUIDs.

### 26.9.2 Examples# TOC

- Grab and show the input of a FireWire DV/HDV device.

```
ffplay -f iec61883 -i auto
```

- Grab and record the input of a FireWire DV/HDV device, using a packet buffer of 100000 packets if the source is HDV.

```
ffmpeg -f iec61883 -i auto -hdvbuffer 100000 out.mpg
```

## 26.10 jack# TOC

JACK input device.

To enable this input device during configuration you need libjack installed on your system.

A JACK input device creates one or more JACK writable clients, one for each audio channel, with name *client\_name*:input\_*N*, where *client\_name* is the name provided by the application, and *N* is a number which identifies the channel. Each writable client will send the acquired data to the FFmpeg input device.

Once you have created one or more JACK readable clients, you need to connect them to one or more JACK writable clients.

To connect or disconnect JACK clients you can use the `jack_connect` and `jack_disconnect` programs, or do it through a graphical interface, for example with `qjackctl`.

To list the JACK clients and their properties you can invoke the command `jack_lsp`.

Follows an example which shows how to capture a JACK readable client with `ffmpeg`.

```
# Create a JACK writable client with name "ffmpeg".
$ ffmpeg -f jack -i ffmpeg -y out.wav

# Start the sample jack_metro readable client.
$ jack_metro -b 120 -d 0.2 -f 4000

# List the current JACK clients.
$ jack_lsp -c
system:capture_1
system:capture_2
system:playback_1
system:playback_2
ffmpeg:input_1
metro:120_bpm

# Connect metro to the ffmpeg writable client.
$ jack_connect metro:120_bpm ffmpeg:input_1
```

For more information read: <http://jackaudio.org/>

### 26.10.1 Options# TOC

channels

Set the number of channels. Default is 2.



## 26.11 lavfi# TOC

Libavfilter input virtual device.

This input device reads data from the open output pads of a libavfilter filtergraph.

For each filtergraph open output, the input device will create a corresponding stream which is mapped to the generated output. Currently only video data is supported. The filtergraph is specified through the option *graph*.

### 26.11.1 Options# TOC

*graph*

Specify the filtergraph to use as input. Each video open output must be labelled by a unique string of the form "outN", where *N* is a number starting from 0 corresponding to the mapped input stream generated by the device. The first unlabelled output is automatically assigned to the "out0" label, but all the others need to be specified explicitly.

The suffix "+subcc" can be appended to the output label to create an extra stream with the closed captions packets attached to that output (experimental; only for EIA-608 / CEA-708 for now). The subcc streams are created after all the normal streams, in the order of the corresponding stream. For example, if there is "out19+subcc", "out7+subcc" and up to "out42", the stream #43 is subcc for stream #7 and stream #44 is subcc for stream #19.

If not specified defaults to the filename specified for the input device.

*graph\_file*

Set the filename of the filtergraph to be read and sent to the other filters. Syntax of the filtergraph is the same as the one specified by the option *graph*.

*dumpgraph*

Dump graph to stderr.

### 26.11.2 Examples# TOC

- Create a color video stream and play it back with `ffplay`:

```
ffplay -f lavfi -graph "color=c=pink [out0]" dummy
```

- As the previous example, but use filename for specifying the graph description, and omit the "out0" label:

```
ffplay -f lavfi color=c=pink
```

- Create three different video test filtered sources and play them:

```
ffplay -f lavfi -graph "testsrc [out0]; testsrc,hflip [out1]; testsrc,negate [out2]" test3
```

- Read an audio stream from a file using the amovie source and play it back with ffplay:

```
ffplay -f lavfi "amovie=test.wav"
```

- Read an audio stream and a video stream and play it back with ffplay:

```
ffplay -f lavfi "movie=test.avi[out0];amovie=test.wav[out1]"
```

- Dump decoded frames to images and closed captions to a file (experimental):

```
ffmpeg -f lavfi -i "movie=test.ts[out0+subcc]" -map v frame%08d.png -map s -c copy -f rawvideo subcc.bin
```

## 26.12 libcdio# TOC

Audio-CD input device based on libcdio.

To enable this input device during configuration you need libcdio installed on your system. It requires the configure option `--enable-libcdio`.

This device allows playing and grabbing from an Audio-CD.

For example to copy with `ffmpeg` the entire Audio-CD in `/dev/sr0`, you may run the command:

```
ffmpeg -f libcdio -i /dev/sr0 cd.wav
```

### 26.12.1 Options# TOC

`speed`

Set drive reading speed. Default value is 0.

The speed is specified CD-ROM speed units. The speed is set through the libcdio `cdio_cddap_speed_set` function. On many CD-ROM drives, specifying a value too large will result in using the fastest speed.

`paranoia_mode`

Set paranoia recovery mode flags. It accepts one of the following values:

```
'disable'
'verify'
'overlap'
'neverskip'
'full'
```

Default value is `'disable'`.

For more information about the available recovery modes, consult the paranoia project documentation.

## 26.13 libdc1394# TOC

IIDC1394 input device, based on libdc1394 and libraw1394.

Requires the configure option `--enable-libdc1394`.

## 26.14 openal# TOC

The OpenAL input device provides audio capture on all systems with a working OpenAL 1.1 implementation.

To enable this input device during configuration, you need OpenAL headers and libraries installed on your system, and need to configure FFmpeg with `--enable-openal`.

OpenAL headers and libraries should be provided as part of your OpenAL implementation, or as an additional download (an SDK). Depending on your installation you may need to specify additional flags via the `--extra-cflags` and `--extra-ldflags` for allowing the build system to locate the OpenAL headers and libraries.

An incomplete list of OpenAL implementations follows:

### Creative

The official Windows implementation, providing hardware acceleration with supported devices and software fallback. See <http://openal.org/>.

### OpenAL Soft

Portable, open source (LGPL) software implementation. Includes backends for the most common sound APIs on the Windows, Linux, Solaris, and BSD operating systems. See <http://kcat.strangesoft.net/openal.html>.

### Apple

OpenAL is part of Core Audio, the official Mac OS X Audio interface. See <http://developer.apple.com/technologies/mac/audio-and-video.html>

This device allows one to capture from an audio input device handled through OpenAL.

You need to specify the name of the device to capture in the provided filename. If the empty string is provided, the device will automatically select the default device. You can get the list of the supported devices by using the option `list_devices`.

## 26.14.1 Options# TOC

`channels`

Set the number of channels in the captured audio. Only the values 1 (monaural) and 2 (stereo) are currently supported. Defaults to 2.

`sample_size`

Set the sample size (in bits) of the captured audio. Only the values 8 and 16 are currently supported. Defaults to 16.

`sample_rate`

Set the sample rate (in Hz) of the captured audio. Defaults to 44.1k.

`list_devices`

If set to true, print a list of devices and exit. Defaults to false.

## 26.14.2 Examples# TOC

Print the list of OpenAL supported devices and exit:

```
$ ffmpeg -list_devices true -f openal -i dummy out.ogg
```

Capture from the OpenAL device DR-BT101 via PulseAudio:

```
$ ffmpeg -f openal -i 'DR-BT101 via PulseAudio' out.ogg
```

Capture from the default device (note the empty string "" as filename):

```
$ ffmpeg -f openal -i '' out.ogg
```

Capture from two devices simultaneously, writing to two different files, within the same ffmpeg command:

```
$ ffmpeg -f openal -i 'DR-BT101 via PulseAudio' out1.ogg -f openal -i 'ALSA Default' out2.ogg
```

Note: not all OpenAL implementations support multiple simultaneous capture - try the latest OpenAL Soft if the above does not work.

## 26.15 oss# TOC

Open Sound System input device.

The filename to provide to the input device is the device node representing the OSS input device, and is usually set to `/dev/dsp`.

For example to grab from `/dev/dsp` using `ffmpeg` use the command:

```
ffmpeg -f oss -i /dev/dsp /tmp/oss.wav
```

For more information about OSS see: <http://manuals.opensound.com/usersguide/dsp.html>

### **26.15.1 Options# TOC**

`sample_rate`

Set the sample rate in Hz. Default is 48000.

`channels`

Set the number of channels. Default is 2.

### **26.16 pulse# TOC**

PulseAudio input device.

To enable this output device you need to configure FFmpeg with `--enable-libpulse`.

The filename to provide to the input device is a source device or the string "default"

To list the PulseAudio source devices and their properties you can invoke the command `pactl list sources`.

More information about PulseAudio can be found on <http://www.pulseaudio.org>.

### **26.16.1 Options# TOC**

`server`

Connect to a specific PulseAudio server, specified by an IP address. Default server is used when not provided.

`name`

Specify the application name PulseAudio will use when showing active clients, by default it is the `LIBAVFORMAT_IDENT` string.

`stream_name`

Specify the stream name PulseAudio will use when showing active streams, by default it is "record".

`sample_rate`

Specify the samplerate in Hz, by default 48kHz is used.

channels

Specify the channels in use, by default 2 (stereo) is set.

frame\_size

Specify the number of bytes per frame, by default it is set to 1024.

fragment\_size

Specify the minimal buffering fragment in PulseAudio, it will affect the audio latency. By default it is unset.

wallclock

Set the initial PTS using the current time. Default is 1.

## 26.16.2 Examples# TOC

Record a stream from default device:

```
ffmpeg -f pulse -i default /tmp/pulse.wav
```

## 26.17 qtkit# TOC

QTKit input device.

The filename passed as input is parsed to contain either a device name or index. The device index can also be given by using -video\_device\_index. A given device index will override any given device name. If the desired device consists of numbers only, use -video\_device\_index to identify it. The default device will be chosen if an empty string or the device name "default" is given. The available devices can be enumerated by using -list\_devices.

```
ffmpeg -f qtkit -i "0" out.mpg
```

```
ffmpeg -f qtkit -video_device_index 0 -i "" out.mpg
```

```
ffmpeg -f qtkit -i "default" out.mpg
```

```
ffmpeg -f qtkit -list_devices true -i ""
```

### 26.17.1 Options# TOC

frame\_rate

Set frame rate. Default is 30.

`list_devices`

If set to `true`, print a list of devices and exit. Default is `false`.

`video_device_index`

Select the video device by index for devices with the same name (starts at 0).

## 26.18 sndio# TOC

sndio input device.

To enable this input device during configuration you need `libsndio` installed on your system.

The filename to provide to the input device is the device node representing the sndio input device, and is usually set to `/dev/audio0`.

For example to grab from `/dev/audio0` using `ffmpeg` use the command:

```
ffmpeg -f sndio -i /dev/audio0 /tmp/oss.wav
```

### 26.18.1 Options# TOC

`sample_rate`

Set the sample rate in Hz. Default is 48000.

`channels`

Set the number of channels. Default is 2.

## 26.19 video4linux2, v4l2# TOC

Video4Linux2 input video device.

"v4l2" can be used as alias for "video4linux2".

If `FFmpeg` is built with `v4l-utils` support (by using the `--enable-libv4l2` configure option), it is possible to use it with the `-use_libv4l2` input device option.

The name of the device to grab is a file device node, usually Linux systems tend to automatically create such nodes when the device (e.g. an USB webcam) is plugged into the system, and has a name of the kind `/dev/videoN`, where *N* is a number associated to the device.

Video4Linux2 devices usually support a limited set of *widthxheight* sizes and frame rates. You can check which are supported using `-list_formats all` for Video4Linux2 devices. Some devices, like TV cards, support one or more standards. It is possible to list all the supported standards using `-list_standards all`.

The time base for the timestamps is 1 microsecond. Depending on the kernel version and configuration, the timestamps may be derived from the real time clock (origin at the Unix Epoch) or the monotonic clock (origin usually at boot time, unaffected by NTP or manual changes to the clock). The `-timestamps abs` or `-ts abs` option can be used to force conversion into the real time clock.

Some usage examples of the video4linux2 device with `ffmpeg` and `ffplay`:

- List supported formats for a video4linux2 device:

```
ffplay -f video4linux2 -list_formats all /dev/video0
```

- Grab and show the input of a video4linux2 device:

```
ffplay -f video4linux2 -framerate 30 -video_size hd720 /dev/video0
```

- Grab and record the input of a video4linux2 device, leave the frame rate and size as previously set:

```
ffmpeg -f video4linux2 -input_format mjpeg -i /dev/video0 out.mpeg
```

For more information about Video4Linux, check <http://linuxtv.org/>.

### 26.19.1 Options# TOC

`standard`

Set the standard. Must be the name of a supported standard. To get a list of the supported standards, use the `list_standards` option.

`channel`

Set the input channel number. Default to -1, which means using the previously selected channel.

`video_size`

Set the video frame size. The argument must be a string in the form *WIDTHxHEIGHT* or a valid size abbreviation.

`pixel_format`

Select the pixel format (only valid for raw video input).

`input_format`

Set the preferred pixel format (for raw video) or a codec name. This option allows one to select the input format, when several are available.

`framerate`



Set the preferred video frame rate.

`list_formats`

List available formats (supported pixel formats, codecs, and frame sizes) and exit.

Available values are:

‘all’

Show all available (compressed and non-compressed) formats.

‘raw’

Show only raw video (non-compressed) formats.

‘compressed’

Show only compressed formats.

`list_standards`

List supported standards and exit.

Available values are:

‘all’

Show all supported standards.

`timestamps, ts`

Set type of timestamps for grabbed frames.

Available values are:

‘default’

Use timestamps from the kernel.

‘abs’

Use absolute timestamps (wall clock).

‘mono2abs’

Force conversion from monotonic to absolute timestamps.

Default value is `default`.

`use_libv4l2`

Use libv4l2 (v4l-utils) conversion functions. Default is 0.

## 26.20 vfwcap# TOC

VfW (Video for Windows) capture input device.

The filename passed as input is the capture driver number, ranging from 0 to 9. You may use "list" as filename to print a list of drivers. Any other filename will be interpreted as device number 0.

### 26.20.1 Options# TOC

`video_size`

Set the video frame size.

`framerate`

Set the grabbing frame rate. Default value is `ntsc`, corresponding to a frame rate of 30000/1001.

## 26.21 x11grab# TOC

X11 video input device.

To enable this input device during configuration you need libxcb installed on your system. It will be automatically detected during configuration.

Alternatively, the configure option `--enable-x11grab` exists for legacy Xlib users.

This device allows one to capture a region of an X11 display.

The filename passed as input has the syntax:

```
[hostname]:display_number.screen_number[+x_offset,y_offset]
```

*hostname:display\_number.screen\_number* specifies the X11 display name of the screen to grab from. *hostname* can be omitted, and defaults to "localhost". The environment variable `DISPLAY` contains the default display name.

*x\_offset* and *y\_offset* specify the offsets of the grabbed area with respect to the top-left border of the X11 screen. They default to 0.

Check the X11 documentation (e.g. `man X`) for more detailed information.

Use the `xdpinfo` program for getting basic information about the properties of your X11 display (e.g. `grep` for "name" or "dimensions").

For example to grab from `:0.0` using `ffmpeg`:

```
ffmpeg -f x11grab -framerate 25 -video_size cif -i :0.0 out.mpg
```

Grab at position 10,20:

```
ffmpeg -f x11grab -framerate 25 -video_size cif -i :0.0+10,20 out.mpg
```

## 26.21.1 Options# TOC

`draw_mouse`

Specify whether to draw the mouse pointer. A value of 0 specify not to draw the pointer. Default value is 1.

`follow_mouse`

Make the grabbed area follow the mouse. The argument can be `centered` or a number of pixels *PIXELS*.

When it is specified with "centered", the grabbing region follows the mouse pointer and keeps the pointer at the center of region; otherwise, the region follows only when the mouse pointer reaches within *PIXELS* (greater than zero) to the edge of region.

For example:

```
ffmpeg -f x11grab -follow_mouse centered -framerate 25 -video_size cif -i :0.0 out.mpg
```

To follow only when the mouse pointer reaches within 100 pixels to edge:

```
ffmpeg -f x11grab -follow_mouse 100 -framerate 25 -video_size cif -i :0.0 out.mpg
```

`framerate`

Set the grabbing frame rate. Default value is `ntsc`, corresponding to a frame rate of `30000/1001`.

`show_region`

Show grabbed region on screen.

If `show_region` is specified with 1, then the grabbing region will be indicated on screen. With this option, it is easy to know what is being grabbed if only a portion of the screen is grabbed.

`region_border`

Set the region border thickness if `-show_region 1` is used. Range is 1 to 128 and default is 3 (XCB-based x11grab only).

For example:

```
ffmpeg -f x11grab -show_region 1 -framerate 25 -video_size cif -i :0.0+10,20 out.mpg
```

With *follow\_mouse*:

```
ffmpeg -f x11grab -follow_mouse centered -show_region 1 -framerate 25 -video_size cif -i :0.0 out.mpg
```

`video_size`

Set the video frame size. Default value is vga.

`use_shm`

Use the MIT-SHM extension for shared memory. Default value is 1. It may be necessary to disable it for remote displays (legacy x11grab only).

### 26.21.2 *grab\_x grab\_y* AVOption# TOC

The syntax is:

```
-grab_x x_offset -grab_y y_offset
```

Set the grabbing region coordinates. They are expressed as offset from the top left corner of the X11 window. The default value is 0.

## 27 Output Devices# TOC

Output devices are configured elements in FFmpeg that can write multimedia data to an output device attached to your system.

When you configure your FFmpeg build, all the supported output devices are enabled by default. You can list all available ones using the configure option "`--list-outdevs`".

You can disable all the output devices using the configure option "`--disable-outdevs`", and selectively enable an output device using the option "`--enable-outdev=OUTDEV`", or you can disable a particular input device using the option "`--disable-outdev=OUTDEV`".

The option "`-devices`" of the ff\* tools will display the list of enabled output devices.

A description of the currently available output devices follows.

## 27.1 alsa# TOC

ALSA (Advanced Linux Sound Architecture) output device.

### 27.1.1 Examples# TOC

- Play a file on default ALSA device:

```
ffmpeg -i INPUT -f alsa default
```

- Play a file on soundcard 1, audio device 7:

```
ffmpeg -i INPUT -f alsa hw:1,7
```

## 27.2 caca# TOC

CACA output device.

This output device allows one to show a video stream in CACA window. Only one CACA window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need to configure FFmpeg with `--enable-libcaca`. libcaca is a graphics library that outputs text instead of pixels.

For more information about libcaca, check: <http://caca.zoy.org/wiki/libcaca>

### 27.2.1 Options# TOC

`window_title`

Set the CACA window title, if not specified default to the filename specified for the output device.

`window_size`

Set the CACA window size, can be a string of the form *widthxheight* or a video size abbreviation. If not specified it defaults to the size of the input video.

`driver`

Set display driver.

`algorithm`

Set dithering algorithm. Dithering is necessary because the picture being rendered has usually far more colours than the available palette. The accepted values are listed with `-list_dither_algorithms`.

antialias

Set antialias method. Antialiasing smoothens the rendered image and avoids the commonly seen staircase effect. The accepted values are listed with `-list_dither antialiases`.

charset

Set which characters are going to be used when rendering text. The accepted values are listed with `-list_dither charsets`.

color

Set color to be used when rendering text. The accepted values are listed with `-list_dither colors`.

list\_drivers

If set to true, print a list of available drivers and exit.

list\_dither

List available dither options related to the argument. The argument must be one of algorithms, antialiases, charsets, colors.

### 27.2.2 Examples# TOC

- The following command shows the ffmpeg output is an CACA window, forcing its size to 80x25:

```
ffmpeg -i INPUT -vcodec rawvideo -pix_fmt rgb24 -window_size 80x25 -f caca -
```

- Show the list of available drivers and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_drivers true -
```

- Show the list of available dither colors and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_dither colors -
```

### 27.3 decklink# TOC

The decklink output device provides playback capabilities for Blackmagic DeckLink devices.

To enable this output device, you need the Blackmagic DeckLink SDK and you need to configure with the appropriate `--extra-cflags` and `--extra-ldflags`. On Windows, you need to run the IDL files through `widl`.

DeckLink is very picky about the formats it supports. Pixel format is always uyvy422, framerate and video size must be determined for your device with `-list_formats 1`. Audio sample rate is always 48 kHz.

### 27.3.1 Options# TOC

`list_devices`

If set to `true`, print a list of devices and exit. Defaults to `false`.

`list_formats`

If set to `true`, print a list of supported formats and exit. Defaults to `false`.

`preroll`

Amount of time to preroll video in seconds. Defaults to 0.5.

### 27.3.2 Examples# TOC

- List output devices:

```
ffmpeg -i test.avi -f decklink -list_devices 1 dummy
```

- List supported formats:

```
ffmpeg -i test.avi -f decklink -list_formats 1 'DeckLink Mini Monitor'
```

- Play video clip:

```
ffmpeg -i test.avi -f decklink -pix_fmt uyvy422 'DeckLink Mini Monitor'
```

- Play video clip with non-standard framerate or video size:

```
ffmpeg -i test.avi -f decklink -pix_fmt uyvy422 -s 720x486 -r 24000/1001 'DeckLink Mini Monitor'
```

## 27.4 fbdev# TOC

Linux framebuffer output device.

The Linux framebuffer is a graphic hardware-independent abstraction layer to show graphics on a computer monitor, typically on the console. It is accessed through a file device node, usually `/dev/fb0`.

For more detailed information read the file `Documentation/fb/framebuffer.txt` included in the Linux source tree.

### 27.4.1 Options# TOC

`xoffset`

`yoffset`

Set x/y coordinate of top left corner. Default is 0.

## 27.4.2 Examples# TOC

Play a file on framebuffer device `/dev/fb0`. Required pixel format depends on current framebuffer settings.

```
ffmpeg -re -i INPUT -vcodec rawvideo -pix_fmt bgra -f fbdev /dev/fb0
```

See also <http://linux-fbdev.sourceforge.net/>, and `fbset(1)`.

## 27.5 opengl# TOC

OpenGL output device.

To enable this output device you need to configure FFmpeg with `--enable-opengl`.

This output device allows one to render to OpenGL context. Context may be provided by application or default SDL window is created.

When device renders to external context, application must implement handlers for following messages:

AV\_DEV\_TO\_APP\_CREATE\_WINDOW\_BUFFER - create OpenGL context on current thread.

AV\_DEV\_TO\_APP\_PREPARE\_WINDOW\_BUFFER - make OpenGL context current.

AV\_DEV\_TO\_APP\_DISPLAY\_WINDOW\_BUFFER - swap buffers.

AV\_DEV\_TO\_APP\_DESTROY\_WINDOW\_BUFFER - destroy OpenGL context. Application is also required to inform a device about current resolution by sending AV\_APP\_TO\_DEV\_WINDOW\_SIZE message.

### 27.5.1 Options# TOC

`background`

Set background color. Black is a default.

`no_window`

Disables default SDL window when set to non-zero value. Application must provide OpenGL context and both `window_size_cb` and `window_swap_buffers_cb` callbacks when set.

`window_title`

Set the SDL window title, if not specified default to the filename specified for the output device. Ignored when `no_window` is set.

`window_size`

Set preferred window size, can be a string of the form `widthxheight` or a video size abbreviation. If not specified it defaults to the size of the input video, downscaled according to the aspect ratio. Mostly usable when `no_window` is not set.



## 27.5.2 Examples# TOC

Play a file on SDL window using OpenGL rendering:

```
ffmpeg -i INPUT -f opengl "window title"
```

## 27.6 oss# TOC

OSS (Open Sound System) output device.

## 27.7 pulse# TOC

PulseAudio output device.

To enable this output device you need to configure FFmpeg with `--enable-libpulse`.

More information about PulseAudio can be found on <http://www.pulseaudio.org>

### 27.7.1 Options# TOC

`server`

Connect to a specific PulseAudio server, specified by an IP address. Default server is used when not provided.

`name`

Specify the application name PulseAudio will use when showing active clients, by default it is the `LIBAVFORMAT_IDENT` string.

`stream_name`

Specify the stream name PulseAudio will use when showing active streams, by default it is set to the specified output name.

`device`

Specify the device to use. Default device is used when not provided. List of output devices can be obtained with command `pactl list sinks`.

`buffer_size`

`buffer_duration`

Control the size and duration of the PulseAudio buffer. A small buffer gives more control, but requires more frequent updates.

`buffer_size` specifies size in bytes while `buffer_duration` specifies duration in milliseconds.

When both options are provided then the highest value is used (duration is recalculated to bytes using stream parameters). If they are set to 0 (which is default), the device will use the default PulseAudio duration value. By default PulseAudio set buffer duration to around 2 seconds.

`prebuf`

Specify pre-buffering size in bytes. The server does not start with playback before at least `prebuf` bytes are available in the buffer. By default this option is initialized to the same value as `buffer_size` or `buffer_duration` (whichever is bigger).

`minreq`

Specify minimum request size in bytes. The server does not request less than `minreq` bytes from the client, instead waits until the buffer is free enough to request more bytes at once. It is recommended to not set this option, which will initialize this to a value that is deemed sensible by the server.

## 27.7.2 Examples# TOC

Play a file on default device on default server:

```
ffmpeg -i INPUT -f pulse "stream name"
```

## 27.8 sdl# TOC

SDL (Simple DirectMedia Layer) output device.

This output device allows one to show a video stream in an SDL window. Only one SDL window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need `libsdl` installed on your system when configuring your build.

For more information about SDL, check: <http://www.libsdl.org/>

### 27.8.1 Options# TOC

`window_title`

Set the SDL window title, if not specified default to the filename specified for the output device.

`icon_title`

Set the name of the iconified SDL window, if not specified it is set to the same value of *window\_title*.

`window_size`

Set the SDL window size, can be a string of the form *widthxheight* or a video size abbreviation. If not specified it defaults to the size of the input video, downscaled according to the aspect ratio.

`window_fullscreen`

Set fullscreen mode when non-zero value is provided. Default value is zero.

## 27.8.2 Interactive commands# TOC

The window created by the device can be controlled through the following interactive commands.

`q`, `ESC`

Quit the device immediately.

## 27.8.3 Examples# TOC

The following command shows the `ffmpeg` output is an SDL window, forcing its size to the `qcif` format:

```
ffmpeg -i INPUT -vcodec rawvideo -pix_fmt yuv420p -window_size qcif -f sdl "SDL output"
```

## 27.9 sndio# TOC

`sndio` audio output device.

## 27.10 xv# TOC

`XV` (XVideo) output device.

This output device allows one to show a video stream in a X Window System window.

### 27.10.1 Options# TOC

`display_name`

Specify the hardware display name, which determines the display and communications domain to be used.

The display name or `DISPLAY` environment variable can be a string in the format *hostname[:number[.screen\_number]]*.

*hostname* specifies the name of the host machine on which the display is physically attached. *number* specifies the number of the display server on that host machine. *screen\_number* specifies the screen to be used on that server.

If unspecified, it defaults to the value of the `DISPLAY` environment variable.

For example, `dual-headed:0.1` would specify screen 1 of display 0 on the machine named “dual-headed”.

Check the X11 specification for more detailed information about the display name format.

`window_id`

When set to non-zero value then device doesn't create new window, but uses existing one with provided *window\_id*. By default this options is set to zero and device creates its own window.

`window_size`

Set the created window size, can be a string of the form *widthxheight* or a video size abbreviation. If not specified it defaults to the size of the input video. Ignored when *window\_id* is set.

`window_x`

`window_y`

Set the X and Y window offsets for the created window. They are both set to 0 by default. The values may be ignored by the window manager. Ignored when *window\_id* is set.

`window_title`

Set the window title, if not specified default to the filename specified for the output device. Ignored when *window\_id* is set.

For more information about XVideo see <http://www.x.org/>.

## 27.10.2 Examples# TOC

- Decode, display and encode video input with `ffmpeg` at the same time:

```
ffmpeg -i INPUT OUTPUT -f xv display
```

- Decode and display the input video to multiple X11 windows:

```
ffmpeg -i INPUT -f xv normal -vf negate -f xv negated
```

## 28 Resampler Options# TOC

The audio resampler supports the following named options.

Options may be set by specifying *-option value* in the FFmpeg tools, *option=value* for the `aresample` filter, by setting the value explicitly in the `SwrContext` options or using the `libavutil/opt.h` API for programmatic use.

`ich, in_channel_count`

Set the number of input channels. Default value is 0. Setting this value is not mandatory if the corresponding channel layout `in_channel_layout` is set.

`och, out_channel_count`

Set the number of output channels. Default value is 0. Setting this value is not mandatory if the corresponding channel layout `out_channel_layout` is set.

`uch, used_channel_count`

Set the number of used input channels. Default value is 0. This option is only used for special remapping.

`isr, in_sample_rate`

Set the input sample rate. Default value is 0.

`osr, out_sample_rate`

Set the output sample rate. Default value is 0.

`isf, in_sample_fmt`

Specify the input sample format. It is set by default to `none`.

`osf, out_sample_fmt`

Specify the output sample format. It is set by default to `none`.

`tsf, internal_sample_fmt`

Set the internal sample format. Default value is `none`. This will automatically be chosen when it is not explicitly set.

`icl, in_channel_layout`

`ocl, out_channel_layout`

Set the input/output channel layout.

See (ffmpeg-utils)the Channel Layout section in the ffmpeg-utils(1) manual for the required syntax.

`clev, center_mix_level`

Set the center mix level. It is a value expressed in deciBel, and must be in the interval `[-32,32]`.

`slev, surround_mix_level`

Set the surround mix level. It is a value expressed in deciBel, and must be in the interval `[-32,32]`.

lfe\_mix\_level

Set LFE mix into non LFE level. It is used when there is a LFE input but no LFE output. It is a value expressed in deciBel, and must be in the interval [-32,32].

rmvol, rematrix\_volume

Set rematrix volume. Default value is 1.0.

rematrix\_maxval

Set maximum output value for rematrixing. This can be used to prevent clipping vs. preventing volume reduction. A value of 1.0 prevents clipping.

flags, swr\_flags

Set flags used by the converter. Default value is 0.

It supports the following individual flags:

res

force resampling, this flag forces resampling to be used even when the input and output sample rates match.

dither\_scale

Set the dither scale. Default value is 1.

dither\_method

Set dither method. Default value is 0.

Supported values:

'rectangular'

select rectangular dither

'triangular'

select triangular dither

'triangular\_hp'

select triangular dither with high pass

'lipshitz'

select lipshitz noise shaping dither

‘shibata’

select shibata noise shaping dither

‘low\_shibata’

select low shibata noise shaping dither

‘high\_shibata’

select high shibata noise shaping dither

‘f\_weighted’

select f-weighted noise shaping dither

‘modified\_e\_weighted’

select modified-e-weighted noise shaping dither

‘improved\_e\_weighted’

select improved-e-weighted noise shaping dither

resampler

Set resampling engine. Default value is swr.

Supported values:

‘swr’

select the native SW Resampler; filter options precision and cheby are not applicable in this case.

‘soxr’

select the SoX Resampler (where available); compensation, and filter options filter\_size, phase\_shift, filter\_type & kaiser\_beta, are not applicable in this case.

filter\_size

For swr only, set resampling filter size, default value is 32.

phase\_shift

For swr only, set resampling phase shift, default value is 10, and must be in the interval [0,30].

`linear_interp`

Use Linear Interpolation if set to 1, default value is 0.

`cutoff`

Set cutoff frequency (swr: 6dB point; soxr: 0dB point) ratio; must be a float value between 0 and 1. Default value is 0.97 with swr, and 0.91 with soxr (which, with a sample-rate of 44100, preserves the entire audio band to 20kHz).

`precision`

For soxr only, the precision in bits to which the resampled signal will be calculated. The default value of 20 (which, with suitable dithering, is appropriate for a destination bit-depth of 16) gives SoX's 'High Quality'; a value of 28 gives SoX's 'Very High Quality'.

`cheby`

For soxr only, selects passband rolloff none (Chebyshev) & higher-precision approximation for 'irrational' ratios. Default value is 0.

`async`

For swr only, simple 1 parameter audio sync to timestamps using stretching, squeezing, filling and trimming. Setting this to 1 will enable filling and trimming, larger values represent the maximum amount in samples that the data may be stretched or squeezed for each second. Default value is 0, thus no compensation is applied to make the samples match the audio timestamps.

`first_pts`

For swr only, assume the first pts should be this value. The time unit is 1 / sample rate. This allows for padding/trimming at the start of stream. By default, no assumption is made about the first frame's expected pts, so no padding or trimming is done. For example, this could be set to 0 to pad the beginning with silence if an audio stream starts after the video stream or to trim any samples with a negative pts due to encoder delay.

`min_comp`

For swr only, set the minimum difference between timestamps and audio data (in seconds) to trigger stretching/squeezing/filling or trimming of the data to make it match the timestamps. The default is that stretching/squeezing/filling and trimming is disabled (`min_comp = FLT_MAX`).

`min_hard_comp`

For swr only, set the minimum difference between timestamps and audio data (in seconds) to trigger adding/dropping samples to make it match the timestamps. This option effectively is a threshold to select between hard (trim/fill) and soft (squeeze/stretch) compensation. Note that all compensation is



by default disabled through `min_comp`. The default is 0.1.

`comp_duration`

For swr only, set duration (in seconds) over which data is stretched/squeezed to make it match the timestamps. Must be a non-negative double float value, default value is 1.0.

`max_soft_comp`

For swr only, set maximum factor by which data is stretched/squeezed to make it match the timestamps. Must be a non-negative double float value, default value is 0.

`matrix_encoding`

Select matrixed stereo encoding.

It accepts the following values:

‘none’

select none

‘dolby’

select Dolby

‘dplii’

select Dolby Pro Logic II

Default value is none.

`filter_type`

For swr only, select resampling filter type. This only affects resampling operations.

It accepts the following values:

‘cubic’

select cubic

‘blackman\_nuttall’

select Blackman Nuttall Windowed Sinc

‘kaiser’

select Kaiser Windowed Sinc

kaiser\_beta

For swr only, set Kaiser Window Beta value. Must be an integer in the interval [2,16], default value is 9.

output\_sample\_bits

For swr only, set number of used output sample bits for dithering. Must be an integer in the interval [0,64], default value is 0, which means it's not used.

## 29 Scaler Options# TOC

The video scaler supports the following named options.

Options may be set by specifying *-option value* in the FFmpeg tools. For programmatic use, they can be set explicitly in the SwsContext options or through the libavutil/opt.h API.

sws\_flags

Set the scaler flags. This is also used to set the scaling algorithm. Only a single algorithm should be selected.

It accepts the following values:

'fast\_bilinear'

Select fast bilinear scaling algorithm.

'bilinear'

Select bilinear scaling algorithm.

'bicubic'

Select bicubic scaling algorithm.

'experimental'

Select experimental scaling algorithm.

'neighbor'

Select nearest neighbor rescaling algorithm.

'area'

Select averaging area rescaling algorithm.

`'bicublin'`

Select bicubic scaling algorithm for the luma component, bilinear for chroma components.

`'gauss'`

Select Gaussian rescaling algorithm.

`'sinc'`

Select sinc rescaling algorithm.

`'lanczos'`

Select lanczos rescaling algorithm.

`'spline'`

Select natural bicubic spline rescaling algorithm.

`'print_info'`

Enable printing/debug logging.

`'accurate_rnd'`

Enable accurate rounding.

`'full_chroma_int'`

Enable full chroma interpolation.

`'full_chroma_inp'`

Select full chroma input.

`'bitexact'`

Enable bitexact output.

`srcw`

Set source width.

`srch`

Set source height.

`dstw`

Set destination width.

`dsth`

Set destination height.

`src_format`

Set source pixel format (must be expressed as an integer).

`dst_format`

Set destination pixel format (must be expressed as an integer).

`src_range`

Select source range.

`dst_range`

Select destination range.

`param0, param1`

Set scaling algorithm parameters. The specified values are specific of some scaling algorithms and ignored by others. The specified values are floating point number values.

`sws_dither`

Set the dithering algorithm. Accepts one of the following values. Default value is 'auto'.

'auto'

automatic choice

'none'

no dithering

'bayer'

bayer dither

'ed'

error diffusion dither

`'a_dither'`

arithmetic dither, based using addition

`'x_dither'`

arithmetic dither, based using xor (more random/less apparent patterning than `a_dither`).

`alphablend`

Set the alpha blending to use when the input has alpha but the output does not. Default value is `'none'`.

`'uniform_color'`

Blend onto a uniform background color

`'checkerboard'`

Blend onto a checkerboard

`'none'`

No blending

## 30 Filtering Introduction# TOC

Filtering in FFmpeg is enabled through the libavfilter library.

In libavfilter, a filter can have multiple inputs and multiple outputs. To illustrate the sorts of things that are possible, we consider the following filtergraph.

```

                    [main]
input --> split -----> overlay --> output
      |
      |[tmp]                [flip]|
      +-----> crop --> vflip -----+
```

This filtergraph splits the input stream in two streams, then sends one stream through the crop filter and the vflip filter, before merging it back with the other stream by overlaying it on top. You can use the following command to achieve this:

```
ffmpeg -i INPUT -vf "split [main][tmp]; [tmp] crop=iw:ih/2:0:0, vflip [flip]; [main][flip] overlay=0:H/2" OUTPUT
```

The result will be that the top half of the video is mirrored onto the bottom half of the output video.

Filters in the same linear chain are separated by commas, and distinct linear chains of filters are separated by semicolons. In our example, *crop,vflip* are in one linear chain, *split* and *overlay* are separately in another. The points where the linear chains join are labelled by names enclosed in square brackets. In the example, the split filter generates two outputs that are associated to the labels *[main]* and *[tmp]*.

The stream sent to the second output of *split*, labelled as *[tmp]*, is processed through the *crop* filter, which crops away the lower half part of the video, and then vertically flipped. The *overlay* filter takes in input the first unchanged output of the split filter (which was labelled as *[main]*), and overlay on its lower half the output generated by the *crop,vflip* filterchain.

Some filters take in input a list of parameters: they are specified after the filter name and an equal sign, and are separated from each other by a colon.

There exist so-called *source filters* that do not have an audio/video input, and *sink filters* that will not have audio/video output.

## 31 graph2dot# TOC

The `graph2dot` program included in the FFmpeg `tools` directory can be used to parse a filtergraph description and issue a corresponding textual representation in the dot language.

Invoke the command:

```
graph2dot -h
```

to see how to use `graph2dot`.

You can then pass the dot description to the `dot` program (from the graphviz suite of programs) and obtain a graphical representation of the filtergraph.

For example the sequence of commands:

```
echo GRAPH_DESCRIPTION | \
tools/graph2dot -o graph.tmp && \
dot -Tpng graph.tmp -o graph.png && \
display graph.png
```

can be used to create and display an image representing the graph described by the *GRAPH\_DESCRIPTION* string. Note that this string must be a complete self-contained graph, with its inputs and outputs explicitly defined. For example if your command line is of the form:

```
ffmpeg -i infile -vf scale=640:360 outfile
```

your *GRAPH\_DESCRIPTION* string will need to be of the form:

```
nullsrc,scale=640:360,nullsink
```

you may also need to set the *nullsrc* parameters and add a *format* filter in order to simulate a specific input file.

## 32 Filtergraph description# TOC

A filtergraph is a directed graph of connected filters. It can contain cycles, and there can be multiple links between a pair of filters. Each link has one input pad on one side connecting it to one filter from which it takes its input, and one output pad on the other side connecting it to one filter accepting its output.

Each filter in a filtergraph is an instance of a filter class registered in the application, which defines the features and the number of input and output pads of the filter.

A filter with no input pads is called a "source", and a filter with no output pads is called a "sink".

### 32.1 Filtergraph syntax# TOC

A filtergraph has a textual representation, which is recognized by the `-filter/-vf/-af` and `-filter_complex` options in `ffmpeg` and `-vf/-af` in `ffplay`, and by the `avfilter_graph_parse_ptr()` function defined in `libavfilter/avfilter.h`.

A filterchain consists of a sequence of connected filters, each one connected to the previous one in the sequence. A filterchain is represented by a list of `","`-separated filter descriptions.

A filtergraph consists of a sequence of filterchains. A sequence of filterchains is represented by a list of `";"`-separated filterchain descriptions.

A filter is represented by a string of the form:

`[in_link_1]...[in_link_N]filter_name=arguments[out_link_1]...[out_link_M]`

*filter\_name* is the name of the filter class of which the described filter is an instance of, and has to be the name of one of the filter classes registered in the program. The name of the filter class is optionally followed by a string `"=arguments"`.

*arguments* is a string which contains the parameters used to initialize the filter instance. It may have one of two forms:

- A `':'`-separated list of *key=value* pairs.
- A `':'`-separated list of *value*. In this case, the keys are assumed to be the option names in the order they are declared. E.g. the `fade` filter declares three options in this order – `type`, `start_frame` and `nb_frames`. Then the parameter list `in:0:30` means that the value `in` is assigned to the option `type`, `0` to `start_frame` and `30` to `nb_frames`.
- A `':'`-separated list of mixed direct *value* and long *key=value* pairs. The direct *value* must precede the *key=value* pairs, and follow the same constraints order of the previous point. The following *key=value* pairs can be set in any preferred order.

If the option value itself is a list of items (e.g. the `format` filter takes a list of pixel formats), the items in the list are usually separated by `'|'`.

The list of arguments can be quoted using the character `'` as initial and ending mark, and the character `\` for escaping the characters within the quoted text; otherwise the argument string is considered terminated when the next special character (belonging to the set `[ ] = ; , '`) is encountered.

The name and arguments of the filter are optionally preceded and followed by a list of link labels. A link label allows one to name a link and associate it to a filter output or input pad. The preceding labels *in\_link\_1* ... *in\_link\_N*, are associated to the filter input pads, the following labels *out\_link\_1* ... *out\_link\_M*, are associated to the output pads.

When two link labels with the same name are found in the filtergraph, a link between the corresponding input and output pad is created.

If an output pad is not labelled, it is linked by default to the first unlabelled input pad of the next filter in the filterchain. For example in the filterchain

```
nullsrc, split[L1], [L2]overlay, nullsink
```

the `split` filter instance has two output pads, and the `overlay` filter instance two input pads. The first output pad of `split` is labelled "L1", the first input pad of `overlay` is labelled "L2", and the second output pad of `split` is linked to the second input pad of `overlay`, which are both unlabelled.

In a filter description, if the input label of the first filter is not specified, "in" is assumed; if the output label of the last filter is not specified, "out" is assumed.

In a complete filterchain all the unlabelled filter input and output pads must be connected. A filtergraph is considered valid if all the filter input and output pads of all the filterchains are connected.

Libavfilter will automatically insert scale filters where format conversion is required. It is possible to specify `sws_flags` for those automatically inserted scalers by prepending `sws_flags=flags;` to the filtergraph description.

Here is a BNF description of the filtergraph syntax:

```
NAME          ::= sequence of alphanumeric characters and '_'
LINKLABEL     ::= "[" NAME "]"
LINKLABELS    ::= LINKLABEL [LINKLABELS]
FILTER_ARGUMENTS ::= sequence of chars (possibly quoted)
FILTER        ::= [LINKLABELS] NAME ["=" FILTER_ARGUMENTS] [LINKLABELS]
FILTERCHAIN   ::= FILTER [ , FILTERCHAIN ]
FILTERGRAPH   ::= [sws_flags=flags;] FILTERCHAIN [ ; FILTERGRAPH ]
```

## 32.2 Notes on filtergraph escaping# TOC

Filtergraph description composition entails several levels of escaping. See (ffmpeg-utils)the "Quoting and escaping" section in the ffmpeg-utils(1) manual for more information about the employed escaping procedure.



A first level escaping affects the content of each filter option value, which may contain the special character `:` used to separate values, or one of the escaping characters `\` `'`.

A second level escaping affects the whole filter description, which may contain the escaping characters `\` `'` or the special characters `[ ] , ;` used by the filtergraph description.

Finally, when you specify a filtergraph on a shell commandline, you need to perform a third level escaping for the shell special characters contained within it.

For example, consider the following string to be embedded in the drawtext filter description `text` value:

```
this is a 'string': may contain one, or more, special characters
```

This string contains the `'` special escaping character, and the `:` special character, so it needs to be escaped in this way:

```
text=this is a \'string\': may contain one, or more, special characters
```

A second level of escaping is required when embedding the filter description in a filtergraph description, in order to escape all the filtergraph special characters. Thus the example above becomes:

```
drawtext=text=this is a \\\'string\\\': may contain one\\, or more\\, special characters
```

(note that in addition to the `\` `'` escaping special characters, also `,` needs to be escaped).

Finally an additional level of escaping is needed when writing the filtergraph description in a shell command, which depends on the escaping rules of the adopted shell. For example, assuming that `\` is special and needs to be escaped with another `\`, the previous string will finally result in:

```
-vf "drawtext=text=this is a \\\\'string\\\\\\': may contain one\\\\, or more\\\\, special characters"
```

## 33 Timeline editing# TOC

Some filters support a generic `enable` option. For the filters supporting timeline editing, this option can be set to an expression which is evaluated before sending a frame to the filter. If the evaluation is non-zero, the filter will be enabled, otherwise the frame will be sent unchanged to the next filter in the filtergraph.

The expression accepts the following values:

`'t'`

timestamp expressed in seconds, NAN if the input timestamp is unknown

`'n'`

sequential number of the input frame, starting from 0

`'pos'`

the position in the file of the input frame, NAN if unknown

`'w'`

`'h'`

width and height of the input frame if video

Additionally, these filters support an `enable` command that can be used to re-define the expression.

Like any other filtering option, the `enable` option follows the same rules.

For example, to enable a blur filter (`smartblur`) from 10 seconds to 3 minutes, and a curves filter starting at 3 seconds:

```
smartblur = enable='between(t,10,3*60)',  
curves    = enable='gte(t,3)' : preset=cross_process
```

## 34 Audio Filters# TOC

When you configure your FFmpeg build, you can disable any of the existing filters using `--disable-filters`. The configure output will show the audio filters included in your build.

Below is a description of the currently available audio filters.

### 34.1 acrossfade# TOC

Apply cross fade from one input audio stream to another input audio stream. The cross fade is applied for specified duration near the end of first stream.

The filter accepts the following options:

`nb_samples, ns`

Specify the number of samples for which the cross fade effect has to last. At the end of the cross fade effect the first input audio will be completely silent. Default is 44100.

`duration, d`

Specify the duration of the cross fade effect. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. By default the duration is determined by `nb_samples`. If set this option is used instead of `nb_samples`.

`overlap, o`

Should first stream end overlap with second stream start. Default is enabled.

curve1

Set curve for cross fade transition for first stream.

curve2

Set curve for cross fade transition for second stream.

For description of available curve types see afade filter description.

### 34.1.1 Examples# TOC

- Cross fade from one input to another:

```
ffmpeg -i first.flac -i second.flac -filter_complex acrossfade=d=10:c1=exp:c2=exp output.flac
```

- Cross fade from one input to another but without overlapping:

```
ffmpeg -i first.flac -i second.flac -filter_complex acrossfade=d=10:o=0:c1=exp:c2=exp output.flac
```

## 34.2 adelay# TOC

Delay one or more audio channels.

Samples in delayed channel are filled with silence.

The filter accepts the following option:

delays

Set list of delays in milliseconds for each channel separated by '|'. At least one delay greater than 0 should be provided. Unused delays will be silently ignored. If number of given delays is smaller than number of channels all remaining channels will not be delayed.

### 34.2.1 Examples# TOC

- Delay first channel by 1.5 seconds, the third channel by 0.5 seconds and leave the second channel (and any other channels that may be present) unchanged.

```
adelay=1500|0|500
```

## 34.3 aecho# TOC

Apply echoing to the input audio.

Echoes are reflected sound and can occur naturally amongst mountains (and sometimes large buildings) when talking or shouting; digital echo effects emulate this behaviour and are often used to help fill out the sound of a single instrument or vocal. The time difference between the original signal and the reflection is the `delay`, and the loudness of the reflected signal is the `decay`. Multiple echoes can have different delays and decays.

A description of the accepted parameters follows.

`in_gain`

Set input gain of reflected signal. Default is 0.6.

`out_gain`

Set output gain of reflected signal. Default is 0.3.

`delays`

Set list of time intervals in milliseconds between original signal and reflections separated by '|'. Allowed range for each delay is (0 - 90000.0]. Default is 1000.

`decays`

Set list of loudnesses of reflected signals separated by '|'. Allowed range for each decay is (0 - 1.0]. Default is 0.5.

### 34.3.1 Examples# TOC

- Make it sound as if there are twice as many instruments as are actually playing:

```
aecho=0.8:0.88:60:0.4
```

- If delay is very short, then it sound like a (metallic) robot playing music:

```
aecho=0.8:0.88:6:0.4
```

- A longer delay will sound like an open air concert in the mountains:

```
aecho=0.8:0.9:1000:0.3
```

- Same as above but with one more mountain:

```
aecho=0.8:0.9:1000|1800:0.3|0.25
```

## 34.4 aeval# TOC

Modify an audio signal according to the specified expressions.

This filter accepts one or more expressions (one for each channel), which are evaluated and used to modify a corresponding audio signal.

It accepts the following parameters:

`exprs`

Set the '|' -separated expressions list for each separate channel. If the number of input channels is greater than the number of expressions, the last specified expression is used for the remaining output channels.

`channel_layout, c`

Set output channel layout. If not specified, the channel layout is specified by the number of expressions. If set to 'same', it will use by default the same input channel layout.

Each expression in *exprs* can contain the following constants and functions:

`ch`

channel number of the current expression

`n`

number of the evaluated sample, starting from 0

`s`

sample rate

`t`

time of the evaluated sample expressed in seconds

`nb_in_channels`

`nb_out_channels`

input and output number of channels

`val(CH)`

the value of input channel with number *CH*

Note: this filter is slow. For faster processing you should use a dedicated filter.

### 34.4.1 Examples# TOC

- Half volume:

```
aeval=val(ch)/2:c=same
```

- Invert phase of the second channel:

```
aeval=val(0)|-val(1)
```

## 34.5 afade# TOC

Apply fade-in/out effect to input audio.

A description of the accepted parameters follows.

`type, t`

Specify the effect type, can be either `in` for fade-in, or `out` for a fade-out effect. Default is `in`.

`start_sample, ss`

Specify the number of the start sample for starting to apply the fade effect. Default is 0.

`nb_samples, ns`

Specify the number of samples for which the fade effect has to last. At the end of the fade-in effect the output audio will have the same volume as the input audio, at the end of the fade-out transition the output audio will be silence. Default is 44100.

`start_time, st`

Specify the start time of the fade effect. Default is 0. The value must be specified as a time duration; see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. If set this option is used instead of *start\_sample*.

`duration, d`

Specify the duration of the fade effect. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. At the end of the fade-in effect the output audio will have the same volume as the input audio, at the end of the fade-out transition the output audio will be silence. By default the duration is determined by *nb\_samples*. If set this option is used instead of *nb\_samples*.

`curve`

Set curve for fade transition.

It accepts the following values:

`tri`

select triangular, linear slope (default)

`qsin`

select quarter of sine wave

hsin

select half of sine wave

esin

select exponential sine wave

log

select logarithmic

ipar

select inverted parabola

qua

select quadratic

cub

select cubic

squ

select square root

cbr

select cubic root

par

select parabola

exp

select exponential

iqsin

select inverted quarter of sine wave

ihsin

select inverted half of sine wave

dese

```
select double-exponential seat  
desi  
select double-exponential sigmoid
```

### 34.5.1 Examples# TOC

- Fade in first 15 seconds of audio:

```
afade=t=in:ss=0:d=15
```

- Fade out last 25 seconds of a 900 seconds audio:

```
afade=t=out:st=875:d=25
```

### 34.6 aformat# TOC

Set output format constraints for the input audio. The framework will negotiate the most appropriate format to minimize conversions.

It accepts the following parameters:

`sample_fmts`

A '|' -separated list of requested sample formats.

`sample_rates`

A '|' -separated list of requested sample rates.

`channel_layouts`

A '|' -separated list of requested channel layouts.

See (ffmpeg-utils)the Channel Layout section in the ffmpeg-utils(1) manual for the required syntax.

If a parameter is omitted, all values are allowed.

Force the output to either unsigned 8-bit or signed 16-bit stereo

```
aformat=sample_fmts=u8|s16:channel_layouts=stereo
```

### 34.7 allpass# TOC

Apply a two-pole all-pass filter with central frequency (in Hz) *frequency*, and filter-width *width*. An all-pass filter changes the audio's frequency to phase relationship without changing its frequency to amplitude relationship.



The filter accepts the following options:

`frequency, f`

Set frequency in Hz.

`width_type`

Set method to specify band-width of filter.

`h`

Hz

`q`

Q-Factor

`o`

octave

`s`

slope

`width, w`

Specify the band-width of a filter in `width_type` units.

## 34.8 amerge# TOC

Merge two or more audio streams into a single multi-channel stream.

The filter accepts the following options:

`inputs`

Set the number of inputs. Default is 2.

If the channel layouts of the inputs are disjoint, and therefore compatible, the channel layout of the output will be set accordingly and the channels will be reordered as necessary. If the channel layouts of the inputs are not disjoint, the output will have all the channels of the first input then all the channels of the second input, in that order, and the channel layout of the output will be the default value corresponding to the total number of channels.

For example, if the first input is in 2.1 (FL+FR+LF) and the second input is FC+BL+BR, then the output will be in 5.1, with the channels in the following order: a1, a2, b1, a3, b2, b3 (a1 is the first channel of the first input, b1 is the first channel of the second input).

On the other hand, if both input are in stereo, the output channels will be in the default order: a1, a2, b1, b2, and the channel layout will be arbitrarily set to 4.0, which may or may not be the expected value.

All inputs must have the same sample rate, and format.

If inputs do not have the same duration, the output will stop with the shortest.

### 34.8.1 Examples# TOC

- Merge two mono files into a stereo stream:

```
amovie=left.wav [l] ; amovie=right.mp3 [r] ; [l] [r] amerge
```

- Multiple merges assuming 1 video stream and 6 audio streams in `input.mkv`:

```
ffmpeg -i input.mkv -filter_complex "[0:1][0:2][0:3][0:4][0:5][0:6] amerge=inputs=6" -c:a pcm_s16le output.mkv
```

## 34.9 amix# TOC

Mixes multiple audio inputs into a single output.

Note that this filter only supports float samples (the *amerge* and *pan* audio filters support many formats). If the *amix* input has integer samples then *aresample* will be automatically inserted to perform the conversion to float samples.

For example

```
ffmpeg -i INPUT1 -i INPUT2 -i INPUT3 -filter_complex amix=inputs=3:duration=first:dropout_transition=3 OUTPUT
```

will mix 3 input audio streams to a single output with the same duration as the first input and a dropout transition time of 3 seconds.

It accepts the following parameters:

`inputs`

The number of inputs. If unspecified, it defaults to 2.

`duration`

How to determine the end-of-stream.

`longest`

The duration of the longest input. (default)

`shortest`

The duration of the shortest input.

`first`

The duration of the first input.

`dropout_transition`

The transition time, in seconds, for volume renormalization when an input stream ends. The default value is 2 seconds.

## 34.10 `anull#` TOC

Pass the audio source unchanged to the output.

## 34.11 `apad#` TOC

Pad the end of an audio stream with silence.

This can be used together with `ffmpeg -shortest` to extend audio streams to the same length as the video stream.

A description of the accepted options follows.

`packet_size`

Set silence packet size. Default value is 4096.

`pad_len`

Set the number of samples of silence to add to the end. After the value is reached, the stream is terminated. This option is mutually exclusive with `whole_len`.

`whole_len`

Set the minimum total number of samples in the output audio stream. If the value is longer than the input audio length, silence is added to the end, until the value is reached. This option is mutually exclusive with `pad_len`.

If neither the `pad_len` nor the `whole_len` option is set, the filter will add silence to the end of the input stream indefinitely.

### 34.11.1 `Examples#` TOC

- Add 1024 samples of silence to the end of the input:

```
apad=pad_len=1024
```

- Make sure the audio output will contain at least 10000 samples, pad the input with silence if required:

```
apad=whole_len=10000
```

- Use `ffmpeg` to pad the audio input with silence, so that the video stream will always result the shortest and will be converted until the end in the output file when using the `shortest` option:

```
ffmpeg -i VIDEO -i AUDIO -filter_complex "[1:0]apad" -shortest OUTPUT
```

## 34.12 aphaser# TOC

Add a phasing effect to the input audio.

A phaser filter creates series of peaks and troughs in the frequency spectrum. The position of the peaks and troughs are modulated so that they vary over time, creating a sweeping effect.

A description of the accepted parameters follows.

`in_gain`

Set input gain. Default is 0.4.

`out_gain`

Set output gain. Default is 0.74

`delay`

Set delay in milliseconds. Default is 3.0.

`decay`

Set decay. Default is 0.4.

`speed`

Set modulation speed in Hz. Default is 0.5.

`type`

Set modulation type. Default is triangular.

It accepts the following values:

`'triangular, t'`

`'sinusoidal, s'`

## 34.13 aresample# TOC

Resample the input audio to the specified parameters, using the libswresample library. If none are specified then the filter will automatically convert between its input and output.

This filter is also able to stretch/squeeze the audio data to make it match the timestamps or to inject silence / cut out audio to make it match the timestamps, do a combination of both or do neither.

The filter accepts the syntax `[sample_rate:]resampler_options`, where *sample\_rate* expresses a sample rate and *resampler\_options* is a list of *key=value* pairs, separated by ":". See the ffmpeg-resampler manual for the complete list of supported options.

### 34.13.1 Examples# TOC

- Resample the input audio to 44100Hz:

```
aresample=44100
```

- Stretch/squeeze samples to the given timestamps, with a maximum of 1000 samples per second compensation:

```
aresample=async=1000
```

## 34.14 asetnsamples# TOC

Set the number of samples per each output audio frame.

The last output packet may contain a different number of samples, as the filter will flush all the remaining samples when the input audio signal its end.

The filter accepts the following options:

`nb_out_samples, n`

Set the number of frames per each output audio frame. The number is intended as the number of samples *per each channel*. Default value is 1024.

`pad, p`

If set to 1, the filter will pad the last audio frame with zeroes, so that the last frame will contain the same number of samples as the previous ones. Default value is 1.

For example, to set the number of per-frame samples to 1234 and disable padding for the last frame, use:

```
asetnsamples=n=1234:p=0
```

## 34.15 asetrate# TOC

Set the sample rate without altering the PCM data. This will result in a change of speed and pitch.

The filter accepts the following options:

`sample_rate, r`

Set the output sample rate. Default is 44100 Hz.

## 34.16 ashowinfo# TOC

Show a line containing various information for each input audio frame. The input audio is not modified.

The shown line contains a sequence of key/value pairs of the form *key:value*.

The following values are shown in the output:

`n`

The (sequential) number of the input frame, starting from 0.

`pts`

The presentation timestamp of the input frame, in time base units; the time base depends on the filter input pad, and is usually  $1/\text{sample\_rate}$ .

`pts_time`

The presentation timestamp of the input frame in seconds.

`pos`

position of the frame in the input stream, -1 if this information is unavailable and/or meaningless (for example in case of synthetic audio)

`fmt`

The sample format.

`chlayout`

The channel layout.

`rate`

The sample rate for the audio frame.

`nb_samples`

The number of samples (per channel) in the frame.

`checksum`

The Adler-32 checksum (printed in hexadecimal) of the audio data. For planar audio, the data is treated as if all the planes were concatenated.

`plane_checksums`

A list of Adler-32 checksums for each data plane.

## 34.17 `astats`# TOC

Display time domain statistical information about the audio channels. Statistics are calculated and displayed for each audio channel and, where applicable, an overall figure is also given.

It accepts the following option:

`length`

Short window length in seconds, used for peak and trough RMS measurement. Default is 0.05 (50 milliseconds). Allowed range is [0.1 - 10].

`metadata`

Set metadata injection. All the metadata keys are prefixed with `lavfi.astats.X`, where X is channel number starting from 1 or string `Overall`. Default is disabled.

Available keys for each channel are: `DC_offset` `Min_level` `Max_level` `Min_difference` `Max_difference` `Mean_difference` `Peak_level` `RMS_peak` `RMS_trough` `Crest_factor` `Flat_factor` `Peak_count` `Bit_depth`

and for Overall: `DC_offset` `Min_level` `Max_level` `Min_difference` `Max_difference` `Mean_difference` `Peak_level` `RMS_level` `RMS_peak` `RMS_trough` `Flat_factor` `Peak_count` `Bit_depth` `Number_of_samples`

For example full key look like this `lavfi.astats.1.DC_offset` or this `lavfi.astats.Overall.Peak_count`.

For description what each key means read below.

`reset`

Set number of frame after which stats are going to be recalculated. Default is disabled.

A description of each shown parameter follows:

DC offset

Mean amplitude displacement from zero.

Min level

Minimal sample level.

Max level

Maximal sample level.

Min difference

Minimal difference between two consecutive samples.

Max difference

Maximal difference between two consecutive samples.

Mean difference

Mean difference between two consecutive samples. The average of each difference between two consecutive samples.

Peak level dB

RMS level dB

Standard peak and RMS level measured in dBFS.

RMS peak dB

RMS trough dB

Peak and trough values for RMS level measured over a short window.

Crest factor

Standard ratio of peak to RMS level (note: not in dB).

Flat factor

Flatness (i.e. consecutive samples with the same value) of the signal at its peak levels (i.e. either *Min level* or *Max level*).

Peak count



Number of occasions (not the number of samples) that the signal attained either *Min level* or *Max level*.

Bit depth

Overall bit depth of audio. Number of bits used for each sample.

## 34.18 astreamsync# TOC

Forward two audio streams and control the order the buffers are forwarded.

The filter accepts the following options:

`expr, e`

Set the expression deciding which stream should be forwarded next: if the result is negative, the first stream is forwarded; if the result is positive or zero, the second stream is forwarded. It can use the following variables:

*b1 b2*

number of buffers forwarded so far on each stream

*s1 s2*

number of samples forwarded so far on each stream

*t1 t2*

current timestamp of each stream

The default value is  $t1 - t2$ , which means to always forward the stream that has a smaller timestamp.

### 34.18.1 Examples# TOC

Stress-test `amerge` by randomly sending buffers on the wrong input, while avoiding too much of a desynchronization:

```
amovie=file.ogg [a] ; amovie=file.mp3 [b] ;  
[a] [b] astreamsync=(2*random(1))-1+tanh(5*(t1-t2)) [a2] [b2] ;  
[a2] [b2] amerge
```

## 34.19 asyncts# TOC

Synchronize audio data with timestamps by squeezing/stretching it and/or dropping samples/adding silence when needed.

This filter is not built by default, please use aresample to do squeezing/stretching.

It accepts the following parameters:

`compensate`

Enable stretching/squeezing the data to make it match the timestamps. Disabled by default. When disabled, time gaps are covered with silence.

`min_delta`

The minimum difference between timestamps and audio data (in seconds) to trigger adding/dropping samples. The default value is 0.1. If you get an imperfect sync with this filter, try setting this parameter to 0.

`max_comp`

The maximum compensation in samples per second. Only relevant with `compensate=1`. The default value is 500.

`first_pts`

Assume that the first PTS should be this value. The time base is 1 / sample rate. This allows for padding/trimming at the start of the stream. By default, no assumption is made about the first frame's expected PTS, so no padding or trimming is done. For example, this could be set to 0 to pad the beginning with silence if an audio stream starts after the video stream or to trim any samples with a negative PTS due to encoder delay.

## 34.20 atempo# TOC

Adjust audio tempo.

The filter accepts exactly one parameter, the audio tempo. If not specified then the filter will assume nominal 1.0 tempo. Tempo must be in the [0.5, 2.0] range.

### 34.20.1 Examples# TOC

- Slow down audio to 80% tempo:

`atempo=0.8`

- To speed up audio to 125% tempo:

`atempo=1.25`

## 34.21 atrim# TOC

Trim the input so that the output contains one continuous subpart of the input.

It accepts the following parameters:

`start`

Timestamp (in seconds) of the start of the section to keep. I.e. the audio sample with the timestamp *start* will be the first sample in the output.

`end`

Specify time of the first audio sample that will be dropped, i.e. the audio sample immediately preceding the one with the timestamp *end* will be the last sample in the output.

`start_pts`

Same as *start*, except this option sets the start timestamp in samples instead of seconds.

`end_pts`

Same as *end*, except this option sets the end timestamp in samples instead of seconds.

`duration`

The maximum duration of the output in seconds.

`start_sample`

The number of the first sample that should be output.

`end_sample`

The number of the first sample that should be dropped.

`start`, `end`, and `duration` are expressed as time duration specifications; see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

Note that the first two sets of the `start/end` options and the `duration` option look at the frame timestamp, while the `_sample` options simply count the samples that pass through the filter. So `start/end_pts` and `start/end_sample` will give different results when the timestamps are wrong, inexact or do not start at zero. Also note that this filter does not modify the timestamps. If you wish to have the output timestamps start at zero, insert the `asetpts` filter after the `atrim` filter.

If multiple `start` or `end` options are set, this filter tries to be greedy and keep all samples that match at least one of the specified constraints. To keep only the part that matches all the constraints at once, chain multiple `atrim` filters.

The defaults are such that all the input is kept. So it is possible to set e.g. just the end values to keep everything before the specified time.

Examples:

- Drop everything except the second minute of input:

```
ffmpeg -i INPUT -af atrim=60:120
```

- Keep only the first 1000 samples:

```
ffmpeg -i INPUT -af atrim=end_sample=1000
```

## 34.22 bandpass# TOC

Apply a two-pole Butterworth band-pass filter with central frequency *frequency*, and (3dB-point) band-width *width*. The *csg* option selects a constant skirt gain (peak gain = Q) instead of the default: constant 0dB peak gain. The filter roll off at 6dB per octave (20dB per decade).

The filter accepts the following options:

*frequency*, *f*

Set the filter's central frequency. Default is 3000.

*csg*

Constant skirt gain if set to 1. Defaults to 0.

*width\_type*

Set method to specify band-width of filter.

*h*

Hz

*q*

Q-Factor

*o*

octave

*s*

slope

`width, w`

Specify the band-width of a filter in `width_type` units.

### 34.23 bandreject# TOC

Apply a two-pole Butterworth band-reject filter with central frequency *frequency*, and (3dB-point) band-width *width*. The filter roll off at 6dB per octave (20dB per decade).

The filter accepts the following options:

`frequency, f`

Set the filter's central frequency. Default is 3000.

`width_type`

Set method to specify band-width of filter.

`h`

Hz

`q`

Q-Factor

`o`

octave

`s`

slope

`width, w`

Specify the band-width of a filter in `width_type` units.

### 34.24 bass# TOC

Boost or cut the bass (lower) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

The filter accepts the following options:

`gain, g`

Give the gain at 0 Hz. Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of clipping when using a positive gain.

frequency, *f*

Set the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 100 Hz.

width\_type

Set method to specify band-width of filter.

h

Hz

q

Q-Factor

o

octave

s

slope

width, *w*

Determine how steep is the filter's shelf transition.

## 34.25 biquad# TOC

Apply a biquad IIR filter with the given coefficients. Where *b0*, *b1*, *b2* and *a0*, *a1*, *a2* are the numerator and denominator coefficients respectively.

## 34.26 bs2b# TOC

Bauer stereo to binaural transformation, which improves headphone listening of stereo audio records.

It accepts the following parameters:

profile

Pre-defined crossfeed level.

default

Default level (fcut=700, feed=50).

cmoy

Chu Moy circuit (fcut=700, feed=60).

jmeier

Jan Meier circuit (fcut=650, feed=95).

fcut

Cut frequency (in Hz).

feed

Feed level (in Hz).

## 34.27 channelmap# TOC

Remap input channels to new locations.

It accepts the following parameters:

channel\_layout

The channel layout of the output stream.

map

Map channels from input to output. The argument is a '|'-separated list of mappings, each in the *in\_channel-out\_channel* or *in\_channel* form. *in\_channel* can be either the name of the input channel (e.g. FL for front left) or its index in the input channel layout. *out\_channel* is the name of the output channel or its index in the output channel layout. If *out\_channel* is not given then it is implicitly an index, starting with zero and increasing by one for each mapping.

If no mapping is present, the filter will implicitly map input channels to output channels, preserving indices.

For example, assuming a 5.1+downmix input MOV file,

```
ffmpeg -i in.mov -filter 'channelmap=map=DL-FL|DR-FR' out.wav
```

will create an output WAV file tagged as stereo from the downmix channels of the input.

To fix a 5.1 WAV improperly encoded in AAC's native channel order

```
ffmpeg -i in.wav -filter 'channelmap=1|2|0|5|3|4:5.1' out.wav
```

## 34.28 channelsplit# TOC

Split each channel from an input audio stream into a separate output stream.

It accepts the following parameters:

`channel_layout`

The channel layout of the input stream. The default is "stereo".

For example, assuming a stereo input MP3 file,

```
ffmpeg -i in.mp3 -filter_complex channelsplit out.mkv
```

will create an output Matroska file with two audio streams, one containing only the left channel and the other the right channel.

Split a 5.1 WAV file into per-channel files:

```
ffmpeg -i in.wav -filter_complex  
'channelsplit=channel_layout=5.1[FL][FR][FC][LFE][SL][SR]'  
-map '[FL]' front_left.wav -map '[FR]' front_right.wav -map '[FC]'  
front_center.wav -map '[LFE]' lfe.wav -map '[SL]' side_left.wav -map '[SR]'  
side_right.wav
```

## 34.29 chorus# TOC

Add a chorus effect to the audio.

Can make a single vocal sound like a chorus, but can also be applied to instrumentation.

Chorus resembles an echo effect with a short delay, but whereas with echo the delay is constant, with chorus, it is varied using sinusoidal or triangular modulation. The modulation depth defines the range the modulated delay is played before or after the delay. Hence the delayed sound will sound slower or faster, that is the delayed sound tuned around the original one, like in a chorus where some vocals are slightly off key.

It accepts the following parameters:

`in_gain`

Set input gain. Default is 0.4.

`out_gain`



Set output gain. Default is 0.4.

delays

Set delays. A typical delay is around 40ms to 60ms.

decays

Set decays.

speeds

Set speeds.

depths

Set depths.

### 34.29.1 Examples# TOC

- A single delay:

```
chorus=0.7:0.9:55:0.4:0.25:2
```

- Two delays:

```
chorus=0.6:0.9:50|60:0.4|0.32:0.25|0.4:2|1.3
```

- Fuller sounding chorus with three delays:

```
chorus=0.5:0.9:50|60|40:0.4|0.32|0.3:0.25|0.4|0.3:2|2.3|1.3
```

### 34.30 compand# TOC

Compress or expand the audio's dynamic range.

It accepts the following parameters:

attacks

decays

A list of times in seconds for each channel over which the instantaneous level of the input signal is averaged to determine its volume. *attacks* refers to increase of volume and *decays* refers to decrease of volume. For most situations, the attack time (response to the audio getting louder) should be shorter than the decay time, because the human ear is more sensitive to sudden loud audio than sudden soft audio. A typical value for attack is 0.3 seconds and a typical value for decay is 0.8 seconds. If specified number of attacks & decays is lower than number of channels, the last set attack/decay will be used for all remaining channels.

#### points

A list of points for the transfer function, specified in dB relative to the maximum possible signal amplitude. Each key points list must be defined using the following syntax:

`x0/y0 | x1/y1 | x2/y2 | . . . .` or `x0/y0 x1/y1 x2/y2 . . . .`

The input values must be in strictly increasing order but the transfer function does not have to be monotonically rising. The point 0/0 is assumed but may be overridden (by 0/out-dBn). Typical values for the transfer function are -70/-70 | -60/-20.

#### soft-knee

Set the curve radius in dB for all joints. It defaults to 0.01.

#### gain

Set the additional gain in dB to be applied at all points on the transfer function. This allows for easy adjustment of the overall gain. It defaults to 0.

#### volume

Set an initial volume, in dB, to be assumed for each channel when filtering starts. This permits the user to supply a nominal level initially, so that, for example, a very large gain is not applied to initial signal levels before the companding has begun to operate. A typical value for audio which is initially quiet is -90 dB. It defaults to 0.

#### delay

Set a delay, in seconds. The input audio is analyzed immediately, but audio is delayed before being fed to the volume adjuster. Specifying a delay approximately equal to the attack/decay times allows the filter to effectively operate in predictive rather than reactive mode. It defaults to 0.

### 34.30.1 Examples# TOC

- Make music with both quiet and loud passages suitable for listening to in a noisy environment:

```
compand=.3|.3:1|1:-90/-60|-60/-40|-40/-30|-20/-20:6:0:-90:0.2
```

Another example for audio with whisper and explosion parts:

```
compand=0|0:1|1:-90/-900|-70/-70|-30/-9|0/-3:6:0:0:0
```

- A noise gate for when the noise is at a lower level than the signal:

```
compand=.1|.1:.2|.2:-900/-900|-50.1/-900|-50/-50:.01:0:-90:.1
```

- Here is another noise gate, this time for when the noise is at a higher level than the signal (making it, in some ways, similar to squelch):

```
compand=.1|.1:.1|.1:-45.1/-45.1|-45/-900|0/-900:.01:45:-90:.1
```

### 34.31 dcshift# TOC

Apply a DC shift to the audio.

This can be useful to remove a DC offset (caused perhaps by a hardware problem in the recording chain) from the audio. The effect of a DC offset is reduced headroom and hence volume. The `astats` filter can be used to determine if a signal has a DC offset.

`shift`

Set the DC shift, allowed range is [-1, 1]. It indicates the amount to shift the audio.

`limitergain`

Optional. It should have a value much less than 1 (e.g. 0.05 or 0.02) and is used to prevent clipping.

### 34.32 dynaudnorm# TOC

Dynamic Audio Normalizer.

This filter applies a certain amount of gain to the input audio in order to bring its peak magnitude to a target level (e.g. 0 dBFS). However, in contrast to more "simple" normalization algorithms, the Dynamic Audio Normalizer *dynamically* re-adjusts the gain factor to the input audio. This allows for applying extra gain to the "quiet" sections of the audio while avoiding distortions or clipping the "loud" sections. In other words: The Dynamic Audio Normalizer will "even out" the volume of quiet and loud sections, in the sense that the volume of each section is brought to the same target level. Note, however, that the Dynamic Audio Normalizer achieves this goal *without* applying "dynamic range compressing". It will retain 100% of the dynamic range *within* each section of the audio file.

`f`

Set the frame length in milliseconds. In range from 10 to 8000 milliseconds. Default is 500 milliseconds. The Dynamic Audio Normalizer processes the input audio in small chunks, referred to as frames. This is required, because a peak magnitude has no meaning for just a single sample value. Instead, we need to determine the peak magnitude for a contiguous sequence of sample values. While a "standard" normalizer would simply use the peak magnitude of the complete file, the Dynamic Audio Normalizer determines the peak magnitude individually for each frame. The length of a frame is specified in milliseconds. By default, the Dynamic Audio Normalizer uses a frame length of 500 milliseconds, which has been found to give good results with most files. Note that the exact frame length, in number of samples, will be determined automatically, based on the sampling rate of the individual input audio file.

`g`

Set the Gaussian filter window size. In range from 3 to 301, must be odd number. Default is 31.

Probably the most important parameter of the Dynamic Audio Normalizer is the `window_size` of the Gaussian smoothing filter. The filter's window size is specified in frames, centered around the current frame. For the sake of simplicity, this must be an odd number. Consequently, the default value of 31 takes into account the current frame, as well as the 15 preceding frames and the 15 subsequent frames. Using a larger window results in a stronger smoothing effect and thus in less gain variation, i.e. slower gain adaptation. Conversely, using a smaller window results in a weaker smoothing effect and thus in more gain variation, i.e. faster gain adaptation. In other words, the more you increase this value, the more the Dynamic Audio Normalizer will behave like a "traditional" normalization filter. On the contrary, the more you decrease this value, the more the Dynamic Audio Normalizer will behave like a dynamic range compressor.

p

Set the target peak value. This specifies the highest permissible magnitude level for the normalized audio input. This filter will try to approach the target peak magnitude as closely as possible, but at the same time it also makes sure that the normalized signal will never exceed the peak magnitude. A frame's maximum local gain factor is imposed directly by the target peak magnitude. The default value is 0.95 and thus leaves a headroom of 5%\*. It is not recommended to go above this value.

m

Set the maximum gain factor. In range from 1.0 to 100.0. Default is 10.0. The Dynamic Audio Normalizer determines the maximum possible (local) gain factor for each input frame, i.e. the maximum gain factor that does not result in clipping or distortion. The maximum gain factor is determined by the frame's highest magnitude sample. However, the Dynamic Audio Normalizer additionally bounds the frame's maximum gain factor by a predetermined (global) maximum gain factor. This is done in order to avoid excessive gain factors in "silent" or almost silent frames. By default, the maximum gain factor is 10.0, For most inputs the default value should be sufficient and it usually is not recommended to increase this value. Though, for input with an extremely low overall volume level, it may be necessary to allow even higher gain factors. Note, however, that the Dynamic Audio Normalizer does not simply apply a "hard" threshold (i.e. cut off values above the threshold). Instead, a "sigmoid" threshold function will be applied. This way, the gain factors will smoothly approach the threshold value, but never exceed that value.

r

Set the target RMS. In range from 0.0 to 1.0. Default is 0.0 - disabled. By default, the Dynamic Audio Normalizer performs "peak" normalization. This means that the maximum local gain factor for each frame is defined (only) by the frame's highest magnitude sample. This way, the samples can be amplified as much as possible without exceeding the maximum signal level, i.e. without clipping. Optionally, however, the Dynamic Audio Normalizer can also take into account the frame's root mean square, abbreviated RMS. In electrical engineering, the RMS is commonly used to determine the power of a time-varying signal. It is therefore considered that the RMS is a better approximation of the "perceived loudness" than just looking at the signal's peak magnitude. Consequently, by adjusting all frames to a constant RMS value, a uniform "perceived loudness" can be established. If a target RMS value has been specified, a frame's local gain factor is defined as the factor that would result in exactly that RMS value. Note, however, that the maximum local gain factor is still restricted

by the frame's highest magnitude sample, in order to prevent clipping.

n

Enable channels coupling. By default is enabled. By default, the Dynamic Audio Normalizer will amplify all channels by the same amount. This means the same gain factor will be applied to all channels, i.e. the maximum possible gain factor is determined by the "loudest" channel. However, in some recordings, it may happen that the volume of the different channels is uneven, e.g. one channel may be "quieter" than the other one(s). In this case, this option can be used to disable the channel coupling. This way, the gain factor will be determined independently for each channel, depending only on the individual channel's highest magnitude sample. This allows for harmonizing the volume of the different channels.

c

Enable DC bias correction. By default is disabled. An audio signal (in the time domain) is a sequence of sample values. In the Dynamic Audio Normalizer these sample values are represented in the -1.0 to 1.0 range, regardless of the original input format. Normally, the audio signal, or "waveform", should be centered around the zero point. That means if we calculate the mean value of all samples in a file, or in a single frame, then the result should be 0.0 or at least very close to that value. If, however, there is a significant deviation of the mean value from 0.0, in either positive or negative direction, this is referred to as a DC bias or DC offset. Since a DC bias is clearly undesirable, the Dynamic Audio Normalizer provides optional DC bias correction. With DC bias correction enabled, the Dynamic Audio Normalizer will determine the mean value, or "DC correction" offset, of each input frame and subtract that value from all of the frame's sample values which ensures those samples are centered around 0.0 again. Also, in order to avoid "gaps" at the frame boundaries, the DC correction offset values will be interpolated smoothly between neighbouring frames.

b

Enable alternative boundary mode. By default is disabled. The Dynamic Audio Normalizer takes into account a certain neighbourhood around each frame. This includes the preceding frames as well as the subsequent frames. However, for the "boundary" frames, located at the very beginning and at the very end of the audio file, not all neighbouring frames are available. In particular, for the first few frames in the audio file, the preceding frames are not known. And, similarly, for the last few frames in the audio file, the subsequent frames are not known. Thus, the question arises which gain factors should be assumed for the missing frames in the "boundary" region. The Dynamic Audio Normalizer implements two modes to deal with this situation. The default boundary mode assumes a gain factor of exactly 1.0 for the missing frames, resulting in a smooth "fade in" and "fade out" at the beginning and at the end of the input, respectively.

s

Set the compress factor. In range from 0.0 to 30.0. Default is 0.0. By default, the Dynamic Audio Normalizer does not apply "traditional" compression. This means that signal peaks will not be pruned and thus the full dynamic range will be retained within each local neighbourhood. However, in some cases it may be desirable to combine the Dynamic Audio Normalizer's normalization algorithm with a more "traditional" compression. For this purpose, the Dynamic Audio Normalizer provides an

optional compression (thresholding) function. If (and only if) the compression feature is enabled, all input frames will be processed by a soft knee thresholding function prior to the actual normalization process. Put simply, the thresholding function is going to prune all samples whose magnitude exceeds a certain threshold value. However, the Dynamic Audio Normalizer does not simply apply a fixed threshold value. Instead, the threshold value will be adjusted for each individual frame. In general, smaller parameters result in stronger compression, and vice versa. Values below 3.0 are not recommended, because audible distortion may appear.

### 34.33 earwax# TOC

Make audio easier to listen to on headphones.

This filter adds ‘cues’ to 44.1kHz stereo (i.e. audio CD format) audio so that when listened to on headphones the stereo image is moved from inside your head (standard for headphones) to outside and in front of the listener (standard for speakers).

Ported from SoX.

### 34.34 equalizer# TOC

Apply a two-pole peaking equalisation (EQ) filter. With this filter, the signal-level at and around a selected frequency can be increased or decreased, whilst (unlike bandpass and bandreject filters) that at all other frequencies is unchanged.

In order to produce complex equalisation curves, this filter can be given several times, each with a different central frequency.

The filter accepts the following options:

`frequency, f`

Set the filter’s central frequency in Hz.

`width_type`

Set method to specify band-width of filter.

`h`

Hz

`q`

Q-Factor

`o`

octave

s

slope

width, w

Specify the band-width of a filter in width\_type units.

gain, g

Set the required gain or attenuation in dB. Beware of clipping when using a positive gain.

### 34.34.1 Examples# TOC

- Attenuate 10 dB at 1000 Hz, with a bandwidth of 200 Hz:

```
equalizer=f=1000:width_type=h:width=200:g=-10
```

- Apply 2 dB gain at 1000 Hz with Q 1 and attenuate 5 dB at 100 Hz with Q 2:

```
equalizer=f=1000:width_type=q:width=1:g=2,equalizer=f=100:width_type=q:width=2:g=-5
```

### 34.35 flanger# TOC

Apply a flanging effect to the audio.

The filter accepts the following options:

delay

Set base delay in milliseconds. Range from 0 to 30. Default value is 0.

depth

Set added swep delay in milliseconds. Range from 0 to 10. Default value is 2.

regen

Set percentage regeneration (delayed signal feedback). Range from -95 to 95. Default value is 0.

width

Set percentage of delayed signal mixed with original. Range from 0 to 100. Default value is 71.

speed

Set sweeps per second (Hz). Range from 0.1 to 10. Default value is 0.5.

shape

Set swept wave shape, can be *triangular* or *sinusoidal*. Default value is *sinusoidal*.

phase

Set swept wave percentage-shift for multi channel. Range from 0 to 100. Default value is 25.

interp

Set delay-line interpolation, *linear* or *quadratic*. Default is *linear*.

### 34.36 highpass# TOC

Apply a high-pass filter with 3dB point frequency. The filter can be either single-pole, or double-pole (the default). The filter roll off at 6dB per pole per octave (20dB per pole per decade).

The filter accepts the following options:

frequency, f

Set frequency in Hz. Default is 3000.

poles, p

Set number of poles. Default is 2.

width\_type

Set method to specify band-width of filter.

h

Hz

q

Q-Factor

o

octave

s

slope



width, w

Specify the band-width of a filter in width\_type units. Applies only to double-pole filter. The default is 0.707q and gives a Butterworth response.

## 34.37 join# TOC

Join multiple input streams into one multi-channel stream.

It accepts the following parameters:

inputs

The number of input streams. It defaults to 2.

channel\_layout

The desired output channel layout. It defaults to stereo.

map

Map channels from inputs to output. The argument is a '|'-separated list of mappings, each in the *input\_idx.in\_channel-out\_channel* form. *input\_idx* is the 0-based index of the input stream. *in\_channel* can be either the name of the input channel (e.g. FL for front left) or its index in the specified input stream. *out\_channel* is the name of the output channel.

The filter will attempt to guess the mappings when they are not specified explicitly. It does so by first trying to find an unused matching input channel and if that fails it picks the first unused input channel.

Join 3 inputs (with properly set channel layouts):

```
ffmpeg -i INPUT1 -i INPUT2 -i INPUT3 -filter_complex join=inputs=3 OUTPUT
```

Build a 5.1 output from 6 single-channel streams:

```
ffmpeg -i fl -i fr -i fc -i sl -i sr -i lfe -filter_complex  
'join=inputs=6:channel_layout=5.1:map=0.0-FL|1.0-FR|2.0-FC|3.0-SL|4.0-SR|5.0-LFE'  
out
```

## 34.38 ladspa# TOC

Load a LADSPA (Linux Audio Developer's Simple Plugin API) plugin.

To enable compilation of this filter you need to configure FFmpeg with `--enable-ladspa`.

file, f

Specifies the name of LADSPA plugin library to load. If the environment variable LADSPA\_PATH is defined, the LADSPA plugin is searched in each one of the directories specified by the colon separated list in LADSPA\_PATH, otherwise in the standard LADSPA paths, which are in this order:  
HOME/.ladspa/lib/, /usr/local/lib/ladspa/, /usr/lib/ladspa/.

plugin, p

Specifies the plugin within the library. Some libraries contain only one plugin, but others contain many of them. If this is not set filter will list all available plugins within the specified library.

controls, c

Set the '|' separated list of controls which are zero or more floating point values that determine the behavior of the loaded plugin (for example delay, threshold or gain). Controls need to be defined using the following syntax: c0=value0|c1=value1|c2=value2|..., where *valuei* is the value set on the *i*-th control. If controls is set to help, all available controls and their valid ranges are printed.

sample\_rate, s

Specify the sample rate, default to 44100. Only used if plugin have zero inputs.

nb\_samples, n

Set the number of samples per channel per each output frame, default is 1024. Only used if plugin have zero inputs.

duration, d

Set the minimum duration of the sourced audio. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. Note that the resulting duration may be greater than the specified duration, as the generated audio is always cut at the end of a complete frame. If not specified, or the expressed duration is negative, the audio is supposed to be generated forever. Only used if plugin have zero inputs.

### 34.38.1 Examples# TOC

- List all available plugins within amp (LADSPA example plugin) library:

```
ladspa=file=amp
```

- List all available controls and their valid ranges for vcf\_notch plugin from VCF library:

```
ladspa=f=vcf:p=vcf_notch:c=help
```

- Simulate low quality audio equipment using Computer Music Toolkit (CMT) plugin library:

```
ladspa=file=cmt:plugin=lofi:controls=c0=22|c1=12|c2=12
```

- Add reverberation to the audio using TAP-plugins (Tom's Audio Processing plugins):

```
ladspa=file=tap_reverb:tap_reverb
```

- Generate white noise, with 0.2 amplitude:

```
ladspa=file=cmt:noise_source_white:c=c0=.2
```

- Generate 20 bpm clicks using plugin C\* Click - Metronome from the C\* Audio Plugin Suite (CAPS) library:

```
ladspa=file=caps:Click:c=c1=20'
```

- Apply C\* Eq10X2 - Stereo 10-band equaliser effect:

```
ladspa=caps:Eq10X2:c=c0=-48|c9=-24|c3=12|c4=2
```

### 34.38.2 Commands# TOC

This filter supports the following commands:

`cN`

Modify the  $N$ -th control value.

If the specified value is not valid, it is ignored and prior one is kept.

### 34.39 lowpass# TOC

Apply a low-pass filter with 3dB point frequency. The filter can be either single-pole or double-pole (the default). The filter roll off at 6dB per pole per octave (20dB per pole per decade).

The filter accepts the following options:

`frequency, f`

Set frequency in Hz. Default is 500.

`poles, p`

Set number of poles. Default is 2.

`width_type`

Set method to specify band-width of filter.

`h`

Hz

`q`

Q-Factor

o

octave

s

slope

width, w

Specify the band-width of a filter in width\_type units. Applies only to double-pole filter. The default is 0.707q and gives a Butterworth response.

## 34.40 pan# TOC

Mix channels with specific gain levels. The filter accepts the output channel layout followed by a set of channels definitions.

This filter is also designed to efficiently remap the channels of an audio stream.

The filter accepts parameters of the form: "*l|outdef|outdef|...*"

*l*

output channel layout or number of channels

*outdef*

output channel specification, of the form: "*out\_name=[gain\*]in\_name[+[gain\*]in\_name...]*"

*out\_name*

output channel to define, either a channel name (FL, FR, etc.) or a channel number (c0, c1, etc.)

*gain*

multiplicative coefficient for the channel, 1 leaving the volume unchanged

*in\_name*

input channel to use, see out\_name for details; it is not possible to mix named and numbered input channels

If the '=' in a channel specification is replaced by '<', then the gains for that specification will be renormalized so that the total is 1, thus avoiding clipping noise.

### 34.40.1 Mixing examples# TOC

For example, if you want to down-mix from stereo to mono, but with a bigger factor for the left channel:

```
pan=lc|c0=0.9*c0+0.1*c1
```

A customized down-mix to stereo that works automatically for 3-, 4-, 5- and 7-channels surround:

```
pan=stereo| FL < FL + 0.5*FC + 0.6*BL + 0.6*SL | FR < FR + 0.5*FC + 0.6*BR + 0.6*SR
```

Note that `ffmpeg` integrates a default down-mix (and up-mix) system that should be preferred (see `"-ac"` option) unless you have very specific needs.

### 34.40.2 Remapping examples# TOC

The channel remapping will be effective if, and only if:

- gain coefficients are zeroes or ones,
- only one input per channel output,

If all these conditions are satisfied, the filter will notify the user ("Pure channel mapping detected"), and use an optimized and lossless method to do the remapping.

For example, if you have a 5.1 source and want a stereo audio stream by dropping the extra channels:

```
pan="stereo| c0=FL | c1=FR"
```

Given the same source, you can also switch front left and front right channels and keep the input channel layout:

```
pan="5.1| c0=c1 | c1=c0 | c2=c2 | c3=c3 | c4=c4 | c5=c5"
```

If the input is a stereo audio stream, you can mute the front left channel (and still keep the stereo channel layout) with:

```
pan="stereo|c1=c1"
```

Still with a stereo audio stream input, you can copy the right channel in both front left and right:

```
pan="stereo| c0=FR | c1=FR"
```

## 34.41 replaygain# TOC

ReplayGain scanner filter. This filter takes an audio stream as an input and outputs it unchanged. At end of filtering it displays `track_gain` and `track_peak`.

## 34.42 resample# TOC

Convert the audio sample format, sample rate and channel layout. It is not meant to be used directly.

## 34.43 sidechaincompress# TOC

This filter acts like normal compressor but has the ability to compress detected signal using second input signal. It needs two input streams and returns one output stream. First input stream will be processed depending on second stream signal. The filtered signal then can be filtered with other filters in later stages of processing. See pan and amerge filter.

The filter accepts the following options:

threshold

If a signal of second stream raises above this level it will affect the gain reduction of first stream. By default is 0.125. Range is between 0.00097563 and 1.

ratio

Set a ratio about which the signal is reduced. 1:2 means that if the level raised 4dB above the threshold, it will be only 2dB above after the reduction. Default is 2. Range is between 1 and 20.

attack

Amount of milliseconds the signal has to rise above the threshold before gain reduction starts. Default is 20. Range is between 0.01 and 2000.

release

Amount of milliseconds the signal has to fall below the threshold before reduction is decreased again. Default is 250. Range is between 0.01 and 9000.

makeup

Set the amount by how much signal will be amplified after processing. Default is 2. Range is from 1 and 64.

knee

Curve the sharp knee around the threshold to enter gain reduction more softly. Default is 2.82843. Range is between 1 and 8.

link

Choose if the average level between all channels of side-chain stream or the louder(maximum) channel of side-chain stream affects the reduction. Default is average.

detection

Should the exact signal be taken in case of `peak` or an RMS one in case of `rms`. Default is `rms` which is mainly smoother.

### 34.43.1 Examples# TOC

- Full `ffmpeg` example taking 2 audio inputs, 1st input to be compressed depending on the signal of 2nd input and later compressed signal to be merged with 2nd input:

```
ffmpeg -i main.flac -i sidechain.flac -filter_complex "[1:a]asplit=2[sc][mix];[0:a][sc]sidechaincompress[compr];[compr][mix]amerge"
```

### 34.44 silencedetect# TOC

Detect silence in an audio stream.

This filter logs a message when it detects that the input audio volume is less or equal to a noise tolerance value for a duration greater or equal to the minimum detected noise duration.

The printed times and duration are expressed in seconds.

The filter accepts the following options:

`duration, d`

Set silence duration until notification (default is 2 seconds).

`noise, n`

Set noise tolerance. Can be specified in dB (in case "dB" is appended to the specified value) or amplitude ratio. Default is -60dB, or 0.001.

#### 34.44.1 Examples# TOC

- Detect 5 seconds of silence with -50dB noise tolerance:

```
silencedetect=n=-50dB:d=5
```

- Complete example with `ffmpeg` to detect silence with 0.0001 noise tolerance in `silence.mp3`:

```
ffmpeg -i silence.mp3 -af silencedetect=noise=0.0001 -f null -
```

### 34.45 silenceremove# TOC

Remove silence from the beginning, middle or end of the audio.

The filter accepts the following options:

`start_periods`

This value is used to indicate if audio should be trimmed at beginning of the audio. A value of zero indicates no silence should be trimmed from the beginning. When specifying a non-zero value, it trims audio up until it finds non-silence. Normally, when trimming silence from beginning of audio the *start\_periods* will be 1 but it can be increased to higher values to trim all audio up to specific count of non-silence periods. Default value is 0.

`start_duration`

Specify the amount of time that non-silence must be detected before it stops trimming audio. By increasing the duration, bursts of noises can be treated as silence and trimmed off. Default value is 0.

`start_threshold`

This indicates what sample value should be treated as silence. For digital audio, a value of 0 may be fine but for audio recorded from analog, you may wish to increase the value to account for background noise. Can be specified in dB (in case "dB" is appended to the specified value) or amplitude ratio. Default value is 0.

`stop_periods`

Set the count for trimming silence from the end of audio. To remove silence from the middle of a file, specify a *stop\_periods* that is negative. This value is then treated as a positive value and is used to indicate the effect should restart processing as specified by *start\_periods*, making it suitable for removing periods of silence in the middle of the audio. Default value is 0.

`stop_duration`

Specify a duration of silence that must exist before audio is not copied any more. By specifying a higher duration, silence that is wanted can be left in the audio. Default value is 0.

`stop_threshold`

This is the same as *start\_threshold* but for trimming silence from the end of audio. Can be specified in dB (in case "dB" is appended to the specified value) or amplitude ratio. Default value is 0.

`leave_silence`

This indicate that *stop\_duration* length of audio should be left intact at the beginning of each period of silence. For example, if you want to remove long pauses between words but do not want to remove the pauses completely. Default value is 0.



### 34.45.1 Examples# TOC

- The following example shows how this filter can be used to start a recording that does not contain the delay at the start which usually occurs between pressing the record button and the start of the performance:

```
silenceremove=1:5:0.02
```

### 34.46 treble# TOC

Boost or cut treble (upper) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

The filter accepts the following options:

gain, g

Give the gain at whichever is the lower of ~22 kHz and the Nyquist frequency. Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of clipping when using a positive gain.

frequency, f

Set the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 3000 Hz.

width\_type

Set method to specify band-width of filter.

h

Hz

q

Q-Factor

o

octave

s

slope

width, w

Determine how steep is the filter's shelf transition.

## 34.47 volume# TOC

Adjust the input audio volume.

It accepts the following parameters:

`volume`

Set audio volume expression.

Output values are clipped to the maximum value.

The output audio volume is given by the relation:

$$\text{output\_volume} = \text{volume} * \text{input\_volume}$$

The default value for *volume* is "1.0".

`precision`

This parameter represents the mathematical precision.

It determines which input sample formats will be allowed, which affects the precision of the volume scaling.

`fixed`

8-bit fixed-point; this limits input sample format to U8, S16, and S32.

`float`

32-bit floating-point; this limits input sample format to FLT. (default)

`double`

64-bit floating-point; this limits input sample format to DBL.

`replaygain`

Choose the behaviour on encountering ReplayGain side data in input frames.

`drop`

Remove ReplayGain side data, ignoring its contents (the default).

`ignore`

Ignore ReplayGain side data, but leave it in the frame.

track

Prefer the track gain, if present.

album

Prefer the album gain, if present.

replaygain\_preamp

Pre-amplification gain in dB to apply to the selected replaygain gain.

Default value for *replaygain\_preamp* is 0.0.

eval

Set when the volume expression is evaluated.

It accepts the following values:

‘once’

only evaluate expression once during the filter initialization, or when the ‘volume’ command is sent

‘frame’

evaluate expression for each incoming frame

Default value is ‘once’.

The volume expression can contain the following parameters.

n

frame number (starting at zero)

nb\_channels

number of channels

nb\_consumed\_samples

number of samples consumed by the filter

nb\_samples

number of samples in the current frame

`pos`

original frame position in the file

`pts`

frame PTS

`sample_rate`

sample rate

`startpts`

PTS at start of stream

`startt`

time at start of stream

`t`

frame time

`tb`

timestamp timebase

`volume`

last set volume value

Note that when `eval` is set to ‘once’ only the *sample\_rate* and *tb* variables are available, all other variables will evaluate to NAN.

### 34.47.1 Commands# TOC

This filter supports the following commands:

`volume`

Modify the volume expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

`replaygain_noclip`

Prevent clipping by limiting the gain applied.

Default value for *replaygain\_noclip* is 1.

### 34.47.2 Examples# TOC

- Halve the input audio volume:

```
volume=volume=0.5
volume=volume=1/2
volume=volume=-6.0206dB
```

In all the above example the named key for *volume* can be omitted, for example like in:

```
volume=0.5
```

- Increase input audio power by 6 decibels using fixed-point precision:

```
volume=volume=6dB:precision=fixed
```

- Fade volume after time 10 with an annihilation period of 5 seconds:

```
volume='if(1t(t,10),1,max(1-(t-10)/5,0))':eval=frame
```

### 34.48 volumedetect# TOC

Detect the volume of the input video.

The filter has no parameters. The input is not modified. Statistics about the volume will be printed in the log when the input stream end is reached.

In particular it will show the mean volume (root mean square), maximum volume (on a per-sample basis), and the beginning of a histogram of the registered volume values (from the maximum value to a cumulated 1/1000 of the samples).

All volumes are in decibels relative to the maximum PCM value.

#### 34.48.1 Examples# TOC

Here is an excerpt of the output:

```
[Parsed_volumedetect_0 0xa23120] mean_volume: -27 dB
[Parsed_volumedetect_0 0xa23120] max_volume: -4 dB
[Parsed_volumedetect_0 0xa23120] histogram_4db: 6
[Parsed_volumedetect_0 0xa23120] histogram_5db: 62
[Parsed_volumedetect_0 0xa23120] histogram_6db: 286
[Parsed_volumedetect_0 0xa23120] histogram_7db: 1042
[Parsed_volumedetect_0 0xa23120] histogram_8db: 2551
[Parsed_volumedetect_0 0xa23120] histogram_9db: 4609
[Parsed_volumedetect_0 0xa23120] histogram_10db: 8409
```

It means that:

- The mean square energy is approximately -27 dB, or  $10^{-2.7}$ .
- The largest sample is at -4 dB, or more precisely between -4 dB and -5 dB.
- There are 6 samples at -4 dB, 62 at -5 dB, 286 at -6 dB, etc.

In other words, raising the volume by +4 dB does not cause any clipping, raising it by +5 dB causes clipping for 6 samples, etc.

## 35 Audio Sources# TOC

Below is a description of the currently available audio sources.

### 35.1 abuffer# TOC

Buffer audio frames, and make them available to the filter chain.

This source is mainly intended for a programmatic use, in particular through the interface defined in `libavfilter/asrc_abuffer.h`.

It accepts the following parameters:

`time_base`

The timebase which will be used for timestamps of submitted frames. It must be either a floating-point number or in *numerator/denominator* form.

`sample_rate`

The sample rate of the incoming audio buffers.

`sample_fmt`

The sample format of the incoming audio buffers. Either a sample format name or its corresponding integer representation from the enum `AVSampleFormat` in `libavutil/samplefmt.h`

`channel_layout`

The channel layout of the incoming audio buffers. Either a channel layout name from `channel_layout_map` in `libavutil/channel_layout.c` or its corresponding integer representation from the `AV_CH_LAYOUT_*` macros in `libavutil/channel_layout.h`

`channels`

The number of channels of the incoming audio buffers. If both *channels* and *channel\_layout* are specified, then they must be consistent.

### 35.1.1 Examples# TOC

```
abuffer=sample_rate=44100:sample_fmt=s16p:channel_layout=stereo
```

will instruct the source to accept planar 16bit signed stereo at 44100Hz. Since the sample format with name "s16p" corresponds to the number 6 and the "stereo" channel layout corresponds to the value 0x3, this is equivalent to:

```
abuffer=sample_rate=44100:sample_fmt=6:channel_layout=0x3
```

### 35.2 aevalsrc# TOC

Generate an audio signal specified by an expression.

This source accepts in input one or more expressions (one for each channel), which are evaluated and used to generate a corresponding audio signal.

This source accepts the following options:

`exprs`

Set the ']'-separated expressions list for each separate channel. In case the `channel_layout` option is not specified, the selected channel layout depends on the number of provided expressions. Otherwise the last specified expression is applied to the remaining output channels.

`channel_layout, c`

Set the channel layout. The number of channels in the specified layout must be equal to the number of specified expressions.

`duration, d`

Set the minimum duration of the sourced audio. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. Note that the resulting duration may be greater than the specified duration, as the generated audio is always cut at the end of a complete frame.

If not specified, or the expressed duration is negative, the audio is supposed to be generated forever.

`nb_samples, n`

Set the number of samples per channel per each output frame, default to 1024.

`sample_rate, s`

Specify the sample rate, default to 44100.

Each expression in *exprs* can contain the following constants:

n

number of the evaluated sample, starting from 0

t

time of the evaluated sample expressed in seconds, starting from 0

s

sample rate

### 35.2.1 Examples# TOC

- Generate silence:

```
aevalsrc=0
```

- Generate a sin signal with frequency of 440 Hz, set sample rate to 8000 Hz:

```
aevalsrc="sin(440*2*PI*t):s=8000"
```

- Generate a two channels signal, specify the channel layout (Front Center + Back Center) explicitly:

```
aevalsrc="sin(420*2*PI*t)|cos(430*2*PI*t):c=FC|BC"
```

- Generate white noise:

```
aevalsrc="-2+random(0)"
```

- Generate an amplitude modulated signal:

```
aevalsrc="sin(10*2*PI*t)*sin(880*2*PI*t)"
```

- Generate 2.5 Hz binaural beats on a 360 Hz carrier:

```
aevalsrc="0.1*sin(2*PI*(360-2.5/2)*t)|0.1*sin(2*PI*(360+2.5/2)*t)"
```

### 35.3 anullsrc# TOC

The null audio source, return unprocessed audio frames. It is mainly useful as a template and to be employed in analysis / debugging tools, or as the source for filters which ignore the input data (for example the sox synth filter).

This source accepts the following options:

`channel_layout, cl`

Specifies the channel layout, and can be either an integer or a string representing a channel layout. The default value of *channel\_layout* is "stereo".



Check the `channel_layout_map` definition in `libavutil/channel_layout.c` for the mapping between strings and channel layout values.

`sample_rate, r`

Specifies the sample rate, and defaults to 44100.

`nb_samples, n`

Set the number of samples per requested frames.

### 35.3.1 Examples# TOC

- Set the sample rate to 48000 Hz and the channel layout to `AV_CH_LAYOUT_MONO`.

```
anullsrc=r=48000:cl=4
```

- Do the same operation with a more obvious syntax:

```
anullsrc=r=48000:cl=mono
```

All the parameters need to be explicitly defined.

## 35.4 flite# TOC

Synthesize a voice utterance using the `libflite` library.

To enable compilation of this filter you need to configure FFmpeg with `--enable-libflite`.

Note that the flite library is not thread-safe.

The filter accepts the following options:

`list_voices`

If set to 1, list the names of the available voices and exit immediately. Default value is 0.

`nb_samples, n`

Set the maximum number of samples per frame. Default value is 512.

`textfile`

Set the filename containing the text to speak.

`text`

Set the text to speak.

`voice, v`

Set the voice to use for the speech synthesis. Default value is `kal`. See also the *list\_voices* option.

### 35.4.1 Examples# TOC

- Read from file `speech.txt`, and synthesize the text using the standard flite voice:

```
flite=textfile=speech.txt
```

- Read the specified text selecting the `slt` voice:

```
flite=text='So fare thee well, poor devil of a Sub-Sub, whose commentator I am':voice=slt
```

- Input text to `ffmpeg`:

```
ffmpeg -f lavfi -i flite=text='So fare thee well, poor devil of a Sub-Sub, whose commentator I am':voice=slt
```

- Make `ffplay` speak the specified text, using `flite` and the `lavfi` device:

```
ffplay -f lavfi flite=text='No more be grieved for which that thou hast done.'
```

For more information about libflite, check: <http://www.speech.cs.cmu.edu/flite/>

## 35.5 sine# TOC

Generate an audio signal made of a sine wave with amplitude 1/8.

The audio signal is bit-exact.

The filter accepts the following options:

`frequency, f`

Set the carrier frequency. Default is 440 Hz.

`beep_factor, b`

Enable a periodic beep every second with frequency *beep\_factor* times the carrier frequency. Default is 0, meaning the beep is disabled.

`sample_rate, r`

Specify the sample rate, default is 44100.

`duration, d`

Specify the duration of the generated audio stream.

`samples_per_frame`

Set the number of samples per output frame, default is 1024.

### 35.5.1 Examples# TOC

- Generate a simple 440 Hz sine wave:

```
sine
```

- Generate a 220 Hz sine wave with a 880 Hz beep each second, for 5 seconds:

```
sine=220:4:d=5  
sine=f=220:b=4:d=5  
sine=frequency=220:beep_factor=4:duration=5
```

## 36 Audio Sinks# TOC

Below is a description of the currently available audio sinks.

### 36.1 abuffersink# TOC

Buffer audio frames, and make them available to the end of filter chain.

This sink is mainly intended for programmatic use, in particular through the interface defined in `libavfilter/buffersink.h` or the options system.

It accepts a pointer to an `AVABufferSinkContext` structure, which defines the incoming buffers' formats, to be passed as the `opaque` parameter to `avfilter_init_filter` for initialization.

### 36.2 anullsink# TOC

Null audio sink; do absolutely nothing with the input audio. It is mainly useful as a template and for use in analysis / debugging tools.

## 37 Video Filters# TOC

When you configure your FFmpeg build, you can disable any of the existing filters using `--disable-filters`. The configure output will show the video filters included in your build.

Below is a description of the currently available video filters.

### 37.1 alphaextract# TOC

Extract the alpha component from the input as a grayscale video. This is especially useful with the *alphamerge* filter.

## 37.2 alphamerge# TOC

Add or replace the alpha component of the primary input with the grayscale value of a second input. This is intended for use with *alphaextract* to allow the transmission or storage of frame sequences that have alpha in a format that doesn't support an alpha channel.

For example, to reconstruct full frames from a normal YUV-encoded video and a separate video created with *alphaextract*, you might use:

```
movie=in_alpha.mkv [alpha]; [in][alpha] alphamerge [out]
```

Since this filter is designed for reconstruction, it operates on frame sequences without considering timestamps, and terminates when either input reaches end of stream. This will cause problems if your encoding pipeline drops frames. If you're trying to apply an image as an overlay to a video stream, consider the *overlay* filter instead.

## 37.3 ass# TOC

Same as the subtitles filter, except that it doesn't require libavcodec and libavformat to work. On the other hand, it is limited to ASS (Advanced Substation Alpha) subtitles files.

This filter accepts the following option in addition to the common options from the subtitles filter:

shaping

Set the shaping engine

Available values are:

'auto'

The default libass shaping engine, which is the best available.

'simple'

Fast, font-agnostic shaper that can do only substitutions

'complex'

Slower shaper using OpenType for substitutions and positioning

The default is `auto`.

## 37.4 atadenoise# TOC

Apply an Adaptive Temporal Averaging Denoiser to the video input.

The filter accepts the following options:

0a

Set threshold A for 1st plane. Default is 0.02. Valid range is 0 to 0.3.

0b

Set threshold B for 1st plane. Default is 0.04. Valid range is 0 to 5.

1a

Set threshold A for 2nd plane. Default is 0.02. Valid range is 0 to 0.3.

1b

Set threshold B for 2nd plane. Default is 0.04. Valid range is 0 to 5.

2a

Set threshold A for 3rd plane. Default is 0.02. Valid range is 0 to 0.3.

2b

Set threshold B for 3rd plane. Default is 0.04. Valid range is 0 to 5.

Threshold A is designed to react on abrupt changes in the input signal and threshold B is designed to react on continuous changes in the input signal.

s

Set number of frames filter will use for averaging. Default is 33. Must be odd number in range [5, 129].

## 37.5 bbox# TOC

Compute the bounding box for the non-black pixels in the input frame luminance plane.

This filter computes the bounding box containing all the pixels with a luminance value greater than the minimum allowed value. The parameters describing the bounding box are printed on the filter log.

The filter accepts the following option:

min\_val

Set the minimal luminance value. Default is 16.

## 37.6 blackdetect# TOC

Detect video intervals that are (almost) completely black. Can be useful to detect chapter transitions, commercials, or invalid recordings. Output lines contains the time for the start, end and duration of the detected black interval expressed in seconds.

In order to display the output lines, you need to set the loglevel at least to the AV\_LOG\_INFO value.

The filter accepts the following options:

`black_min_duration, d`

Set the minimum detected black duration expressed in seconds. It must be a non-negative floating point number.

Default value is 2.0.

`picture_black_ratio_th, pic_th`

Set the threshold for considering a picture "black". Express the minimum value for the ratio:

*`nb_black_pixels / nb_pixels`*

for which a picture is considered black. Default value is 0.98.

`pixel_black_th, pix_th`

Set the threshold for considering a pixel "black".

The threshold expresses the maximum pixel luminance value for which a pixel is considered "black". The provided value is scaled according to the following equation:

*`absolute_threshold = luminance_minimum_value + pixel_black_th * luminance_range_size`*

*`luminance_range_size`* and *`luminance_minimum_value`* depend on the input video format, the range is [0-255] for YUV full-range formats and [16-235] for YUV non full-range formats.

Default value is 0.10.

The following example sets the maximum pixel threshold to the minimum value, and detects only black intervals of 2 or more seconds:

```
blackdetect=d=2:pix_th=0.00
```

## 37.7 blackframe# TOC

Detect frames that are (almost) completely black. Can be useful to detect chapter transitions or commercials. Output lines consist of the frame number of the detected frame, the percentage of blackness, the position in the file if known or -1 and the timestamp in seconds.

In order to display the output lines, you need to set the loglevel at least to the AV\_LOG\_INFO value.

It accepts the following parameters:

`amount`

The percentage of the pixels that have to be below the threshold; it defaults to 98.

`threshold, thresh`

The threshold below which a pixel value is considered black; it defaults to 32.

## 37.8 `blend, tblend`# TOC

Blend two video frames into each other.

The `blend` filter takes two input streams and outputs one stream, the first input is the "top" layer and second input is "bottom" layer. Output terminates when shortest input terminates.

The `tblend` (time blend) filter takes two consecutive frames from one single stream, and outputs the result obtained by blending the new frame on top of the old frame.

A description of the accepted options follows.

`c0_mode`

`c1_mode`

`c2_mode`

`c3_mode`

`all_mode`

Set blend mode for specific pixel component or all pixel components in case of *all\_mode*. Default value is `normal`.

Available values for component modes are:

`'addition'`

`'and'`

`'average'`

`'burn'`

`'darken'`

`'difference'`

`'difference128'`

`'divide'`

`'dodge'`

`'exclusion'`

`'glow'`

`'hardlight'`

'hardmix'  
'lighten'  
'linearlight'  
'multiply'  
'negation'  
'normal'  
'or'  
'overlay'  
'phoenix'  
'pinlight'  
'reflect'  
'screen'  
'softlight'  
'subtract'  
'vividlight'  
'xor'  
c0\_opacity  
c1\_opacity  
c2\_opacity  
c3\_opacity  
all\_opacity

Set blend opacity for specific pixel component or all pixel components in case of *all\_opacity*. Only used in combination with pixel component blend modes.

c0\_expr  
c1\_expr  
c2\_expr  
c3\_expr  
all\_expr

Set blend expression for specific pixel component or all pixel components in case of *all\_expr*. Note that related mode options will be ignored if those are set.

The expressions can use the following variables:

N

The sequential number of the filtered frame, starting from 0.

X

Y

the coordinates of the current sample

W



H

the width and height of currently filtered plane

SW

SH

Width and height scale depending on the currently filtered plane. It is the ratio between the corresponding luma plane number of pixels and the current plane ones. E.g. for YUV4:2:0 the values are 1, 1 for the luma plane, and 0.5, 0.5 for chroma planes.

T

Time of the current frame, expressed in seconds.

TOP, A

Value of pixel component at current location for first video frame (top layer).

BOTTOM, B

Value of pixel component at current location for second video frame (bottom layer).

shortest

Force termination when the shortest input terminates. Default is 0. This option is only defined for the blend filter.

repeatlast

Continue applying the last bottom frame after the end of the stream. A value of 0 disable the filter after the last frame of the bottom layer is reached. Default is 1. This option is only defined for the blend filter.

### 37.8.1 Examples# TOC

- Apply transition from bottom layer to top layer in first 10 seconds:

```
blend=all_expr='A*(if(gte(T,10),1,T/10))+B*(1-(if(gte(T,10),1,T/10)))'
```

- Apply 1x1 checkerboard effect:

```
blend=all_expr='if(eq(mod(X,2),mod(Y,2)),A,B)'
```

- Apply uncover left effect:

```
blend=all_expr='if(gte(N*SW+X,W),A,B)'
```

- Apply uncover down effect:

```
blend=all_expr='if(gte(Y-N*SH,0),A,B)'
```

- Apply uncover up-left effect:

```
blend=all_expr='if(gte(T*SH*40+Y,H)*gte((T*40*SW+X)*W/H,W),A,B)'
```

- Display differences between the current and the previous frame:

```
tblend=all_mode=difference128
```

## 37.9 boxblur# TOC

Apply a boxblur algorithm to the input video.

It accepts the following parameters:

```
luma_radius, lr  
luma_power, lp  
chroma_radius, cr  
chroma_power, cp  
alpha_radius, ar  
alpha_power, ap
```

A description of the accepted options follows.

```
luma_radius, lr  
chroma_radius, cr  
alpha_radius, ar
```

Set an expression for the box radius in pixels used for blurring the corresponding input plane.

The radius value must be a non-negative number, and must not be greater than the value of the expression  $\min(w, h) / 2$  for the luma and alpha planes, and of  $\min(cw, ch) / 2$  for the chroma planes.

Default value for `luma_radius` is "2". If not specified, `chroma_radius` and `alpha_radius` default to the corresponding value set for `luma_radius`.

The expressions can contain the following constants:

```
w  
h
```

The input width and height in pixels.

```
cw  
ch
```

The input chroma image width and height in pixels.

`hsub`  
`vsub`

The horizontal and vertical chroma subsample values. For example, for the pixel format "yuv422p", *hsub* is 2 and *vsub* is 1.

`luma_power, lp`  
`chroma_power, cp`  
`alpha_power, ap`

Specify how many times the boxblur filter is applied to the corresponding plane.

Default value for `luma_power` is 2. If not specified, `chroma_power` and `alpha_power` default to the corresponding value set for `luma_power`.

A value of 0 will disable the effect.

### 37.9.1 Examples# TOC

- Apply a boxblur filter with the luma, chroma, and alpha radii set to 2:

```
boxblur=luma_radius=2:luma_power=1  
boxblur=2:1
```

- Set the luma radius to 2, and alpha and chroma radius to 0:

```
boxblur=2:1:cr=0:ar=0
```

- Set the luma and chroma radii to a fraction of the video dimension:

```
boxblur=luma_radius=min(h\,w)/10:luma_power=1:chroma_radius=min(cw\,ch)/10:chroma_power=1
```

### 37.10 codecview# TOC

Visualize information exported by some codecs.

Some codecs can export information through frames using side-data or other means. For example, some MPEG based codecs export motion vectors through the *export\_mvs* flag in the codec *flags2* option.

The filter accepts the following option:

`mv`

Set motion vectors to visualize.

Available flags for *mv* are:

‘pf’

forward predicted MVs of P-frames

‘bf’

forward predicted MVs of B-frames

‘bb’

backward predicted MVs of B-frames

### 37.10.1 Examples# TOC

- Visualizes multi-directionals MVs from P and B-Frames using `ffplay`:

```
ffplay -flags2 +export_mvs input.mpg -vf codecview=mv=pf+bf+bb
```

### 37.11 colorbalance# TOC

Modify intensity of primary colors (red, green and blue) of input frames.

The filter allows an input frame to be adjusted in the shadows, midtones or highlights regions for the red-cyan, green-magenta or blue-yellow balance.

A positive adjustment value shifts the balance towards the primary color, a negative value towards the complementary color.

The filter accepts the following options:

rs  
gs  
bs

Adjust red, green and blue shadows (darkest pixels).

rm  
gm  
bm

Adjust red, green and blue midtones (medium pixels).

rh  
gh  
bh

Adjust red, green and blue highlights (brightest pixels).

Allowed ranges for options are  $[-1.0, 1.0]$ . Defaults are 0.

### 37.11.1 Examples# TOC

- Add red color cast to shadows:

```
colorbalance=rs=.3
```

## 37.12 colorkey# TOC

RGB colorspace color keying.

The filter accepts the following options:

**color**

The color which will be replaced with transparency.

**similarity**

Similarity percentage with the key color.

0.01 matches only the exact key color, while 1.0 matches everything.

**blend**

Blend percentage.

0.0 makes pixels either fully transparent, or not transparent at all.

Higher values result in semi-transparent pixels, with a higher transparency the more similar the pixels color is to the key color.

### 37.12.1 Examples# TOC

- Make every green pixel in the input image transparent:

```
ffmpeg -i input.png -vf colorkey=green out.png
```

- Overlay a greenscreen-video on top of a static background image.

```
ffmpeg -i background.png -i video.mp4 -filter_complex "[1:v]colorkey=0x3BBD1E:0.3:0.2[ckout];[0:v][ckout]overlay[out]" -map "[out]" output.flv
```

## 37.13 colorlevels# TOC

Adjust video input frames using levels.

The filter accepts the following options:

rimin  
gimin  
bimin  
aimin

Adjust red, green, blue and alpha input black point. Allowed ranges for options are  $[-1.0, 1.0]$ . Defaults are 0.

rimax  
gimax  
bimax  
aimax

Adjust red, green, blue and alpha input white point. Allowed ranges for options are  $[-1.0, 1.0]$ . Defaults are 1.

Input levels are used to lighten highlights (bright tones), darken shadows (dark tones), change the balance of bright and dark tones.

romin  
gomin  
bomin  
aomin

Adjust red, green, blue and alpha output black point. Allowed ranges for options are  $[0, 1.0]$ . Defaults are 0.

romax  
gomax  
bomax  
aomax

Adjust red, green, blue and alpha output white point. Allowed ranges for options are  $[0, 1.0]$ . Defaults are 1.

Output levels allows manual selection of a constrained output level range.

### 37.13.1 Examples# TOC

- Make video output darker:

```
colorlevels=rimin=0.058:gimin=0.058:bimin=0.058
```

- Increase contrast:

```
colorlevels=rimin=0.039:gimin=0.039:bimin=0.039:rimax=0.96:gimax=0.96:bimax=0.96
```

- Make video output lighter:

```
colorlevels=rimax=0.902:gimax=0.902:bimax=0.902
```

- Increase brightness:

```
colorlevels=romin=0.5:gomin=0.5:bomin=0.5
```

## 37.14 colorchannelmixer# TOC

Adjust video input frames by re-mixing color channels.

This filter modifies a color channel by adding the values associated to the other channels of the same pixels. For example if the value to modify is red, the output value will be:

$$red = red * rr + blue * rb + green * rg + alpha * ra$$

The filter accepts the following options:

rr  
rg  
rb  
ra

Adjust contribution of input red, green, blue and alpha channels for output red channel. Default is 1 for *rr*, and 0 for *rg*, *rb* and *ra*.

gr  
gg  
gb  
ga

Adjust contribution of input red, green, blue and alpha channels for output green channel. Default is 1 for *gg*, and 0 for *gr*, *gb* and *ga*.

br  
bg  
bb  
ba

Adjust contribution of input red, green, blue and alpha channels for output blue channel. Default is 1 for *bb*, and 0 for *br*, *bg* and *ba*.

ar  
ag  
ab  
aa

Adjust contribution of input red, green, blue and alpha channels for output alpha channel. Default is 1 for *aa*, and 0 for *ar*, *ag* and *ab*.

Allowed ranges for options are  $[-2.0, 2.0]$ .

### 37.14.1 Examples# TOC

- Convert source to grayscale:

```
colorchannelmixer=.3:.4:.3:0:.3:.4:.3:0:.3:.4:.3
```

- Simulate sepia tones:

```
colorchannelmixer=.393:.769:.189:0:.349:.686:.168:0:.272:.534:.131
```

### 37.15 colormatrix# TOC

Convert color matrix.

The filter accepts the following options:

```
src  
dst
```

Specify the source and destination color matrix. Both values must be specified.

The accepted values are:

‘bt709’

BT.709

‘bt601’

BT.601

‘smpte240m’

SMPTE-240M

‘fcc’

FCC

For example to convert from BT.601 to SMPTE-240M, use the command:

```
colormatrix=bt601:smpte240m
```



## 37.16 copy# TOC

Copy the input source unchanged to the output. This is mainly useful for testing purposes.

## 37.17 crop# TOC

Crop the input video to given dimensions.

It accepts the following parameters:

`w, out_w`

The width of the output video. It defaults to `iw`. This expression is evaluated only once during the filter configuration, or when the `'w'` or `'out_w'` command is sent.

`h, out_h`

The height of the output video. It defaults to `ih`. This expression is evaluated only once during the filter configuration, or when the `'h'` or `'out_h'` command is sent.

`x`

The horizontal position, in the input video, of the left edge of the output video. It defaults to  $(in\_w - out\_w) / 2$ . This expression is evaluated per-frame.

`y`

The vertical position, in the input video, of the top edge of the output video. It defaults to  $(in\_h - out\_h) / 2$ . This expression is evaluated per-frame.

`keep_aspect`

If set to 1 will force the output display aspect ratio to be the same of the input, by changing the output sample aspect ratio. It defaults to 0.

The `out_w`, `out_h`, `x`, `y` parameters are expressions containing the following constants:

`x`

`y`

The computed values for `x` and `y`. They are evaluated for each new frame.

`in_w`

`in_h`

The input width and height.

`iw`  
`ih`

These are the same as *in\_w* and *in\_h*.

`out_w`  
`out_h`

The output (cropped) width and height.

`ow`  
`oh`

These are the same as *out\_w* and *out\_h*.

`a`

same as *iw / ih*

`sar`

input sample aspect ratio

`dar`

input display aspect ratio, it is the same as  $(iw / ih) * sar$

`hsub`  
`vsub`

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

`n`

The number of the input frame, starting from 0.

`pos`

the position in the file of the input frame, NAN if unknown

`t`

The timestamp expressed in seconds. It's NAN if the input timestamp is unknown.

The expression for *out\_w* may depend on the value of *out\_h*, and the expression for *out\_h* may depend on *out\_w*, but they cannot depend on *x* and *y*, as *x* and *y* are evaluated after *out\_w* and *out\_h*.

The  $x$  and  $y$  parameters specify the expressions for the position of the top-left corner of the output (non-cropped) area. They are evaluated for each frame. If the evaluated value is not valid, it is approximated to the nearest valid value.

The expression for  $x$  may depend on  $y$ , and the expression for  $y$  may depend on  $x$ .

### 37.17.1 Examples# TOC

- Crop area with size 100x100 at position (12,34).

```
crop=100:100:12:34
```

Using named options, the example above becomes:

```
crop=w=100:h=100:x=12:y=34
```

- Crop the central input area with size 100x100:

```
crop=100:100
```

- Crop the central input area with size 2/3 of the input video:

```
crop=2/3*in_w:2/3*in_h
```

- Crop the input video central square:

```
crop=out_w=in_h  
crop=in_h
```

- Delimit the rectangle with the top-left corner placed at position 100:100 and the right-bottom corner corresponding to the right-bottom corner of the input image.

```
crop=in_w-100:in_h-100:100:100
```

- Crop 10 pixels from the left and right borders, and 20 pixels from the top and bottom borders

```
crop=in_w-2*10:in_h-2*20
```

- Keep only the bottom right quarter of the input image:

```
crop=in_w/2:in_h/2:in_w/2:in_h/2
```

- Crop height for getting Greek harmony:

```
crop=in_w:1/PHI*in_w
```

- Apply trembling effect:

```
crop=in_w/2:in_h/2:(in_w-out_w)/2+((in_w-out_w)/2)*sin(n/10):(in_h-out_h)/2+((in_h-out_h)/2)*sin(n/7)
```

- Apply erratic camera effect depending on timestamp:

```
crop=in_w/2:in_h/2:(in_w-out_w)/2+((in_w-out_w)/2)*sin(t*10):(in_h-out_h)/2 +((in_h-out_h)/2)*sin(t*13)"
```

- Set x depending on the value of y:

```
crop=in_w/2:in_h/2:y:10+10*sin(n/10)
```

### 37.17.2 Commands# TOC

This filter supports the following commands:

```
w, out_w  
h, out_h  
x  
y
```

Set width/height of the output video and the horizontal/vertical position in the input video. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.18 cropdetect# TOC

Auto-detect the crop size.

It calculates the necessary cropping parameters and prints the recommended parameters via the logging system. The detected dimensions correspond to the non-black area of the input video.

It accepts the following parameters:

```
limit
```

Set higher black value threshold, which can be optionally specified from nothing (0) to everything (255 for 8bit based formats). An intensity value greater to the set value is considered non-black. It defaults to 24. You can also specify a value between 0.0 and 1.0 which will be scaled depending on the bitdepth of the pixel format.

```
round
```

The value which the width/height should be divisible by. It defaults to 16. The offset is automatically adjusted to center the video. Use 2 to get only even dimensions (needed for 4:2:2 video). 16 is best when encoding to most video codecs.

```
reset_count, reset
```

Set the counter that determines after how many frames cropdetect will reset the previously detected largest video area and start over to detect the current optimal crop area. Default value is 0.

This can be useful when channel logos distort the video area. 0 indicates 'never reset', and returns the largest area encountered during playback.

## 37.19 curves# TOC

Apply color adjustments using curves.

This filter is similar to the Adobe Photoshop and GIMP curves tools. Each component (red, green and blue) has its values defined by  $N$  key points tied from each other using a smooth curve. The x-axis represents the pixel values from the input frame, and the y-axis the new pixel values to be set for the output frame.

By default, a component curve is defined by the two points  $(0;0)$  and  $(1;1)$ . This creates a straight line where each original pixel value is "adjusted" to its own value, which means no change to the image.

The filter allows you to redefine these two points and add some more. A new curve (using a natural cubic spline interpolation) will be defined to pass smoothly through all these new coordinates. The new defined points need to be strictly increasing over the x-axis, and their x and y values must be in the  $[0;1]$  interval. If the computed curves happened to go outside the vector spaces, the values will be clipped accordingly.

If there is no key point defined in  $x=0$ , the filter will automatically insert a  $(0;0)$  point. In the same way, if there is no key point defined in  $x=1$ , the filter will automatically insert a  $(1;1)$  point.

The filter accepts the following options:

`preset`

Select one of the available color presets. This option can be used in addition to the `r`, `g`, `b` parameters; in this case, the later options take priority on the preset values. Available presets are:

```
'none'
'color_negative'
'cross_process'
'darker'
'increase_contrast'
'lighter'
'linear_contrast'
'medium_contrast'
'negative'
'strong_contrast'
'vintage'
```

Default is none.

`master, m`

Set the master key points. These points will define a second pass mapping. It is sometimes called a "luminance" or "value" mapping. It can be used with `r`, `g`, `b` or `all` since it acts like a post-processing LUT.

`red, r`

Set the key points for the red component.

`green, g`

Set the key points for the green component.

`blue, b`

Set the key points for the blue component.

`all`

Set the key points for all components (not including master). Can be used in addition to the other key points component options. In this case, the unset component(s) will fallback on this `all` setting.

`psfile`

Specify a Photoshop curves file (`.asv`) to import the settings from.

To avoid some filtergraph syntax conflicts, each key points list need to be defined using the following syntax: `x0/y0 x1/y1 x2/y2 ....`

### 37.19.1 Examples# TOC

- Increase slightly the middle level of blue:

```
curves=blue='0.5/0.58'
```

- Vintage effect:

```
curves=r='0/0.11 .42/.51 1/0.95':g='0.50/0.48':b='0/0.22 .49/.44 1/0.8'
```

Here we obtain the following coordinates for each components:

*red*

```
(0;0.11) (0.42;0.51) (1;0.95)
```

*green*

```
(0;0) (0.50;0.48) (1;1)
```

*blue*

```
(0;0.22) (0.49;0.44) (1;0.80)
```

- The previous example can also be achieved with the associated built-in preset:

```
curves=preset=vintage
```

- Or simply:

```
curves=vintage
```

- Use a Photoshop preset and redefine the points of the green component:

```
curves=psfile='MyCurvesPresets/purple.asv':green='0.45/0.53'
```

## 37.20 dctdnoiz# TOC

Denoise frames using 2D DCT (frequency domain filtering).

This filter is not designed for real time.

The filter accepts the following options:

`sigma, s`

Set the noise sigma constant.

This *sigma* defines a hard threshold of  $3 * \text{sigma}$ ; every DCT coefficient (absolute value) below this threshold will be dropped.

If you need a more advanced filtering, see `expr`.

Default is 0.

`overlap`

Set number overlapping pixels for each block. Since the filter can be slow, you may want to reduce this value, at the cost of a less effective filter and the risk of various artefacts.

If the overlapping value doesn't permit processing the whole input width or height, a warning will be displayed and according borders won't be denoised.

Default value is *blocksize-1*, which is the best possible setting.

`expr, e`

Set the coefficient factor expression.

For each coefficient of a DCT block, this expression will be evaluated as a multiplier value for the coefficient.

If this option is set, the `sigma` option will be ignored.

The absolute value of the coefficient can be accessed through the `c` variable.

`n`

Set the *blocksize* using the number of bits.  $1 < n$  defines the *blocksize*, which is the width and height of the processed blocks.

The default value is 3 (8x8) and can be raised to 4 for a *blocksize* of 16x16. Note that changing this setting has huge consequences on the speed processing. Also, a larger block size does not necessarily mean a better de-noising.

### 37.20.1 Examples# TOC

Apply a denoise with a `sigma` of 4.5:

```
dctdnoiz=4.5
```

The same operation can be achieved using the expression system:

```
dctdnoiz=e='gte(c, 4.5*3)'
```

Violent denoise using a block size of 16x16:

```
dctdnoiz=15:n=4
```

### 37.21 deband# TOC

Remove banding artifacts from input video. It works by replacing banded pixels with average value of referenced pixels.

The filter accepts the following options:

```
1thr  
2thr  
3thr  
4thr
```

Set banding detection threshold for each plane. Default is 0.02. Valid range is 0.00003 to 0.5. If difference between current pixel and reference pixel is less than threshold, it will be considered as banded.

`range, r`

Banding detection range in pixels. Default is 16. If positive, random number in range 0 to set value will be used. If negative, exact absolute value will be used. The range defines square of four pixels around current pixel.



`direction, d`

Set direction in radians from which four pixel will be compared. If positive, random direction from 0 to set direction will be picked. If negative, exact of absolute value will be picked. For example direction 0,  $-\pi$  or  $-2\pi$  radians will pick only pixels on same row and  $-\pi/2$  will pick only pixels on same column.

`blur`

If enabled, current pixel is compared with average value of all four surrounding pixels. The default is enabled. If disabled current pixel is compared with all four surrounding pixels. The pixel is considered banded if only all four differences with surrounding pixels are less than threshold.

## 37.22 decimate# TOC

Drop duplicated frames at regular intervals.

The filter accepts the following options:

`cycle`

Set the number of frames from which one will be dropped. Setting this to  $N$  means one frame in every batch of  $N$  frames will be dropped. Default is 5.

`dupthresh`

Set the threshold for duplicate detection. If the difference metric for a frame is less than or equal to this value, then it is declared as duplicate. Default is 1.1

`scthresh`

Set scene change threshold. Default is 15.

`blockx`

`blocky`

Set the size of the x and y-axis blocks used during metric calculations. Larger blocks give better noise suppression, but also give worse detection of small movements. Must be a power of two. Default is 32.

`ppsrc`

Mark main input as a pre-processed input and activate clean source input stream. This allows the input to be pre-processed with various filters to help the metrics calculation while keeping the frame selection lossless. When set to 1, the first stream is for the pre-processed input, and the second stream is the clean source from where the kept frames are chosen. Default is 0.

chroma

Set whether or not chroma is considered in the metric calculations. Default is 1.

### 37.23 deflate# TOC

Apply deflate effect to the video.

This filter replaces the pixel by the local(3x3) average by taking into account only values lower than the pixel.

It accepts the following options:

threshold0  
threshold1  
threshold2  
threshold3

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

### 37.24 dejudder# TOC

Remove judder produced by partially interlaced telecined content.

Judder can be introduced, for instance, by pullup filter. If the original source was partially telecined content then the output of `pullup`, `dejudder` will have a variable frame rate. May change the recorded frame rate of the container. Aside from that change, this filter will not affect constant frame rate video.

The option available in this filter is:

cycle

Specify the length of the window over which the judder repeats.

Accepts any integer greater than 1. Useful values are:

‘4’

If the original was telecined from 24 to 30 fps (Film to NTSC).

‘5’

If the original was telecined from 25 to 30 fps (PAL to NTSC).

‘20’

If a mixture of the two.

The default is '4'.

## 37.25 delogo# TOC

Suppress a TV station logo by a simple interpolation of the surrounding pixels. Just set a rectangle covering the logo and watch it disappear (and sometimes something even uglier appear - your mileage may vary).

It accepts the following parameters:

$x$   
 $y$

Specify the top left corner coordinates of the logo. They must be specified.

$w$   
 $h$

Specify the width and height of the logo to clear. They must be specified.

$band$ ,  $t$

Specify the thickness of the fuzzy edge of the rectangle (added to  $w$  and  $h$ ). The default value is 4.

$show$

When set to 1, a green rectangle is drawn on the screen to simplify finding the right  $x$ ,  $y$ ,  $w$ , and  $h$  parameters. The default value is 0.

The rectangle is drawn on the outermost pixels which will be (partly) replaced with interpolated values. The values of the next pixels immediately outside this rectangle in each direction will be used to compute the interpolated pixel values inside the rectangle.

### 37.25.1 Examples# TOC

- Set a rectangle covering the area with top left corner coordinates 0,0 and size 100x77, and a band of size 10:

```
delogo=x=0:y=0:w=100:h=77:band=10
```

## 37.26 deshake# TOC

Attempt to fix small changes in horizontal and/or vertical shift. This filter helps remove camera shake from hand-holding a camera, bumping a tripod, moving on a vehicle, etc.

The filter accepts the following options:

x  
y  
w  
h

Specify a rectangular area where to limit the search for motion vectors. If desired the search for motion vectors can be limited to a rectangular area of the frame defined by its top left corner, width and height. These parameters have the same meaning as the drawbox filter which can be used to visualise the position of the bounding box.

This is useful when simultaneous movement of subjects within the frame might be confused for camera motion by the motion vector search.

If any or all of  $x$ ,  $y$ ,  $w$  and  $h$  are set to -1 then the full frame is used. This allows later options to be set without specifying the bounding box for the motion vector search.

Default - search the whole frame.

rx  
ry

Specify the maximum extent of movement in x and y directions in the range 0-64 pixels. Default 16.

edge

Specify how to generate pixels to fill blanks at the edge of the frame. Available values are:

`'blank, 0'`

Fill zeroes at blank locations

`'original, 1'`

Original image at blank locations

`'clamp, 2'`

Extruded edge value at blank locations

`'mirror, 3'`

Mirrored edge at blank locations

Default value is `'mirror'`.

blocksize

Specify the blocksize to use for motion search. Range 4-128 pixels, default 8.

contrast

Specify the contrast threshold for blocks. Only blocks with more than the specified contrast (difference between darkest and lightest pixels) will be considered. Range 1-255, default 125.

search

Specify the search strategy. Available values are:

`'exhaustive, 0'`

Set exhaustive search

`'less, 1'`

Set less exhaustive search.

Default value is `'exhaustive'`.

filename

If set then a detailed log of the motion search is written to the specified file.

opencl

If set to 1, specify using OpenCL capabilities, only available if FFmpeg was configured with `--enable-opencl`. Default value is 0.

## 37.27 detelecine# TOC

Apply an exact inverse of the telecine operation. It requires a predefined pattern specified using the pattern option which must be the same as that passed to the telecine filter.

This filter accepts the following options:

first\_field

`'top, t'`

top field first

`'bottom, b'`

bottom field first The default value is top.

pattern

A string of numbers representing the pulldown pattern you wish to apply. The default value is 23.

`start_frame`

A number representing position of the first frame with respect to the telecine pattern. This is to be used if the stream is cut. The default value is 0.

## 37.28 dilation# TOC

Apply dilation effect to the video.

This filter replaces the pixel by the local(3x3) maximum.

It accepts the following options:

`threshold0`  
`threshold1`  
`threshold2`  
`threshold3`

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

`coordinates`

Flag which specifies the pixel to refer to. Default is 255 i.e. all eight pixels are used.

Flags to local 3x3 coordinates maps like this:

1 2 3 4 5 6 7 8

## 37.29 drawbox# TOC

Draw a colored box on the input image.

It accepts the following parameters:

`x`  
`y`

The expressions which specify the top left corner coordinates of the box. It defaults to 0.

`width, w`  
`height, h`

The expressions which specify the width and height of the box; if 0 they are interpreted as the input width and height. It defaults to 0.

`color, c`

Specify the color of the box to write. For the general syntax of this option, check the "Color" section in the ffmpeg-utils manual. If the special value `invert` is used, the box edge color is the same as the video with inverted luma.

`thickness, t`

The expression which sets the thickness of the box edge. Default value is 3.

See below for the list of accepted constants.

The parameters for  $x$ ,  $y$ ,  $w$  and  $h$  and  $t$  are expressions containing the following constants:

`dar`

The input display aspect ratio, it is the same as  $(w / h) * sar$ .

`hsub`

`vsub`

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

`in_h, ih`

`in_w, iw`

The input width and height.

`sar`

The input sample aspect ratio.

`x`

`y`

The x and y offset coordinates where the box is drawn.

`w`

`h`

The width and height of the drawn box.

`t`

The thickness of the drawn box.

These constants allow the  $x$ ,  $y$ ,  $w$ ,  $h$  and  $t$  expressions to refer to each other, so you may for example specify  $y=x/dar$  or  $h=w/dar$ .

### 37.29.1 Examples# TOC

- Draw a black box around the edge of the input image:

```
drawbox
```

- Draw a box with color red and an opacity of 50%:

```
drawbox=10:20:200:60:red@0.5
```

The previous example can be specified as:

```
drawbox=x=10:y=20:w=200:h=60:color=red@0.5
```

- Fill the box with pink color:

```
drawbox=x=10:y=10:w=100:h=100:color=pink@0.5:t=max
```

- Draw a 2-pixel red 2.40:1 mask:

```
drawbox=x=-t:y=0.5*(ih-iw/2.4)-t:w=iw+t*2:h=iw/2.4+t*2:t=2:c=red
```

### 37.30 drawgraph, adrawgraph# TOC

Draw a graph using input video or audio metadata.

It accepts the following parameters:

m1

Set 1st frame metadata key from which metadata values will be used to draw a graph.

fg1

Set 1st foreground color expression.

m2

Set 2nd frame metadata key from which metadata values will be used to draw a graph.

fg2

Set 2nd foreground color expression.

m3

Set 3rd frame metadata key from which metadata values will be used to draw a graph.

fg3



Set 3rd foreground color expression.

m4

Set 4th frame metadata key from which metadata values will be used to draw a graph.

fg4

Set 4th foreground color expression.

min

Set minimal value of metadata value.

max

Set maximal value of metadata value.

bg

Set graph background color. Default is white.

mode

Set graph mode.

Available values for mode is:

'bar'  
'dot'  
'line'

Default is line.

slide

Set slide mode.

Available values for slide is:

'frame'

Draw new frame when right border is reached.

'replace'

Replace old columns with new ones.

`'scroll'`

Scroll from right to left.

`'rscroll'`

Scroll from left to right.

Default is `frame`.

`size`

Set size of graph video. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. The default value is 900x256.

The foreground color expressions can use the following variables:

`MIN`

Minimal value of metadata value.

`MAX`

Maximal value of metadata value.

`VAL`

Current metadata key value.

The color is defined as 0xAABBGRR.

Example using metadata from signalstats filter:

```
signalstats,drawgraph=lavfi.signalstats.YAVG:min=0:max=255
```

Example using metadata from ebur128 filter:

```
ebur128=metadata=1,adrawgraph=lavfi.r128.M:min=-120:max=5
```

## 37.31 drawgrid# TOC

Draw a grid on the input image.

It accepts the following parameters:

`x`

`y`

The expressions which specify the coordinates of some point of grid intersection (meant to configure offset). Both default to 0.

width, *w*  
height, *h*

The expressions which specify the width and height of the grid cell, if 0 they are interpreted as the input width and height, respectively, minus *thickness*, so image gets framed. Default to 0.

color, *c*

Specify the color of the grid. For the general syntax of this option, check the "Color" section in the *ffmpeg-utils* manual. If the special value *invert* is used, the grid color is the same as the video with inverted luma.

thickness, *t*

The expression which sets the thickness of the grid line. Default value is 1.

See below for the list of accepted constants.

The parameters for *x*, *y*, *w* and *h* and *t* are expressions containing the following constants:

*dar*

The input display aspect ratio, it is the same as  $(w / h) * sar$ .

*hsub*  
*vsub*

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

*in\_h*, *ih*  
*in\_w*, *iw*

The input grid cell width and height.

*sar*

The input sample aspect ratio.

*x*  
*y*

The x and y coordinates of some point of grid intersection (meant to configure offset).

*w*

*h*

The width and height of the drawn cell.

*t*

The thickness of the drawn cell.

These constants allow the *x*, *y*, *w*, *h* and *t* expressions to refer to each other, so you may for example specify *y*=*x*/*dar* or *h*=*w*/*dar*.

### 37.31.1 Examples# TOC

- Draw a grid with cell 100x100 pixels, thickness 2 pixels, with color red and an opacity of 50%:

```
drawgrid=width=100:height=100:thickness=2:color=red@0.5
```

- Draw a white 3x3 grid with an opacity of 50%:

```
drawgrid=w=iw/3:h=ih/3:t=2:c=white@0.5
```

### 37.32 drawtext# TOC

Draw a text string or text from a specified file on top of a video, using the libfreetype library.

To enable compilation of this filter, you need to configure FFmpeg with `--enable-libfreetype`. To enable default font fallback and the *font* option you need to configure FFmpeg with `--enable-libfontconfig`. To enable the *text\_shaping* option, you need to configure FFmpeg with `--enable-libfribidi`.

#### 37.32.1 Syntax# TOC

It accepts the following parameters:

*box*

Used to draw a box around text using the background color. The value must be either 1 (enable) or 0 (disable). The default value of *box* is 0.

*boxborderw*

Set the width of the border to be drawn around the box using *boxcolor*. The default value of *boxborderw* is 0.

*boxcolor*

The color to be used for drawing box around text. For the syntax of this option, check the "Color" section in the ffmpeg-utils manual.

The default value of *boxcolor* is "white".

`borderw`

Set the width of the border to be drawn around the text using *bordercolor*. The default value of *borderw* is 0.

`bordercolor`

Set the color to be used for drawing border around text. For the syntax of this option, check the "Color" section in the ffmpeg-utils manual.

The default value of *bordercolor* is "black".

`expansion`

Select how the *text* is expanded. Can be either none, `strftime` (deprecated) or `normal` (default). See the Text expansion section below for details.

`fix_bounds`

If true, check and fix text coords to avoid clipping.

`fontcolor`

The color to be used for drawing fonts. For the syntax of this option, check the "Color" section in the ffmpeg-utils manual.

The default value of *fontcolor* is "black".

`fontcolor_expr`

String which is expanded the same way as *text* to obtain dynamic *fontcolor* value. By default this option has empty value and is not processed. When this option is set, it overrides *fontcolor* option.

`font`

The font family to be used for drawing text. By default Sans.

`fontfile`

The font file to be used for drawing text. The path must be included. This parameter is mandatory if the fontconfig support is disabled.

`draw`

This option does not exist, please see the timeline system

alpha

Draw the text applying alpha blending. The value can be either a number between 0.0 and 1.0 The expression accepts the same variables *x*, *y* do. The default value is 1. Please see `fontcolor_expr`

fontsize

The font size to be used for drawing text. The default value of *fontsize* is 16.

text\_shaping

If set to 1, attempt to shape the text (for example, reverse the order of right-to-left text and join Arabic characters) before drawing it. Otherwise, just draw the text exactly as given. By default 1 (if supported).

ft\_load\_flags

The flags to be used for loading the fonts.

The flags map the corresponding flags supported by libfreetype, and are a combination of the following values:

*default*  
*no\_scale*  
*no\_hinting*  
*render*  
*no\_bitmap*  
*vertical\_layout*  
*force\_autohint*  
*crop\_bitmap*  
*pedantic*  
*ignore\_global\_advance\_width*  
*no\_recurse*  
*ignore\_transform*  
*monochrome*  
*linear\_design*  
*no\_autohint*

Default value is "default".

For more information consult the documentation for the `FT_LOAD_*` libfreetype flags.

shadowcolor

The color to be used for drawing a shadow behind the drawn text. For the syntax of this option, check the "Color" section in the `ffmpeg-utils` manual.

The default value of *shadowcolor* is "black".

shadowx  
shadowy

The x and y offsets for the text shadow position with respect to the position of the text. They can be either positive or negative values. The default value for both is "0".

start\_number

The starting frame number for the *n/frame\_num* variable. The default value is "0".

tabsize

The size in number of spaces to use for rendering the tab. Default value is 4.

timecode

Set the initial timecode representation in "hh:mm:ss[;.:]ff" format. It can be used with or without text parameter. *timecode\_rate* option must be specified.

timecode\_rate, rate, r

Set the timecode frame rate (timecode only).

text

The text string to be drawn. The text must be a sequence of UTF-8 encoded characters. This parameter is mandatory if no file is specified with the parameter *textfile*.

textfile

A text file containing text to be drawn. The text must be a sequence of UTF-8 encoded characters.

This parameter is mandatory if no text string is specified with the parameter *text*.

If both *text* and *textfile* are specified, an error is thrown.

reload

If set to 1, the *textfile* will be reloaded before each frame. Be sure to update it atomically, or it may be read partially, or even fail.

x  
y

The expressions which specify the offsets where text will be drawn within the video frame. They are relative to the top/left border of the output image.

The default value of  $x$  and  $y$  is "0".

See below for the list of accepted constants and functions.

The parameters for  $x$  and  $y$  are expressions containing the following constants and functions:

`dar`

input display aspect ratio, it is the same as  $(w / h) * sar$

`hsub`

`vsub`

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

`line_h, lh`

the height of each text line

`main_h, h, H`

the input height

`main_w, w, W`

the input width

`max_glyph_a, ascent`

the maximum distance from the baseline to the highest/upper grid coordinate used to place a glyph outline point, for all the rendered glyphs. It is a positive value, due to the grid's orientation with the Y axis upwards.

`max_glyph_d, descent`

the maximum distance from the baseline to the lowest grid coordinate used to place a glyph outline point, for all the rendered glyphs. This is a negative value, due to the grid's orientation, with the Y axis upwards.

`max_glyph_h`

maximum glyph height, that is the maximum height for all the glyphs contained in the rendered text, it is equivalent to *ascent - descent*.

`max_glyph_w`

maximum glyph width, that is the maximum width for all the glyphs contained in the rendered text



`n`

the number of input frame, starting from 0

`rand(min, max)`

return a random number included between *min* and *max*

`sar`

The input sample aspect ratio.

`t`

timestamp expressed in seconds, NAN if the input timestamp is unknown

`text_h, th`

the height of the rendered text

`text_w, tw`

the width of the rendered text

`x`

`y`

the x and y offset coordinates where the text is drawn.

These parameters allow the *x* and *y* expressions to refer each other, so you can for example specify `y=x/dar`.

### 37.32.2 Text expansion# TOC

If `expansion` is set to `strftime`, the filter recognizes `strftime()` sequences in the provided text and expands them accordingly. Check the documentation of `strftime()`. This feature is deprecated.

If `expansion` is set to `none`, the text is printed verbatim.

If `expansion` is set to `normal` (which is the default), the following expansion mechanism is used.

The backslash character `'\'`, followed by any character, always expands to the second character.

Sequence of the form `%{ . . . }` are expanded. The text between the braces is a function name, possibly followed by arguments separated by `':'`. If the arguments contain special characters or delimiters (`':'` or `'`}), they should be escaped.

Note that they probably must also be escaped as the value for the `text` option in the filter argument string and as the filter argument in the filtergraph description, and possibly also for the shell, that makes up to four levels of escaping; using a text file avoids these problems.

The following functions are available:

`expr, e`

The expression evaluation result.

It must take one argument specifying the expression to be evaluated, which accepts the same constants and functions as the *x* and *y* values. Note that not all constants should be used, for example the text size is not known when evaluating the expression, so the constants *text\_w* and *text\_h* will have an undefined value.

`expr_int_format, eif`

Evaluate the expression's value and output as formatted integer.

The first argument is the expression to be evaluated, just as for the *expr* function. The second argument specifies the output format. Allowed values are 'x', 'X', 'd' and 'u'. They are treated exactly as in the `printf` function. The third parameter is optional and sets the number of positions taken by the output. It can be used to add padding with zeros from the left.

`gmtime`

The time at which the filter is running, expressed in UTC. It can accept an argument: a `strftime()` format string.

`localtime`

The time at which the filter is running, expressed in the local time zone. It can accept an argument: a `strftime()` format string.

`metadata`

Frame metadata. It must take one argument specifying metadata key.

`n, frame_num`

The frame number, starting from 0.

`pict_type`

A 1 character description of the current picture type.

`pts`

The timestamp of the current frame. It can take up to two arguments.

The first argument is the format of the timestamp; it defaults to `flt` for seconds as a decimal number with microsecond accuracy; `hms` stands for a formatted `[-]HH:MM:SS.mmm` timestamp with millisecond accuracy.

The second argument is an offset added to the timestamp.

### 37.32.3 Examples# TOC

- Draw "Test Text" with font FreeSerif, using the default values for the optional parameters.

```
drawtext="fontfile=/usr/share/fonts/truetype/freefont/FreeSerif.ttf: text='Test Text' "
```

- Draw 'Test Text' with font FreeSerif of size 24 at position x=100 and y=50 (counting from the top-left corner of the screen), text is yellow with a red box around it. Both the text and the box have an opacity of 20%.

```
drawtext="fontfile=/usr/share/fonts/truetype/freefont/FreeSerif.ttf: text='Test Text':\
x=100: y=50: fontsize=24: fontcolor=yellow@0.2: box=1: boxcolor=red@0.2"
```

Note that the double quotes are not necessary if spaces are not used within the parameter list.

- Show the text at the center of the video frame:

```
drawtext="fontsize=30:fontfile=FreeSerif.ttf:text='hello world':x=(w-text_w)/2:y=(h-text_h-line_h)/2"
```

- Show a text line sliding from right to left in the last row of the video frame. The file LONG\_LINE is assumed to contain a single line with no newlines.

```
drawtext="fontsize=15:fontfile=FreeSerif.ttf:text=LONG_LINE:y=h-line_h:x=-50*t"
```

- Show the content of file CREDITS off the bottom of the frame and scroll up.

```
drawtext="fontsize=20:fontfile=FreeSerif.ttf:textfile=CREDITS:y=h-20*t"
```

- Draw a single green letter "g", at the center of the input video. The glyph baseline is placed at half screen height.

```
drawtext="fontsize=60:fontfile=FreeSerif.ttf:fontcolor=green:text=g:x=(w-max_glyph_w)/2:y=h/2-ascent"
```

- Show text for 1 second every 3 seconds:

```
drawtext="fontfile=FreeSerif.ttf:fontcolor=white:x=100:y=x/dar:enable=1t(mod(t\,3)\,1):text='blink' "
```

- Use fontconfig to set the font. Note that the colons need to be escaped.

```
drawtext='fontfile=Linux Libertine O-40\:style=Semibold:text=FFmpeg'
```

- Print the date of a real-time encoding (see strftime(3)):

```
drawtext='fontfile=FreeSans.ttf:text=%{localtime\:%a %b %d %Y}'
```

- Show text fading in and out (appearing/disappearing):

```
#!/bin/sh
SD=1.0 # display start
DE=10.0 # display end
FD=1.5 # fade in duration
FD=1 # fade out duration
ffplay -f lavfi,color,drawtext=test?fontsize=50:fontfile=FreeSerif.ttf:fontcolor=gray:ff0000[if\\ \\ clip(255*(1-between(t\\, SDG + SPD\\, DDE - SPD)) * ((t - SDG)/SPD)/between(t\\, SDG\\, DDE + SPD)) * ((t - DDE)/SPD)/between(t\\, DDE\\, DDE + SPD)) /\\, 0\\, 255) \\ \\ \\ \\ 2 ]'
```

For more information about libfreetype, check: <http://www.freetype.org/>.

For more information about fontconfig, check:  
<http://freedesktop.org/software/fontconfig/fontconfig-user.html>.

For more information about libfribidi, check: <http://fribidi.org/>.

## 37.33 edgedetect# TOC

Detect and draw edges. The filter uses the Canny Edge Detection algorithm.

The filter accepts the following options:

low  
high

Set low and high threshold values used by the Canny thresholding algorithm.

The high threshold selects the "strong" edge pixels, which are then connected through 8-connectivity with the "weak" edge pixels selected by the low threshold.

*low* and *high* threshold values must be chosen in the range [0,1], and *low* should be lesser or equal to *high*.

Default value for *low* is 20 / 255, and default value for *high* is 50 / 255.

mode

Define the drawing mode.

`'wires'`

Draw white/gray wires on black background.

`'colormix'`

Mix the colors to create a paint/cartoon effect.

Default value is *wires*.

### 37.33.1 Examples# TOC

- Standard edge detection with custom values for the hysteresis thresholding:

```
edgedetect=low=0.1:high=0.4
```

- Painting effect without thresholding:

edgedetect=mode=colormix:high=0

## 37.34 eq# TOC

Set brightness, contrast, saturation and approximate gamma adjustment.

The filter accepts the following options:

contrast

Set the contrast expression. The value must be a float value in range  $-2.0$  to  $2.0$ . The default value is "0".

brightness

Set the brightness expression. The value must be a float value in range  $-1.0$  to  $1.0$ . The default value is "0".

saturation

Set the saturation expression. The value must be a float in range  $0.0$  to  $3.0$ . The default value is "1".

gamma

Set the gamma expression. The value must be a float in range  $0.1$  to  $10.0$ . The default value is "1".

gamma\_r

Set the gamma expression for red. The value must be a float in range  $0.1$  to  $10.0$ . The default value is "1".

gamma\_g

Set the gamma expression for green. The value must be a float in range  $0.1$  to  $10.0$ . The default value is "1".

gamma\_b

Set the gamma expression for blue. The value must be a float in range  $0.1$  to  $10.0$ . The default value is "1".

gamma\_weight

Set the gamma weight expression. It can be used to reduce the effect of a high gamma value on bright image areas, e.g. keep them from getting overamplified and just plain white. The value must be a float in range  $0.0$  to  $1.0$ . A value of  $0.0$  turns the gamma correction all the way down while  $1.0$  leaves it at its full strength. Default is "1".

eval

Set when the expressions for brightness, contrast, saturation and gamma expressions are evaluated.

It accepts the following values:

`'init'`

only evaluate expressions once during the filter initialization or when a command is processed

`'frame'`

evaluate expressions for each incoming frame

Default value is `'init'`.

The expressions accept the following parameters:

n

frame count of the input frame starting from 0

pos

byte position of the corresponding packet in the input file, NAN if unspecified

r

frame rate of the input video, NAN if the input frame rate is unknown

t

timestamp expressed in seconds, NAN if the input timestamp is unknown

### **37.34.1 Commands# TOC**

The filter supports the following commands:

contrast

Set the contrast expression.

brightness

Set the brightness expression.

saturation

Set the saturation expression.

gamma

Set the gamma expression.

gamma\_r

Set the gamma\_r expression.

gamma\_g

Set gamma\_g expression.

gamma\_b

Set gamma\_b expression.

gamma\_weight

Set gamma\_weight expression.

The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

## **37.35 erosion# TOC**

Apply erosion effect to the video.

This filter replaces the pixel by the local(3x3) minimum.

It accepts the following options:

threshold0

threshold1

threshold2

threshold3

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

coordinates

Flag which specifies the pixel to refer to. Default is 255 i.e. all eight pixels are used.

Flags to local 3x3 coordinates maps like this:

1 2 3 4 5 6 7 8

## 37.36 extractplanes# TOC

Extract color channel components from input video stream into separate grayscale video streams.

The filter accepts the following option:

`planes`

Set plane(s) to extract.

Available values for planes are:

`'y'`  
`'u'`  
`'v'`  
`'a'`  
`'r'`  
`'g'`  
`'b'`

Choosing planes not available in the input will result in an error. That means you cannot select `r`, `g`, `b` planes with `y`, `u`, `v` planes at same time.

### 37.36.1 Examples# TOC

- Extract luma, u and v color channel component from input video frame into 3 grayscale outputs:

```
ffmpeg -i video.avi -filter_complex 'extractplanes=y+u+v[y][u][v]' -map '[y]' y.avi -map '[u]' u.avi -map '[v]' v.avi
```

## 37.37 elbg# TOC

Apply a posterize effect using the ELBG (Enhanced LBG) algorithm.

For each input image, the filter will compute the optimal mapping from the input to the output given the codebook length, that is the number of distinct output colors.

This filter accepts the following options.

`codebook_length, l`

Set codebook length. The value must be a positive integer, and represents the number of distinct output colors. Default value is 256.

`nb_steps, n`



Set the maximum number of iterations to apply for computing the optimal mapping. The higher the value the better the result and the higher the computation time. Default value is 1.

`seed, s`

Set a random seed, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to -1, the filter will try to use a good random seed on a best effort basis.

`pal8`

Set pal8 output pixel format. This option does not work with codebook length greater than 256.

## 37.38 fade# TOC

Apply a fade-in/out effect to the input video.

It accepts the following parameters:

`type, t`

The effect type can be either "in" for a fade-in, or "out" for a fade-out effect. Default is in.

`start_frame, s`

Specify the number of the frame to start applying the fade effect at. Default is 0.

`nb_frames, n`

The number of frames that the fade effect lasts. At the end of the fade-in effect, the output video will have the same intensity as the input video. At the end of the fade-out transition, the output video will be filled with the selected `color`. Default is 25.

`alpha`

If set to 1, fade only alpha channel, if one exists on the input. Default value is 0.

`start_time, st`

Specify the timestamp (in seconds) of the frame to start to apply the fade effect. If both `start_frame` and `start_time` are specified, the fade will start at whichever comes last. Default is 0.

`duration, d`

The number of seconds for which the fade effect has to last. At the end of the fade-in effect the output video will have the same intensity as the input video, at the end of the fade-out transition the output video will be filled with the selected `color`. If both `duration` and `nb_frames` are specified, `duration` is used. Default is 0 (`nb_frames` is used by default).

`color, c`

Specify the color of the fade. Default is "black".

### 37.38.1 Examples# TOC

- Fade in the first 30 frames of video:

```
fade=in:0:30
```

The command above is equivalent to:

```
fade=t=in:s=0:n=30
```

- Fade out the last 45 frames of a 200-frame video:

```
fade=out:155:45  
fade=type=out:start_frame=155:nb_frames=45
```

- Fade in the first 25 frames and fade out the last 25 frames of a 1000-frame video:

```
fade=in:0:25, fade=out:975:25
```

- Make the first 5 frames yellow, then fade in from frame 5-24:

```
fade=in:5:20:color=yellow
```

- Fade in alpha over first 25 frames of video:

```
fade=in:0:25:alpha=1
```

- Make the first 5.5 seconds black, then fade in for 0.5 seconds:

```
fade=t=in:st=5.5:d=0.5
```

### 37.39 fftfilt# TOC

Apply arbitrary expressions to samples in frequency domain

`dc_Y`

Adjust the dc value (gain) of the luma plane of the image. The filter accepts an integer value in range 0 to 1000. The default value is set to 0.

`dc_U`

Adjust the dc value (gain) of the 1st chroma plane of the image. The filter accepts an integer value in range 0 to 1000. The default value is set to 0.

`dc_V`

Adjust the dc value (gain) of the 2nd chroma plane of the image. The filter accepts an integer value in range 0 to 1000. The default value is set to 0.

`weight_Y`

Set the frequency domain weight expression for the luma plane.

`weight_U`

Set the frequency domain weight expression for the 1st chroma plane.

`weight_V`

Set the frequency domain weight expression for the 2nd chroma plane.

The filter accepts the following variables:

`X`

`Y`

The coordinates of the current sample.

`W`

`H`

The width and height of the image.

### 37.39.1 Examples# TOC

- High-pass:

```
fftfilt=dc_Y=128:weight_Y='squish(1-(Y+X)/100)'
```

- Low-pass:

```
fftfilt=dc_Y=0:weight_Y='squish((Y+X)/100-1)'
```

- Sharpen:

```
fftfilt=dc_Y=0:weight_Y='1+squish(1-(Y+X)/100)'
```

### 37.40 field# TOC

Extract a single field from an interlaced image using stride arithmetic to avoid wasting CPU time. The output frames are marked as non-interlaced.

The filter accepts the following options:

type

Specify whether to extract the top (if the value is 0 or `top`) or the bottom field (if the value is 1 or `bottom`).

## 37.41 fieldmatch# TOC

Field matching filter for inverse telecine. It is meant to reconstruct the progressive frames from a telecined stream. The filter does not drop duplicated frames, so to achieve a complete inverse telecine `fieldmatch` needs to be followed by a decimation filter such as `decimate` in the filtergraph.

The separation of the field matching and the decimation is notably motivated by the possibility of inserting a de-interlacing filter fallback between the two. If the source has mixed telecined and real interlaced content, `fieldmatch` will not be able to match fields for the interlaced parts. But these remaining combed frames will be marked as interlaced, and thus can be de-interlaced by a later filter such as `yadif` before decimation.

In addition to the various configuration options, `fieldmatch` can take an optional second stream, activated through the `ppsrc` option. If enabled, the frames reconstruction will be based on the fields and frames from this second stream. This allows the first input to be pre-processed in order to help the various algorithms of the filter, while keeping the output lossless (assuming the fields are matched properly). Typically, a field-aware denoiser, or brightness/contrast adjustments can help.

Note that this filter uses the same algorithms as TIVTC/TFM (AviSynth project) and VIVTC/VFM (VapourSynth project). The later is a light clone of TFM from which `fieldmatch` is based on. While the semantic and usage are very close, some behaviour and options names can differ.

The `decimate` filter currently only works for constant frame rate input. If your input has mixed telecined (30fps) and progressive content with a lower framerate like 24fps use the following filterchain to produce the necessary cfr stream: `dejudder, fps=30000/1001, fieldmatch, decimate`.

The filter accepts the following options:

order

Specify the assumed field order of the input stream. Available values are:

`'auto'`

Auto detect parity (use FFmpeg's internal parity value).

`'bff'`

Assume bottom field first.

`'tff'`

Assume top field first.

Note that it is sometimes recommended not to trust the parity announced by the stream.

Default value is *auto*.

mode

Set the matching mode or strategy to use. `pc` mode is the safest in the sense that it won't risk creating jerkiness due to duplicate frames when possible, but if there are bad edits or blended fields it will end up outputting combed frames when a good match might actually exist. On the other hand, `pcn_ub` mode is the most risky in terms of creating jerkiness, but will almost always find a good frame if there is one. The other values are all somewhere in between `pc` and `pcn_ub` in terms of risking jerkiness and creating duplicate frames versus finding good matches in sections with bad edits, orphaned fields, blended fields, etc.

More details about `p/c/n/u/b` are available in `p/c/n/u/b` meaning section.

Available values are:

`'pc'`

2-way matching (`p/c`)

`'pc_n'`

2-way matching, and trying 3rd match if still combed (`p/c + n`)

`'pc_u'`

2-way matching, and trying 3rd match (same order) if still combed (`p/c + u`)

`'pc_n_ub'`

2-way matching, trying 3rd match if still combed, and trying 4th/5th matches if still combed (`p/c + n + u/b`)

`'pcn'`

3-way matching (`p/c/n`)

`'pcn_ub'`

3-way matching, and trying 4th/5th matches if all 3 of the original matches are detected as combed (`p/c/n + u/b`)

The parenthesis at the end indicate the matches that would be used for that mode assuming `order=tff` (and `field` on *auto* or *top*).

In terms of speed `pc` mode is by far the fastest and `pcn_ub` is the slowest.

Default value is `pc_n`.

#### `ppsrc`

Mark the main input stream as a pre-processed input, and enable the secondary input stream as the clean source to pick the fields from. See the filter introduction for more details. It is similar to the `clip2` feature from VFM/TFM.

Default value is 0 (disabled).

#### `field`

Set the field to match from. It is recommended to set this to the same value as `order` unless you experience matching failures with that setting. In certain circumstances changing the field that is used to match from can have a large impact on matching performance. Available values are:

`'auto'`

Automatic (same value as `order`).

`'bottom'`

Match from the bottom field.

`'top'`

Match from the top field.

Default value is `auto`.

#### `mchroma`

Set whether or not chroma is included during the match comparisons. In most cases it is recommended to leave this enabled. You should set this to 0 only if your clip has bad chroma problems such as heavy rainbowing or other artifacts. Setting this to 0 could also be used to speed things up at the cost of some accuracy.

Default value is 1.

#### `y0`

#### `y1`

These define an exclusion band which excludes the lines between `y0` and `y1` from being included in the field matching decision. An exclusion band can be used to ignore subtitles, a logo, or other things that may interfere with the matching. `y0` sets the starting scan line and `y1` sets the ending line; all lines in between `y0` and `y1` (including `y0` and `y1`) will be ignored. Setting `y0` and `y1` to the same value will disable the feature. `y0` and `y1` defaults to 0.

## scthresh

Set the scene change detection threshold as a percentage of maximum change on the luma plane. Good values are in the [ 8.0 , 14.0 ] range. Scene change detection is only relevant in case `combatch=sc`. The range for `scthresh` is [ 0.0 , 100.0 ].

Default value is 12.0.

## combatch

When `combatch` is not *none*, `fieldmatch` will take into account the combed scores of matches when deciding what match to use as the final match. Available values are:

‘none’

No final matching based on combed scores.

‘sc’

Combed scores are only used when a scene change is detected.

‘full’

Use combed scores all the time.

Default is *sc*.

## combdbg

Force `fieldmatch` to calculate the combed metrics for certain matches and print them. This setting is known as `micout` in TFM/VFM vocabulary. Available values are:

‘none’

No forced calculation.

‘pcn’

Force p/c/n calculations.

‘pcnub’

Force p/c/n/u/b calculations.

Default value is *none*.

## cthresh

This is the area combing threshold used for combed frame detection. This essentially controls how "strong" or "visible" combing must be to be detected. Larger values mean combing must be more visible and smaller values mean combing can be less visible or strong and still be detected. Valid settings are from -1 (every pixel will be detected as combed) to 255 (no pixel will be detected as combed). This is basically a pixel difference value. A good range is [ 8 , 12 ].

Default value is 9.

chroma

Sets whether or not chroma is considered in the combed frame decision. Only disable this if your source has chroma problems (rainbowing, etc.) that are causing problems for the combed frame detection with chroma enabled. Actually, using `chroma=0` is usually more reliable, except for the case where there is chroma only combing in the source.

Default value is 0.

blockx

blocky

Respectively set the x-axis and y-axis size of the window used during combed frame detection. This has to do with the size of the area in which `combpel` pixels are required to be detected as combed for a frame to be declared combed. See the `combpel` parameter description for more info. Possible values are any number that is a power of 2 starting at 4 and going up to 512.

Default value is 16.

combpel

The number of combed pixels inside any of the `blocky` by `blockx` size blocks on the frame for the frame to be detected as combed. While `cthresh` controls how "visible" the combing must be, this setting controls "how much" combing there must be in any localized area (a window defined by the `blockx` and `blocky` settings) on the frame. Minimum value is 0 and maximum is `blocky * blockx` (at which point no frames will ever be detected as combed). This setting is known as MI in TFM/VFM vocabulary.

Default value is 80.

### 37.41.1 p/c/n/u/b meaning# TOC

#### 37.41.1.1 p/c/n# TOC

We assume the following telecined stream:

```
Top fields:      1 2 2 3 4
Bottom fields:   1 2 3 4 4
```



The numbers correspond to the progressive frame the fields relate to. Here, the first two frames are progressive, the 3rd and 4th are combed, and so on.

When `fieldmatch` is configured to run a matching from bottom (`field=bottom`) this is how this input stream get transformed:

```
Input stream:
      T      1 2 2 3 4
      B      1 2 3 4 4  <-- matching reference

Matches:
      c c n n c

Output stream:
      T      1 2 3 4 4
      B      1 2 3 4 4
```

As a result of the field matching, we can see that some frames get duplicated. To perform a complete inverse telecine, you need to rely on a decimation filter after this operation. See for instance the decimate filter.

The same operation now matching from top fields (`field=top`) looks like this:

```
Input stream:
      T      1 2 2 3 4  <-- matching reference
      B      1 2 3 4 4

Matches:
      c c p p c

Output stream:
      T      1 2 2 3 4
      B      1 2 2 3 4
```

In these examples, we can see what *p*, *c* and *n* mean; basically, they refer to the frame and field of the opposite parity:

- *p* matches the field of the opposite parity in the previous frame
- *c* matches the field of the opposite parity in the current frame
- *n* matches the field of the opposite parity in the next frame

### 37.41.1.2 u/b# TOC

The *u* and *b* matching are a bit special in the sense that they match from the opposite parity flag. In the following examples, we assume that we are currently matching the 2nd frame (Top:2, bottom:2). According to the match, a 'x' is placed above and below each matched fields.

With bottom matching (`field=bottom`):

Match:	c	p	n	b	u
	x	x	x	x	x
Top	1 2 2	1 2 2	1 2 2	1 2 2	1 2 2
Bottom	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
	x	x	x	x	x

Output frames:	2	1	2	2	2
	2	2	2	1	3

With top matching (`field=top`):

Match:	c	p	n	b	u
	x	x	x	x	x
Top	1 2 2	1 2 2	1 2 2	1 2 2	1 2 2
Bottom	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
	x	x	x	x	x

Output frames:	2	2	2	1	2
	2	1	3	2	2

## 37.41.2 Examples# TOC

Simple IVTC of a top field first telecined stream:

```
fieldmatch=order=tff:combmatch=none, decimate
```

Advanced IVTC, with fallback on yadif for still combed frames:

```
fieldmatch=order=tff:combmatch=full, yadif=deint=interlaced, decimate
```

## 37.42 fieldorder# TOC

Transform the field order of the input video.

It accepts the following parameters:

`order`

The output field order. Valid values are *tff* for top field first or *bff* for bottom field first.

The default value is 'tff'.

The transformation is done by shifting the picture content up or down by one line, and filling the remaining line with appropriate picture content. This method is consistent with most broadcast field order converters.

If the input video is not flagged as being interlaced, or it is already flagged as being of the required output field order, then this filter does not alter the incoming video.

It is very useful when converting to or from PAL DV material, which is bottom field first.

For example:

```
ffmpeg -i in.vob -vf "fieldorder=bff" out.dv
```

## 37.43 fifo# TOC

Buffer input images and send them when they are requested.

It is mainly useful when auto-inserted by the libavfilter framework.

It does not take parameters.

## 37.44 find\_rect# TOC

Find a rectangular object

It accepts the following options:

`object`

Filepath of the object image, needs to be in gray8.

`threshold`

Detection threshold, default is 0.5.

`mipmaps`

Number of mipmaps, default is 3.

`xmin, ymin, xmax, ymax`

Specifies the rectangle in which to search.

### 37.44.1 Examples# TOC

- Generate a representative palette of a given video using `ffmpeg`:

```
ffmpeg -i file.ts -vf find_rect=newref.pgm,cover_rect=cover.jpg:mode=cover new.mkv
```

## 37.45 cover\_rect# TOC

Cover a rectangular object

It accepts the following options:

`cover`

Filepath of the optional cover image, needs to be in yuv420.

`mode`

Set covering mode.

It accepts the following values:

`'cover'`

cover it by the supplied image

`'blur'`

cover it by interpolating the surrounding pixels

Default value is *blur*.

### 37.45.1 Examples# TOC

- Generate a representative palette of a given video using `ffmpeg`:

```
ffmpeg -i file.ts -vf find_rect=newref.pgm,cover_rect=cover.jpg:mode=cover new.mkv
```

## 37.46 format# TOC

Convert the input video to one of the specified pixel formats. Libavfilter will try to pick one that is suitable as input to the next filter.

It accepts the following parameters:

`pix_fmts`

A `'|'`-separated list of pixel format names, such as `"pix_fmts=yuv420p|monow|rgb24"`.

### 37.46.1 Examples# TOC

- Convert the input video to the *yuv420p* format

```
format=pix_fmts=yuv420p
```

Convert the input video to any of the formats in the list

```
format=pix_fmts=yuv420p|yuv444p|yuv410p
```

## 37.47 fps# TOC

Convert the video to specified constant frame rate by duplicating or dropping frames as necessary.

It accepts the following parameters:

`fps`

The desired output frame rate. The default is 25.

`round`

Rounding method.

Possible values are:

`zero`

zero round towards 0

`inf`

round away from 0

`down`

round towards -infinity

`up`

round towards +infinity

`near`

round to nearest

The default is `near`.

`start_time`

Assume the first PTS should be the given value, in seconds. This allows for padding/trimming at the start of stream. By default, no assumption is made about the first frame's expected PTS, so no padding or trimming is done. For example, this could be set to 0 to pad the beginning with duplicates of the first frame if a video stream starts after the audio stream or to trim any frames with a negative

PTS.

Alternatively, the options can be specified as a flat string: *fps[:round]*.

See also the `setpts` filter.

### 37.47.1 Examples# TOC

- A typical usage in order to set the fps to 25:

```
fps=fps=25
```

- Sets the fps to 24, using abbreviation and rounding method to round to nearest:

```
fps=fps=film:round=near
```

## 37.48 framepack# TOC

Pack two different video streams into a stereoscopic video, setting proper metadata on supported codecs. The two views should have the same size and framerate and processing will stop when the shorter video ends. Please note that you may conveniently adjust view properties with the `scale` and `fps` filters.

It accepts the following parameters:

`format`

The desired packing format. Supported values are:

`sbs`

The views are next to each other (default).

`tab`

The views are on top of each other.

`lines`

The views are packed by line.

`columns`

The views are packed by column.

`frameseq`

The views are temporally interleaved.

Some examples:

```
# Convert left and right views into a frame-sequential video
ffmpeg -i LEFT -i RIGHT -filter_complex framepack=frameseq OUTPUT

# Convert views into a side-by-side video with the same output resolution as the input
ffmpeg -i LEFT -i RIGHT -filter_complex [0:v]scale=wiw/2[left],[1:v]scale=wiw/2[right],[left][right]framepack=sbs OUTPUT
```

## 37.49 framerate# TOC

Change the frame rate by interpolating new video output frames from the source frames.

This filter is not designed to function correctly with interlaced media. If you wish to change the frame rate of interlaced media then you are required to deinterlace before this filter and re-interlace after this filter.

A description of the accepted options follows.

fps

Specify the output frames per second. This option can also be specified as a value alone. The default is 50.

interp\_start

Specify the start of a range where the output frame will be created as a linear interpolation of two frames. The range is [0-255], the default is 15.

interp\_end

Specify the end of a range where the output frame will be created as a linear interpolation of two frames. The range is [0-255], the default is 240.

scene

Specify the level at which a scene change is detected as a value between 0 and 100 to indicate a new scene; a low value reflects a low probability for the current frame to introduce a new scene, while a higher value means the current frame is more likely to be one. The default is 7.

flags

Specify flags influencing the filter process.

Available value for *flags* is:

scene\_change\_detect, scd

Enable scene change detection using the value of the option *scene*. This flag is enabled by default.

## 37.50 framestep# TOC

Select one frame every N-th frame.

This filter accepts the following option:

`step`

Select frame after every `step` frames. Allowed values are positive integers higher than 0. Default value is 1.

## 37.51 frei0r# TOC

Apply a frei0r effect to the input video.

To enable the compilation of this filter, you need to install the frei0r header and configure FFmpeg with `--enable-frei0r`.

It accepts the following parameters:

`filter_name`

The name of the frei0r effect to load. If the environment variable `FREI0R_PATH` is defined, the frei0r effect is searched for in each of the directories specified by the colon-separated list in `FREI0R_PATH`. Otherwise, the standard frei0r paths are searched, in this order:  
`HOME/.frei0r-1/lib/, /usr/local/lib/frei0r-1/, /usr/lib/frei0r-1/.`

`filter_params`

A `'|'`-separated list of parameters to pass to the frei0r effect.

A frei0r effect parameter can be a boolean (its value is either "y" or "n"), a double, a color (specified as *R/G/B*, where *R*, *G*, and *B* are floating point numbers between 0.0 and 1.0, inclusive) or by a color description specified in the "Color" section in the ffmpeg-utils manual), a position (specified as *X/Y*, where *X* and *Y* are floating point numbers) and/or a string.

The number and types of parameters depend on the loaded effect. If an effect parameter is not specified, the default value is set.

### 37.51.1 Examples# TOC

- Apply the distort0r effect, setting the first two double parameters:

```
frei0r=filter_name=distort0r:filter_params=0.5|0.01
```

- Apply the colordistance effect, taking a color as the first parameter:



```
frei0r=colordistance:0.2/0.3/0.4
frei0r=colordistance:violet
frei0r=colordistance:0x112233
```

- Apply the perspective effect, specifying the top left and top right image positions:

```
frei0r=perspective:0.2/0.2|0.8/0.2
```

For more information, see <http://frei0r.dyne.org>

## 37.52 fspp# TOC

Apply fast and simple postprocessing. It is a faster version of spp.

It splits (I)DCT into horizontal/vertical passes. Unlike the simple post- processing filter, one of them is performed once per block, not per pixel. This allows for much higher speed.

The filter accepts the following options:

quality

Set quality. This option defines the number of levels for averaging. It accepts an integer in the range 4-5. Default value is 4.

qp

Force a constant quantization parameter. It accepts an integer in range 0-63. If not set, the filter will use the QP from the video stream (if available).

strength

Set filter strength. It accepts an integer in range -15 to 32. Lower values mean more details but also more artifacts, while higher values make the image smoother but also blurrier. Default value is 0 â PSNR optimal.

use\_bframe\_qp

Enable the use of the QP from the B-Frames if set to 1. Using this option may cause flicker since the B-Frames have often larger QP. Default is 0 (not enabled).

## 37.53 geq# TOC

The filter accepts the following options:

lum\_expr, lum

Set the luminance expression.

`cb_expr, cb`

Set the chrominance blue expression.

`cr_expr, cr`

Set the chrominance red expression.

`alpha_expr, a`

Set the alpha expression.

`red_expr, r`

Set the red expression.

`green_expr, g`

Set the green expression.

`blue_expr, b`

Set the blue expression.

The colorspace is selected according to the specified options. If one of the `lum_expr`, `cb_expr`, or `cr_expr` options is specified, the filter will automatically select a YCbCr colorspace. If one of the `red_expr`, `green_expr`, or `blue_expr` options is specified, it will select an RGB colorspace.

If one of the chrominance expression is not defined, it falls back on the other one. If no alpha expression is specified it will evaluate to opaque value. If none of chrominance expressions are specified, they will evaluate to the luminance expression.

The expressions can use the following variables and functions:

`N`

The sequential number of the filtered frame, starting from 0.

`X`

`Y`

The coordinates of the current sample.

`W`

`H`

The width and height of the image.

SW  
SH

Width and height scale depending on the currently filtered plane. It is the ratio between the corresponding luma plane number of pixels and the current plane ones. E.g. for YUV4:2:0 the values are 1, 1 for the luma plane, and 0.5, 0.5 for chroma planes.

T

Time of the current frame, expressed in seconds.

$p(x, y)$

Return the value of the pixel at location (x,y) of the current plane.

$lum(x, y)$

Return the value of the pixel at location (x,y) of the luminance plane.

$cb(x, y)$

Return the value of the pixel at location (x,y) of the blue-difference chroma plane. Return 0 if there is no such plane.

$cr(x, y)$

Return the value of the pixel at location (x,y) of the red-difference chroma plane. Return 0 if there is no such plane.

$r(x, y)$

$g(x, y)$

$b(x, y)$

Return the value of the pixel at location (x,y) of the red/green/blue component. Return 0 if there is no such component.

$alpha(x, y)$

Return the value of the pixel at location (x,y) of the alpha plane. Return 0 if there is no such plane.

For functions, if  $x$  and  $y$  are outside the area, the value will be automatically clipped to the closer edge.

### 37.53.1 Examples# TOC

- Flip the image horizontally:

$geq=p(W-X\backslash, Y)$

- Generate a bidimensional sine wave, with angle  $\pi/3$  and a wavelength of 100 pixels:

```
geq=128 + 100*sin(2*(PI/100)*(cos(PI/3)*(X-50*T) + sin(PI/3)*Y)):128:128
```

- Generate a fancy enigmatic moving light:

```
nullsrc=s=256x256,geq=random(1)/hypot(X-cos(N*0.07)*W/2-W/2\,Y-sin(N*0.09)*H/2-H/2)^2*1000000*sin(N*0.02):128:128
```

- Generate a quick emboss effect:

```
format=gray,geq=lum_expr='(p(X,Y)+(256-p(X-4,Y-4)))/2'
```

- Modify RGB components depending on pixel position:

```
geq=r='X/W*r(X,Y)':g='(1-X/W)*g(X,Y)':b='(H-Y)/H*b(X,Y)'
```

- Create a radial gradient that is the same size as the input (also see the vignette filter):

```
geq=lum=255*gauss((X/W-0.5)*3)*gauss((Y/H-0.5)*3)/gauss(0)/gauss(0),format=gray
```

- Create a linear gradient to use as a mask for another filter, then compose with overlay. In this example the video will gradually become more blurry from the top to the bottom of the y-axis as defined by the linear gradient:

```
ffmpeg -i input.mp4 -filter_complex "geq=lum=255*(Y/H),format=gray[grad];[0:v]boxblur=4[blur];[blur][grad]alphaseg[alpha];[0:v][alpha]overlay" output.mp4
```

## 37.54 gradfun# TOC

Fix the banding artifacts that are sometimes introduced into nearly flat regions by truncation to 8bit color depth. Interpolate the gradients that should go where the bands are, and dither them.

It is designed for playback only. Do not use it prior to lossy compression, because compression tends to lose the dither and bring back the bands.

It accepts the following parameters:

**strength**

The maximum amount by which the filter will change any one pixel. This is also the threshold for detecting nearly flat regions. Acceptable values range from .51 to 64; the default value is 1.2. Out-of-range values will be clipped to the valid range.

**radius**

The neighborhood to fit the gradient to. A larger radius makes for smoother gradients, but also prevents the filter from modifying the pixels near detailed regions. Acceptable values are 8-32; the default value is 16. Out-of-range values will be clipped to the valid range.

Alternatively, the options can be specified as a flat string: *strength[:radius]*

### 37.54.1 Examples# TOC

- Apply the filter with a 3.5 strength and radius of 8:

```
gradfun=3.5:8
```

- Specify radius, omitting the strength (which will fall-back to the default value):

```
gradfun=radius=8
```

### 37.55 haldclut# TOC

Apply a Hald CLUT to a video stream.

First input is the video stream to process, and second one is the Hald CLUT. The Hald CLUT input can be a simple picture or a complete video stream.

The filter accepts the following options:

`shortest`

Force termination when the shortest input terminates. Default is 0.

`repeatlast`

Continue applying the last CLUT after the end of the stream. A value of 0 disable the filter after the last frame of the CLUT is reached. Default is 1.

`haldclut` also has the same interpolation options as `lut3d` (both filters share the same internals).

More information about the Hald CLUT can be found on Eskil Steenberg's website (Hald CLUT author) at <http://www.quelsolaar.com/technology/clut.html>.

### 37.55.1 Workflow examples# TOC

#### 37.55.1.1 Hald CLUT video stream# TOC

Generate an identity Hald CLUT stream altered with various effects:

```
ffmpeg -f lavfi -i haldclutsrc=8 -vf "hue=H=2*PI*t:s=sin(2*PI*t)+1, curves=cross_process" -t 10 -c:v ffv1 clut.nut
```

Note: make sure you use a lossless codec.

Then use it with `haldclut` to apply it on some random stream:

```
ffmpeg -f lavfi -i mandelbrot -i clut.nut -filter_complex '[0][1] haldclut' -t 20 mandelclut.mkv
```

The Hald CLUT will be applied to the 10 first seconds (duration of `clut.nut`), then the latest picture of that CLUT stream will be applied to the remaining frames of the `mandelbrot` stream.

### 37.55.1.2 Hald CLUT with preview# TOC

A Hald CLUT is supposed to be a squared image of `Level*Level*Level` by `Level*Level*Level` pixels. For a given Hald CLUT, FFmpeg will select the biggest possible square starting at the top left of the picture. The remaining padding pixels (bottom or right) will be ignored. This area can be used to add a preview of the Hald CLUT.

Typically, the following generated Hald CLUT will be supported by the `haldclut` filter:

```
ffmpeg -f lavfi -i haldclutsrc=8 -vf "  
    pad=iw+320 [padded_clut];  
    smptebars=s=320x256, split [a][b];  
    [padded_clut][a] overlay=W-320:h, curves=color_negative [main];  
    [main][b] overlay=W-320" -frames:v 1 clut.png
```

It contains the original and a preview of the effect of the CLUT: SMPTE color bars are displayed on the right-top, and below the same color bars processed by the color changes.

Then, the effect of this Hald CLUT can be visualized with:

```
ffplay input.mkv -vf "movie=clut.png, [in] haldclut"
```

### 37.56 hflip# TOC

Flip the input video horizontally.

For example, to horizontally flip the input video with `ffmpeg`:

```
ffmpeg -i in.avi -vf "hflip" out.avi
```

### 37.57 histeq# TOC

This filter applies a global color histogram equalization on a per-frame basis.

It can be used to correct video that has a compressed range of pixel intensities. The filter redistributes the pixel intensities to equalize their distribution across the intensity range. It may be viewed as an "automatically adjusting contrast filter". This filter is useful only for correcting degraded or poorly captured source video.

The filter accepts the following options:

`strength`

Determine the amount of equalization to be applied. As the strength is reduced, the distribution of pixel intensities more-and-more approaches that of the input frame. The value must be a float number in the range [0,1] and defaults to 0.200.

`intensity`

Set the maximum intensity that can generated and scale the output values appropriately. The strength should be set as desired and then the intensity can be limited if needed to avoid washing-out. The value must be a float number in the range [0,1] and defaults to 0.210.

`antibanding`

Set the antibanding level. If enabled the filter will randomly vary the luminance of output pixels by a small amount to avoid banding of the histogram. Possible values are `none`, `weak` or `strong`. It defaults to `none`.

## 37.58 histogram# TOC

Compute and draw a color distribution histogram for the input video.

The computed histogram is a representation of the color component distribution in an image.

The filter accepts the following options:

`mode`

Set histogram mode.

It accepts the following values:

`'levels'`

Standard histogram that displays the color components distribution in an image. Displays color graph for each color component. Shows distribution of the Y, U, V, A or R, G, B components, depending on input format, in the current frame. Below each graph a color component scale meter is shown.

`'color'`

Displays chroma values (U/V color placement) in a two dimensional graph (which is called a vectorscope). The brighter a pixel in the vectorscope, the more pixels of the input frame correspond to that pixel (i.e., more pixels have this chroma value). The V component is displayed on the horizontal (X) axis, with the leftmost side being V = 0 and the rightmost side being V = 255. The U component is displayed on the vertical (Y) axis, with the top representing U = 0 and the bottom representing U = 255.

The position of a white pixel in the graph corresponds to the chroma value of a pixel of the input clip. The graph can therefore be used to read the hue (color flavor) and the saturation (the dominance of the hue in the color). As the hue of a color changes, it moves around the square. At the center of the square the saturation is zero, which means that the corresponding pixel has no color. If the amount of a specific color is increased (while leaving the other colors unchanged) the saturation increases, and the indicator moves towards the edge of the square.

`'color2'`

Chroma values in vectorscope, similar as `color` but actual chroma values are displayed.

`'waveform'`

Per row/column color component graph. In row mode, the graph on the left side represents color component value 0 and the right side represents value = 255. In column mode, the top side represents color component value = 0 and bottom side represents value = 255.

Default value is `levels`.

`level_height`

Set height of level in `levels`. Default value is 200. Allowed range is [50, 2048].

`scale_height`

Set height of color scale in `levels`. Default value is 12. Allowed range is [0, 40].

`step`

Set step for waveform mode. Smaller values are useful to find out how many values of the same luminance are distributed across input rows/columns. Default value is 10. Allowed range is [1, 255].

`waveform_mode`

Set mode for waveform. Can be either `row`, or `column`. Default is `row`.

`waveform_mirror`

Set mirroring mode for waveform. 0 means unmirrored, 1 means mirrored. In mirrored mode, higher values will be represented on the left side for `row` mode and at the top for `column` mode. Default is 0 (unmirrored).

`display_mode`

Set display mode for waveform and `levels`. It accepts the following values:

`'parade'`

Display separate graph for the color components side by side in `row` waveform mode or one below the other in `column` waveform mode for waveform histogram mode. For `levels` histogram mode, per color component graphs are placed below each other.

Using this display mode in waveform histogram mode makes it easy to spot color casts in the highlights and shadows of an image, by comparing the contours of the top and the bottom graphs of each waveform. Since whites, grays, and blacks are characterized by exactly equal amounts of red, green, and blue, neutral areas of the picture should display three waveforms of roughly equal width/height. If not, the correction is easy to perform by making level adjustments the



three waveforms.

`'overlay'`

Presents information identical to that in the `parade`, except that the graphs representing color components are superimposed directly over one another.

This display mode in `waveform histogram` mode makes it easier to spot relative differences or similarities in overlapping areas of the color components that are supposed to be identical, such as neutral whites, grays, or blacks.

Default is `parade`.

`levels_mode`

Set mode for `levels`. Can be either `linear`, or `logarithmic`. Default is `linear`.

`components`

Set what color components to display for mode `levels`. Default is 7.

## 37.58.1 Examples# TOC

- Calculate and draw histogram:

```
ffplay -i input -vf histogram
```

## 37.59 hqdn3d# TOC

This is a high precision/quality 3d denoise filter. It aims to reduce image noise, producing smooth images and making still images really still. It should enhance compressibility.

It accepts the following optional parameters:

`luma_spatial`

A non-negative floating point number which specifies spatial luma strength. It defaults to 4.0.

`chroma_spatial`

A non-negative floating point number which specifies spatial chroma strength. It defaults to  $3.0 * \textit{luma\_spatial} / 4.0$ .

`luma_tmp`

A floating point number which specifies luma temporal strength. It defaults to  $6.0 * \textit{luma\_spatial} / 4.0$ .

`chroma_tmp`

A floating point number which specifies chroma temporal strength. It defaults to  $\text{luma\_tmp} * \text{chroma\_spatial} / \text{luma\_spatial}$ .

## 37.60 hqx# TOC

Apply a high-quality magnification filter designed for pixel art. This filter was originally created by Maxim Stepin.

It accepts the following option:

`n`

Set the scaling dimension: 2 for hq2x, 3 for hq3x and 4 for hq4x. Default is 3.

## 37.61 hstack# TOC

Stack input videos horizontally.

All streams must be of same pixel format and of same height.

Note that this filter is faster than using overlay and pad filter to create same output.

The filter accept the following option:

`nb_inputs`

Set number of input streams. Default is 2.

## 37.62 hue# TOC

Modify the hue and/or the saturation of the input.

It accepts the following parameters:

`h`

Specify the hue angle as a number of degrees. It accepts an expression, and defaults to "0".

`s`

Specify the saturation in the [-10,10] range. It accepts an expression and defaults to "1".

`H`

Specify the hue angle as a number of radians. It accepts an expression, and defaults to "0".

`b`

Specify the brightness in the [-10,10] range. It accepts an expression and defaults to "0".

`h` and `H` are mutually exclusive, and can't be specified at the same time.

The `b`, `h`, `H` and `s` option values are expressions containing the following constants:

`n`

frame count of the input frame starting from 0

`pts`

presentation timestamp of the input frame expressed in time base units

`r`

frame rate of the input video, NAN if the input frame rate is unknown

`t`

timestamp expressed in seconds, NAN if the input timestamp is unknown

`tb`

time base of the input video

### 37.62.1 Examples# TOC

- Set the hue to 90 degrees and the saturation to 1.0:

```
hue=h=90:s=1
```

- Same command but expressing the hue in radians:

```
hue=H=PI/2:s=1
```

- Rotate hue and make the saturation swing between 0 and 2 over a period of 1 second:

```
hue="H=2*PI*t: s=sin(2*PI*t)+1"
```

- Apply a 3 seconds saturation fade-in effect starting at 0:

```
hue="s=min(t/3\,1)"
```

The general fade-in expression can be written as:

```
hue="s=min(0\, max((t-START)/DURATION\, 1))"
```

- Apply a 3 seconds saturation fade-out effect starting at 5 seconds:

```
hue="s=max(0\, min(1\, (8-t)/3))"
```

The general fade-out expression can be written as:

```
hue="s=max(0\, min(1\, (START+DURATION-t)/DURATION))"
```

### 37.62.2 Commands# TOC

This filter supports the following commands:

b  
s  
h  
H

Modify the hue and/or the saturation and/or brightness of the input video. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.63 idet# TOC

Detect video interlacing type.

This filter tries to detect if the input frames are interlaced, progressive, top or bottom field first. It will also try and detect fields that are repeated between adjacent frames (a sign of telecine).

Single frame detection considers only immediately adjacent frames when classifying each frame. Multiple frame detection incorporates the classification history of previous frames.

The filter will log these metadata values:

`single.current_frame`

Detected type of current frame using single-frame detection. One of: “tff” (top field first), “bff” (bottom field first), “progressive”, or “undetermined”

`single.tff`

Cumulative number of frames detected as top field first using single-frame detection.

`multiple.tff`

Cumulative number of frames detected as top field first using multiple-frame detection.

`single.bff`

Cumulative number of frames detected as bottom field first using single-frame detection.

`multiple.current_frame`

Detected type of current frame using multiple-frame detection. One of: “tff” (top field first), “bff” (bottom field first), “progressive”, or “undetermined”

`multiple.bff`

Cumulative number of frames detected as bottom field first using multiple-frame detection.

`single.progressive`

Cumulative number of frames detected as progressive using single-frame detection.

`multiple.progressive`

Cumulative number of frames detected as progressive using multiple-frame detection.

`single.undetermined`

Cumulative number of frames that could not be classified using single-frame detection.

`multiple.undetermined`

Cumulative number of frames that could not be classified using multiple-frame detection.

`repeated.current_frame`

Which field in the current frame is repeated from the last. One of “neither”, “top”, or “bottom”.

`repeated.neither`

Cumulative number of frames with no repeated field.

`repeated.top`

Cumulative number of frames with the top field repeated from the previous frame’s top field.

`repeated.bottom`

Cumulative number of frames with the bottom field repeated from the previous frame’s bottom field.

The filter accepts the following options:

`intl_thres`

Set interlacing threshold.

`prog_thres`

Set progressive threshold.

`repeat_thres`

Threshold for repeated field detection.

`half_life`

Number of frames after which a given frame's contribution to the statistics is halved (i.e., it contributes only 0.5 to its classification). The default of 0 means that all frames seen are given full weight of 1.0 forever.

`analyze_interlaced_flag`

When this is not 0 then idet will use the specified number of frames to determine if the interlaced flag is accurate, it will not count undetermined frames. If the flag is found to be accurate it will be used without any further computations, if it is found to be inaccurate it will be cleared without any further computations. This allows inserting the idet filter as a low computational method to clean up the interlaced flag

## 37.64 il# TOC

Deinterleave or interleave fields.

This filter allows one to process interlaced images fields without deinterlacing them. Deinterleaving splits the input frame into 2 fields (so called half pictures). Odd lines are moved to the top half of the output image, even lines to the bottom half. You can process (filter) them independently and then re-interleave them.

The filter accepts the following options:

`luma_mode, l`

`chroma_mode, c`

`alpha_mode, a`

Available values for *luma\_mode*, *chroma\_mode* and *alpha\_mode* are:

'none'

Do nothing.

'deinterleave, d'

Deinterleave fields, placing one above the other.

`'interleave, i'`

Interleave fields. Reverse the effect of deinterleaving.

Default value is none.

`luma_swap, ls`  
`chroma_swap, cs`  
`alpha_swap, as`

Swap luma/chroma/alpha fields. Exchange even & odd lines. Default value is 0.

## 37.65 inflate# TOC

Apply inflate effect to the video.

This filter replaces the pixel by the local(3x3) average by taking into account only values higher than the pixel.

It accepts the following options:

`threshold0`  
`threshold1`  
`threshold2`  
`threshold3`

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

## 37.66 interlace# TOC

Simple interlacing filter from progressive contents. This interleaves upper (or lower) lines from odd frames with lower (or upper) lines from even frames, halving the frame rate and preserving image height.

Original Frame 'j'	Original Frame 'j+1'	New Frame (tff)
=====	=====	=====
Line 0 ----->		Frame 'j' Line 0
Line 1	Line 1 ---->	Frame 'j+1' Line 1
Line 2 ----->		Frame 'j' Line 2
Line 3	Line 3 ---->	Frame 'j+1' Line 3
...	...	...

New Frame + 1 will be generated by Frame 'j+2' and Frame 'j+3' and so on

It accepts the following optional parameters:

`scan`

This determines whether the interlaced frame is taken from the even (tff - default) or odd (bff) lines of the progressive frame.

lowpass

Enable (default) or disable the vertical lowpass filter to avoid twitter interlacing and reduce moire patterns.

## 37.67 kerndeint# TOC

Deinterlace input video by applying Donald Graft's adaptive kernel deinterling. Work on interlaced parts of a video to produce progressive frames.

The description of the accepted parameters follows.

thresh

Set the threshold which affects the filter's tolerance when determining if a pixel line must be processed. It must be an integer in the range [0,255] and defaults to 10. A value of 0 will result in applying the process on every pixels.

map

Paint pixels exceeding the threshold value to white if set to 1. Default is 0.

order

Set the fields order. Swap fields if set to 1, leave fields alone if 0. Default is 0.

sharp

Enable additional sharpening if set to 1. Default is 0.

twoway

Enable twoway sharpening if set to 1. Default is 0.

### 37.67.1 Examples# TOC

- Apply default values:

```
kerndeint=thresh=10:map=0:order=0:sharp=0:twoway=0
```

- Enable additional sharpening:

```
kerndeint=sharp=1
```

- Paint processed pixels in white:

```
kerndeint=map=1
```



## 37.68 lenscorrection# TOC

### Correct radial lens distortion

This filter can be used to correct for radial distortion as can result from the use of wide angle lenses, and thereby re-rectify the image. To find the right parameters one can use tools available for example as part of opencv or simply trial-and-error. To use opencv use the calibration sample (under samples/cpp) from the opencv sources and extract the k1 and k2 coefficients from the resulting matrix.

Note that effectively the same filter is available in the open-source tools Krita and Digikam from the KDE project.

In contrast to the vignette filter, which can also be used to compensate lens errors, this filter corrects the distortion of the image, whereas vignette corrects the brightness distribution, so you may want to use both filters together in certain cases, though you will have to take care of ordering, i.e. whether vignetting should be applied before or after lens correction.

### 37.68.1 Options# TOC

The filter accepts the following options:

**cx**

Relative x-coordinate of the focal point of the image, and thereby the center of the distortion. This value has a range [0,1] and is expressed as fractions of the image width.

**cy**

Relative y-coordinate of the focal point of the image, and thereby the center of the distortion. This value has a range [0,1] and is expressed as fractions of the image height.

**k1**

Coefficient of the quadratic correction term. 0.5 means no correction.

**k2**

Coefficient of the double quadratic correction term. 0.5 means no correction.

The formula that generates the correction is:

$$r\_src = r\_tgt * (1 + k1 * (r\_tgt / r\_0)^2 + k2 * (r\_tgt / r\_0)^4)$$

where  $r\_0$  is halve of the image diagonal and  $r\_src$  and  $r\_tgt$  are the distances from the focal point in the source and target images, respectively.

## 37.69 lut3d# TOC

Apply a 3D LUT to an input video.

The filter accepts the following options:

`file`

Set the 3D LUT file name.

Currently supported formats:

‘3dl’

AfterEffects

‘cube’

Iridas

‘dat’

DaVinci

‘m3d’

Pandora

`interp`

Select interpolation mode.

Available values are:

‘nearest’

Use values from the nearest defined point.

‘trilinear’

Interpolate values using the 8 points defining a cube.

‘tetrahedral’

Interpolate values using a tetrahedron.

## 37.70 lut, lutrgb, lutyuv# TOC

Compute a look-up table for binding each pixel component input value to an output value, and apply it to the input video.

*lutyuv* applies a lookup table to a YUV input video, *lutrgb* to an RGB input video.

These filters accept the following parameters:

c0

set first pixel component expression

c1

set second pixel component expression

c2

set third pixel component expression

c3

set fourth pixel component expression, corresponds to the alpha component

r

set red component expression

g

set green component expression

b

set blue component expression

a

alpha component expression

y

set Y/luminance component expression

u

set U/Cb component expression

v

set V/Cr component expression

Each of them specifies the expression to use for computing the lookup table for the corresponding pixel component values.

The exact component associated to each of the *c*\* options depends on the format in input.

The *lut* filter requires either YUV or RGB pixel formats in input, *lutrgb* requires RGB pixel formats in input, and *lutyuv* requires YUV.

The expressions can contain the following constants and functions:

w

h

The input width and height.

val

The input value for the pixel component.

clipval

The input value, clipped to the *minval-maxval* range.

maxval

The maximum value for the pixel component.

minval

The minimum value for the pixel component.

negval

The negated value for the pixel component value, clipped to the *minval-maxval* range; it corresponds to the expression "maxval-clipval+minval".

clip(val)

The computed value in *val*, clipped to the *minval-maxval* range.

gammaval(gamma)

The computed gamma correction value of the pixel component value, clipped to the *minval-maxval* range. It corresponds to the expression

"pow((clipval-minval)/(maxval-minval),gamma)\*(maxval-minval)+minval"

All expressions default to "val".

### 37.70.1 Examples# TOC

- Negate input video:

```
lutrgb="r=maxval+minval-val:g=maxval+minval-val:b=maxval+minval-val"  
lutyuv="y=maxval+minval-val:u=maxval+minval-val:v=maxval+minval-val"
```

The above is the same as:

```
lutrgb="r=negval:g=negval:b=negval"  
lutyuv="y=negval:u=negval:v=negval"
```

- Negate luminance:

```
lutyuv=y=negval
```

- Remove chroma components, turning the video into a graytone image:

```
lutyuv="u=128:v=128"
```

- Apply a luma burning effect:

```
lutyuv="y=2*val"
```

- Remove green and blue components:

```
lutrgb="g=0:b=0"
```

- Set a constant alpha channel value on input:

```
format=rgba,lutrgb=a="maxval-minval/2"
```

- Correct luminance gamma by a factor of 0.5:

```
lutyuv=y=gammaval(0.5)
```

- Discard least significant bits of luma:

```
lutyuv=y='bitand(val, 128+64+32)'
```

### 37.71 mergeplanes# TOC

Merge color channel components from several video streams.

The filter accepts up to 4 input streams, and merge selected input planes to the output video.

This filter accepts the following options:

mapping

Set input to output plane mapping. Default is 0.

The mappings is specified as a bitmap. It should be specified as a hexadecimal number in the form 0xAa[Bb[Cc[Dd]]]. 'Aa' describes the mapping for the first plane of the output stream. 'A' sets the number of the input stream to use (from 0 to 3), and 'a' the plane number of the corresponding input to use (from 0 to 3). The rest of the mappings is similar, 'Bb' describes the mapping for the output stream second plane, 'Cc' describes the mapping for the output stream third plane and 'Dd' describes the mapping for the output stream fourth plane.

format

Set output pixel format. Default is yuva444p.

### 37.71.1 Examples# TOC

- Merge three gray video streams of same width and height into single video stream:

```
[a0][a1][a2]mergeplanes=0x001020:yuv444p
```

- Merge 1st yuv444p stream and 2nd gray video stream into yuva444p video stream:

```
[a0][a1]mergeplanes=0x00010210:yuva444p
```

- Swap Y and A plane in yuva444p stream:

```
format=yuva444p,mergeplanes=0x03010200:yuva444p
```

- Swap U and V plane in yuv420p stream:

```
format=yuv420p,mergeplanes=0x000201:yuv420p
```

- Cast a rgb24 clip to yuv444p:

```
format=rgb24,mergeplanes=0x000102:yuv444p
```

### 37.72 mcdeint# TOC

Apply motion-compensation deinterlacing.

It needs one field per frame as input and must thus be used together with yadif=1/3 or equivalent.

This filter accepts the following options:

mode

Set the deinterlacing mode.

It accepts one of the following values:

`'fast'`  
`'medium'`  
`'slow'`

use iterative motion estimation

`'extra_slow'`

like `'slow'`, but use multiple reference frames.

Default value is `'fast'`.

parity

Set the picture field parity assumed for the input video. It must be one of the following values:

`'0, tff'`

assume top field first

`'1, bff'`

assume bottom field first

Default value is `'bff'`.

qp

Set per-block quantization parameter (QP) used by the internal encoder.

Higher values should result in a smoother motion vector field but less optimal individual vectors.

Default value is 1.

## 37.73 mpdecimate# TOC

Drop frames that do not differ greatly from the previous frame in order to reduce frame rate.

The main use of this filter is for very-low-bitrate encoding (e.g. streaming over dialup modem), but it could in theory be used for fixing movies that were inverse-telecined incorrectly.

A description of the accepted options follows.

max

Set the maximum number of consecutive frames which can be dropped (if positive), or the minimum interval between dropped frames (if negative). If the value is 0, the frame is dropped unregarding the number of previous sequentially dropped frames.

Default value is 0.

`hi`  
`lo`  
`frac`

Set the dropping threshold values.

Values for `hi` and `lo` are for 8x8 pixel blocks and represent actual pixel value differences, so a threshold of 64 corresponds to 1 unit of difference for each pixel, or the same spread out differently over the block.

A frame is a candidate for dropping if no 8x8 blocks differ by more than a threshold of `hi`, and if no more than `frac` blocks (1 meaning the whole image) differ by more than a threshold of `lo`.

Default value for `hi` is 64\*12, default value for `lo` is 64\*5, and default value for `frac` is 0.33.

## 37.74 negate# TOC

Negate input video.

It accepts an integer in input; if non-zero it negates the alpha component (if available). The default value in input is 0.

## 37.75 noformat# TOC

Force libavfilter not to use any of the specified pixel formats for the input to the next filter.

It accepts the following parameters:

`pix_fmts`

A `'|'`-separated list of pixel format names, such as `apix_fmts=yuv420p|monow|rgb24`".

### 37.75.1 Examples# TOC

- Force libavfilter to use a format different from *yuv420p* for the input to the *vflip* filter:

```
noformat=pix_fmts=yuv420p,vflip
```

- Convert the input video to any of the formats not contained in the list:

```
noformat=yuv420p|yuv444p|yuv410p
```



## 37.76 noise# TOC

Add noise on video input frame.

The filter accepts the following options:

`all_seed`  
`c0_seed`  
`c1_seed`  
`c2_seed`  
`c3_seed`

Set noise seed for specific pixel component or all pixel components in case of *all\_seed*. Default value is 123457.

`all_strength, alls`  
`c0_strength, c0s`  
`c1_strength, c1s`  
`c2_strength, c2s`  
`c3_strength, c3s`

Set noise strength for specific pixel component or all pixel components in case *all\_strength*. Default value is 0. Allowed range is [0, 100].

`all_flags, allf`  
`c0_flags, c0f`  
`c1_flags, c1f`  
`c2_flags, c2f`  
`c3_flags, c3f`

Set pixel component flags or set flags for all components if *all\_flags*. Available values for component flags are:

‘a’

averaged temporal noise (smoother)

‘p’

mix random noise with a (semi)regular pattern

‘t’

temporal noise (noise pattern changes between frames)

‘u’

uniform noise (gaussian otherwise)

### 37.76.1 Examples# TOC

Add temporal and uniform noise to input video:

```
noise=all; s=20; allf=t+u
```

### 37.77 null# TOC

Pass the video source unchanged to the output.

### 37.78 ocv# TOC

Apply a video transform using libopencv.

To enable this filter, install the libopencv library and headers and configure FFmpeg with `--enable-libopencv`.

It accepts the following parameters:

`filter_name`

The name of the libopencv filter to apply.

`filter_params`

The parameters to pass to the libopencv filter. If not specified, the default values are assumed.

Refer to the official libopencv documentation for more precise information:

<http://docs.opencv.org/master/modules/imgproc/doc/filtering.html>

Several libopencv filters are supported; see the following subsections.

#### 37.78.1 dilate# TOC

Dilate an image by using a specific structuring element. It corresponds to the libopencv function `cvDilate`.

It accepts the parameters: *struct\_el* *nb\_iterations*.

*struct\_el* represents a structuring element, and has the syntax: *cols**x**rows*+*anchor\_x**anchor\_y*/*shape*

*cols* and *rows* represent the number of columns and rows of the structuring element, *anchor\_x* and *anchor\_y* the anchor point, and *shape* the shape for the structuring element. *shape* must be "rect", "cross", "ellipse", or "custom".

If the value for *shape* is "custom", it must be followed by a string of the form "*=filename*". The file with name *filename* is assumed to represent a binary image, with each printable character corresponding to a bright pixel. When a custom *shape* is used, *cols* and *rows* are ignored, the number of columns and rows of the read file are assumed instead.

The default value for *struct\_el* is "3x3+0x0/rect".

*nb\_iterations* specifies the number of times the transform is applied to the image, and defaults to 1.

Some examples:

```
# Use the default values
ocv=dilate

# Dilate using a structuring element with a 5x5 cross, iterating two times
ocv=filter_name=dilate:filter_params=5x5+2x2/cross|2

# Read the shape from the file diamond.shape, iterating two times.
# The file diamond.shape may contain a pattern of characters like this
#   *
#  ***
# *****
#  ***
#   *
# The specified columns and rows are ignored
# but the anchor point coordinates are not
ocv=dilate:0x0+2x2/custom=diamond.shape|2
```

### 37.78.2 erode# TOC

Erode an image by using a specific structuring element. It corresponds to the libopencv function `cvErode`.

It accepts the parameters: *struct\_el:nb\_iterations*, with the same syntax and semantics as the dilate filter.

### 37.78.3 smooth# TOC

Smooth the input video.

The filter takes the following parameters: *type|param1|param2|param3|param4*.

*type* is the type of smooth filter to apply, and must be one of the following values: "blur", "blur\_no\_scale", "median", "gaussian", or "bilateral". The default value is "gaussian".

The meaning of *param1*, *param2*, *param3*, and *param4* depend on the smooth type. *param1* and *param2* accept integer positive values or 0. *param3* and *param4* accept floating point values.

The default value for *param1* is 3. The default value for the other parameters is 0.

These parameters correspond to the parameters assigned to the libopencv function `cvSmooth`.

## 37.79 overlay# TOC

Overlay one video on top of another.

It takes two inputs and has one output. The first input is the "main" video on which the second input is overlaid.

It accepts the following parameters:

A description of the accepted options follows.

`x`

`y`

Set the expression for the x and y coordinates of the overlaid video on the main video. Default value is "0" for both expressions. In case the expression is invalid, it is set to a huge value (meaning that the overlay will not be displayed within the output visible area).

`eof_action`

The action to take when EOF is encountered on the secondary input; it accepts one of the following values:

`repeat`

Repeat the last frame (the default).

`endall`

End both streams.

`pass`

Pass the main input through.

`eval`

Set when the expressions for x, and y are evaluated.

It accepts the following values:

`'init'`

only evaluate expressions once during the filter initialization or when a command is processed

`'frame'`

evaluate expressions for each incoming frame

Default value is 'frame'.

`shortest`

If set to 1, force the output to terminate when the shortest input terminates. Default value is 0.

`format`

Set the format for the output video.

It accepts the following values:

'yuv420'

force YUV420 output

'yuv422'

force YUV422 output

'yuv444'

force YUV444 output

'rgb'

force RGB output

Default value is 'yuv420'.

`rgb` (*deprecated*)

If set to 1, force the filter to accept inputs in the RGB color space. Default value is 0. This option is deprecated, use `format` instead.

`repeatlast`

If set to 1, force the filter to draw the last overlay frame over the main input until the end of the stream. A value of 0 disables this behavior. Default value is 1.

The `x`, and `y` expressions can contain the following parameters.

`main_w`, `W`

`main_h`, `H`

The main input width and height.

overlay\_w, w  
overlay\_h, h

The overlay input width and height.

x  
y

The computed values for  $x$  and  $y$ . They are evaluated for each new frame.

hsub  
vsub

horizontal and vertical chroma subsample values of the output format. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

n

the number of input frame, starting from 0

pos

the position in the file of the input frame, NAN if unknown

t

The timestamp, expressed in seconds. It's NAN if the input timestamp is unknown.

Note that the  $n$ ,  $pos$ ,  $t$  variables are available only when evaluation is done *per frame*, and will evaluate to NAN when `eval` is set to 'init'.

Be aware that frames are taken from each input video in timestamp order, hence, if their initial timestamps differ, it is a good idea to pass the two inputs through a *setpts=PTS-STARTPTS* filter to have them begin in the same zero timestamp, as the example for the *movie* filter does.

You can chain together more overlays but you should test the efficiency of such approach.

### 37.79.1 Commands# TOC

This filter supports the following commands:

x  
y

Modify the  $x$  and  $y$  of the overlay input. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.79.2 Examples# TOC

- Draw the overlay at 10 pixels from the bottom right corner of the main video:

```
overlay=main_w-overlay_w-10:main_h-overlay_h-10
```

Using named options the example above becomes:

```
overlay=x=main_w-overlay_w-10:y=main_h-overlay_h-10
```

- Insert a transparent PNG logo in the bottom left corner of the input, using the `ffmpeg` tool with the `-filter_complex` option:

```
ffmpeg -i input -i logo -filter_complex 'overlay=10:main_h-overlay_h-10' output
```

- Insert 2 different transparent PNG logos (second logo on bottom right corner) using the `ffmpeg` tool:

```
ffmpeg -i input -i logo1 -i logo2 -filter_complex 'overlay=x=10:y=H-h-10,overlay=x=W-w-10:y=H-h-10' output
```

- Add a transparent color layer on top of the main video; `WxH` must specify the size of the main input to the overlay filter:

```
color=color=red@.3:size=WxH [over]; [in][over] overlay [out]
```

- Play an original video and a filtered version (here with the `deshake` filter) side by side using the `ffplay` tool:

```
ffplay input.avi -vf 'split[a][b]; [a]pad=iw*2:ih[src]; [b]deshake[filt]; [src][filt]overlay=w'
```

The above command is the same as:

```
ffplay input.avi -vf 'split[b], pad=iw*2[src], [b]deshake, [src]overlay=w'
```

- Make a sliding overlay appearing from the left to the right top part of the screen starting since time 2:

```
overlay=x='if(gte(t,2), -w+(t-2)*20, NAN)':y=0
```

- Compose output by putting two input videos side to side:

```
ffmpeg -i left.avi -i right.avi -filter_complex "  
nullsrc=size=200x100 [background];  
[0:v] setpts=PTS-STARTPTS, scale=100x100 [left];  
[1:v] setpts=PTS-STARTPTS, scale=100x100 [right];  
[background][left] overlay=shortest=1 [background+left];  
[background+left][right] overlay=shortest=1:x=100 [left+right]  
"
```

- Mask 10-20 seconds of a video by applying the `delogo` filter to a section

```
ffmpeg -i test.avi -codec:v:0 h264 -ar 11025 -b:v 9000k
-vf '[in]split[split_main][split_delogo];[split_delogo]trim=start=360:end=371,delogo=0:0:640:480[delogoed];[split_main][delogoed]overlay=eof_action=pass[out]'
masked.avi
```

- Chain several overlays in cascade:

```
nullsrc=s=200x200 [bg];
testsrc=s=100x100, split=4 [in0][in1][in2][in3];
[in0] lutrgb=r=0, [bg] overlay=0:0 [mid0];
[in1] lutrgb=g=0, [mid0] overlay=100:0 [mid1];
[in2] lutrgb=b=0, [mid1] overlay=0:100 [mid2];
[in3] null, [mid2] overlay=100:100 [out0]
```

## 37.80 owdenoise# TOC

Apply Overcomplete Wavelet denoiser.

The filter accepts the following options:

`depth`

Set depth.

Larger depth values will denoise lower frequency components more, but slow down filtering.

Must be an int in the range 8-16, default is 8.

`luma_strength, ls`

Set luma strength.

Must be a double value in the range 0-1000, default is 1 . 0.

`chroma_strength, cs`

Set chroma strength.

Must be a double value in the range 0-1000, default is 1 . 0.

## 37.81 pad# TOC

Add paddings to the input image, and place the original input at the provided *x*, *y* coordinates.

It accepts the following parameters:

`width, w`

`height, h`

Specify an expression for the size of the output image with the paddings added. If the value for *width* or *height* is 0, the corresponding input size is used for the output.



The *width* expression can reference the value set by the *height* expression, and vice versa.

The default value of *width* and *height* is 0.

*x*

*y*

Specify the offsets to place the input image at within the padded area, with respect to the top/left border of the output image.

The *x* expression can reference the value set by the *y* expression, and vice versa.

The default value of *x* and *y* is 0.

*color*

Specify the color of the padded area. For the syntax of this option, check the "Color" section in the ffmpeg-utils manual.

The default value of *color* is "black".

The value for the *width*, *height*, *x*, and *y* options are expressions containing the following constants:

*in\_w*

*in\_h*

The input video width and height.

*iw*

*ih*

These are the same as *in\_w* and *in\_h*.

*out\_w*

*out\_h*

The output width and height (the size of the padded area), as specified by the *width* and *height* expressions.

*ow*

*oh*

These are the same as *out\_w* and *out\_h*.

*x*

*y*

The x and y offsets as specified by the *x* and *y* expressions, or NAN if not yet specified.

a

same as *iw / ih*

sar

input sample aspect ratio

dar

input display aspect ratio, it is the same as  $(iw / ih) * sar$

hsub

vsub

The horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

### 37.81.1 Examples# TOC

- Add paddings with the color "violet" to the input video. The output video size is 640x480, and the top-left corner of the input video is placed at column 0, row 40

```
pad=640:480:0:40:violet
```

The example above is equivalent to the following command:

```
pad=width=640:height=480:x=0:y=40:color=violet
```

- Pad the input to get an output with dimensions increased by 3/2, and put the input video at the center of the padded area:

```
pad="3/2*iw:3/2*ih:(ow-iw)/2:(oh-ih)/2"
```

- Pad the input to get a squared output with size equal to the maximum value between the input width and height, and put the input video at the center of the padded area:

```
pad="max(iw\,ih):ow:(ow-iw)/2:(oh-ih)/2"
```

- Pad the input to get a final w/h ratio of 16:9:

```
pad="ih*16/9:ih:(ow-iw)/2:(oh-ih)/2"
```

- In case of anamorphic video, in order to set the output display aspect correctly, it is necessary to use *sar* in the expression, according to the relation:

```
(ih * X / ih) * sar = output_dar  
X = output_dar / sar
```

Thus the previous example needs to be modified to:

```
pad="ih*16/9/sar:ih:(ow-iw)/2:(oh-ih)/2"
```

- Double the output size and put the input video in the bottom-right corner of the output padded area:

```
pad="2*iw:2*ih:ow-iw:oh-ih"
```

## 37.82 palettegen# TOC

Generate one palette for a whole video stream.

It accepts the following options:

`max_colors`

Set the maximum number of colors to quantize in the palette. Note: the palette will still contain 256 colors; the unused palette entries will be black.

`reserve_transparent`

Create a palette of 255 colors maximum and reserve the last one for transparency. Reserving the transparency color is useful for GIF optimization. If not set, the maximum of colors in the palette will be 256. You probably want to disable this option for a standalone image. Set by default.

`stats_mode`

Set statistics mode.

It accepts the following values:

`'full'`

Compute full frame histograms.

`'diff'`

Compute histograms only for the part that differs from previous frame. This might be relevant to give more importance to the moving part of your input if the background is static.

Default value is *full*.

The filter also exports the frame metadata `lavfi.color_quant_ratio(nb_color_in / nb_color_out)` which you can use to evaluate the degree of color quantization of the palette. This information is also visible at *info* logging level.

## 37.82.1 Examples# TOC

- Generate a representative palette of a given video using `ffmpeg`:

```
ffmpeg -i input.mkv -vf palettegen palette.png
```

## 37.83 paletteuse# TOC

Use a palette to downsample an input video stream.

The filter takes two inputs: one video stream and a palette. The palette must be a 256 pixels image.

It accepts the following options:

`dither`

Select dithering mode. Available algorithms are:

`'bayer'`

Ordered 8x8 bayer dithering (deterministic)

`'heckbert'`

Dithering as defined by Paul Heckbert in 1982 (simple error diffusion). Note: this dithering is sometimes considered "wrong" and is included as a reference.

`'floyd_steinberg'`

Floyd and Steingberg dithering (error diffusion)

`'sierra2'`

Frankie Sierra dithering v2 (error diffusion)

`'sierra2_4a'`

Frankie Sierra dithering v2 "Lite" (error diffusion)

Default is *sierra2\_4a*.

`bayer_scale`

When *bayer* dithering is selected, this option defines the scale of the pattern (how much the crosshatch pattern is visible). A low value means more visible pattern for less banding, and higher value means less visible pattern at the cost of more banding.

The option must be an integer value in the range [0,5]. Default is 2.

`diff_mode`

If set, define the zone to process

`'rectangle'`

Only the changing rectangle will be reprocessed. This is similar to GIF cropping/offsetting compression mechanism. This option can be useful for speed if only a part of the image is changing, and has use cases such as limiting the scope of the error diffusal dither to the rectangle that bounds the moving scene (it leads to more deterministic output if the scene doesn't change much, and as a result less moving noise and better GIF compression).

Default is *none*.

### 37.83.1 Examples# TOC

- Use a palette (generated for example with `palettegen`) to encode a GIF using `ffmpeg`:

```
ffmpeg -i input.mkv -i palette.png -lavfi paletteuse output.gif
```

### 37.84 perspective# TOC

Correct perspective of video not recorded perpendicular to the screen.

A description of the accepted parameters follows.

`x0`  
`y0`  
`x1`  
`y1`  
`x2`  
`y2`  
`x3`  
`y3`

Set coordinates expression for top left, top right, bottom left and bottom right corners. Default values are `0:0:W:0:0:H:W:H` with which perspective will remain unchanged. If the `sense` option is set to `source`, then the specified points will be sent to the corners of the destination. If the `sense` option is set to `destination`, then the corners of the source will be sent to the specified coordinates.

The expressions can use the following variables:

`W`  
`H`

the width and height of video frame.

interpolation

Set interpolation for perspective correction.

It accepts the following values:

‘linear’

‘cubic’

Default value is ‘linear’.

sense

Set interpretation of coordinate options.

It accepts the following values:

‘0, source’

Send point in the source specified by the given coordinates to the corners of the destination.

‘1, destination’

Send the corners of the source to the point in the destination specified by the given coordinates.

Default value is ‘source’.

## **37.85 phase# TOC**

Delay interlaced video by one field time so that the field order changes.

The intended use is to fix PAL movies that have been captured with the opposite field order to the film-to-video transfer.

A description of the accepted parameters follows.

mode

Set phase mode.

It accepts the following values:

‘t’

Capture field order top-first, transfer bottom-first. Filter will delay the bottom field.

‘b’

Capture field order bottom-first, transfer top-first. Filter will delay the top field.

‘p’

Capture and transfer with the same field order. This mode only exists for the documentation of the other options to refer to, but if you actually select it, the filter will faithfully do nothing.

‘a’

Capture field order determined automatically by field flags, transfer opposite. Filter selects among ‘t’ and ‘b’ modes on a frame by frame basis using field flags. If no field information is available, then this works just like ‘u’.

‘u’

Capture unknown or varying, transfer opposite. Filter selects among ‘t’ and ‘b’ on a frame by frame basis by analyzing the images and selecting the alternative that produces best match between the fields.

‘T’

Capture top-first, transfer unknown or varying. Filter selects among ‘t’ and ‘p’ using image analysis.

‘B’

Capture bottom-first, transfer unknown or varying. Filter selects among ‘b’ and ‘p’ using image analysis.

‘A’

Capture determined by field flags, transfer unknown or varying. Filter selects among ‘t’, ‘b’ and ‘p’ using field flags and image analysis. If no field information is available, then this works just like ‘U’. This is the default mode.

‘U’

Both capture and transfer unknown or varying. Filter selects among ‘t’, ‘b’ and ‘p’ using image analysis only.

## 37.86 pixdesctest# TOC

Pixel format descriptor test filter, mainly useful for internal testing. The output video should be equal to the input video.

For example:

```
format=monow, pixdesctest
```

can be used to test the monowhite pixel format descriptor definition.

## 37.87 pp# TOC

Enable the specified chain of postprocessing subfilters using libpostproc. This library should be automatically selected with a GPL build (`--enable-gpl`). Subfilters must be separated by '/' and can be disabled by prepending a '-'. Each subfilter and some options have a short and a long name that can be used interchangeably, i.e. `dr`/`dering` are the same.

The filters accept the following options:

```
subfilters
```

Set postprocessing subfilters string.

All subfilters share common options to determine their scope:

```
a/autoq
```

Honor the quality commands for this subfilter.

```
c/chrom
```

Do chrominance filtering, too (default).

```
y/nochrom
```

Do luminance filtering only (no chrominance).

```
n/noluma
```

Do chrominance filtering only (no luminance).

These options can be appended after the subfilter name, separated by a '|'.

Available subfilters are:

```
hb/hdeblock[ |difference[ |flatness]]
```

Horizontal deblocking filter

```
difference
```

Difference factor where higher values mean more deblocking (default: 32).



flatness

Flatness threshold where lower values mean more deblocking (default: 39).

vb/vdeblock[ |difference[ |flatness]]

Vertical deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

ha/hadeblock[ |difference[ |flatness]]

Accurate horizontal deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

va/vadeblock[ |difference[ |flatness]]

Accurate vertical deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

The horizontal and vertical deblocking filters share the difference and flatness values so you cannot set different horizontal and vertical thresholds.

h1/x1hdeblock

Experimental horizontal deblocking filter

v1/x1vdeblock

## Experimental vertical deblocking filter

dr/dering

### Deringing filter

tn/tmpnoise[|threshold1[|threshold2[|threshold3]]], temporal noise reducer

threshold1

larger -> stronger filtering

threshold2

larger -> stronger filtering

threshold3

larger -> stronger filtering

al/autolevels[:f/fullyrange], automatic brightness / contrast correction

f/fullyrange

Stretch luminance to 0-255.

lb/linblenddeint

Linear blend deinterlacing filter that deinterlaces the given block by filtering all lines with a (1 2 1) filter.

li/linipoldeint

Linear interpolating deinterlacing filter that deinterlaces the given block by linearly interpolating every second line.

ci/cubicipoldeint

Cubic interpolating deinterlacing filter deinterlaces the given block by cubically interpolating every second line.

md/mediandeint

Median deinterlacing filter that deinterlaces the given block by applying a median filter to every second line.

fd/ffmpegdeint

FFmpeg deinterlacing filter that deinterlaces the given block by filtering every second line with a  $(-1 \ 4 \ 2 \ 4 \ -1)$  filter.

l5/lowpass5

Vertically applied FIR lowpass deinterlacing filter that deinterlaces the given block by filtering all lines with a  $(-1 \ 2 \ 6 \ 2 \ -1)$  filter.

fq/forceQuant[|quantizer]

Overrides the quantizer table from the input with the constant quantizer you specify.

quantizer

Quantizer to use

de/default

Default pp filter combination (hb|a,vb|a,dr|a)

fa/fast

Fast pp filter combination (h1|a,v1|a,dr|a)

ac

High quality pp filter combination (ha|a|128|7,va|a,dr|a)

### 37.87.1 Examples# TOC

- Apply horizontal and vertical deblocking, deringing and automatic brightness/contrast:

pp=hb/vb/dr/al

- Apply default filters without brightness/contrast correction:

pp=de/-al

- Apply default filters and temporal denoiser:

pp=default/tmpnoise|1|2|3

- Apply deblocking on luminance only, and switch vertical deblocking on or off automatically depending on available CPU time:

pp=hb|y/vb|a

## 37.88 pp7# TOC

Apply Postprocessing filter 7. It is variant of the spp filter, similar to spp = 6 with 7 point DCT, where only the center sample is used after IDCT.

The filter accepts the following options:

qp

Force a constant quantization parameter. It accepts an integer in range 0 to 63. If not set, the filter will use the QP from the video stream (if available).

mode

Set thresholding mode. Available modes are:

'hard'

Set hard thresholding.

'soft'

Set soft thresholding (better de-ringing effect, but likely blurrier).

'medium'

Set medium thresholding (good results, default).

## 37.89 psnr# TOC

Obtain the average, maximum and minimum PSNR (Peak Signal to Noise Ratio) between two input videos.

This filter takes in input two input videos, the first input is considered the "main" source and is passed unchanged to the output. The second input is used as a "reference" video for computing the PSNR.

Both video inputs must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The obtained average PSNR is printed through the logging system.

The filter stores the accumulated MSE (mean squared error) of each frame, and at the end of the processing it is averaged across all frames equally, and the following formula is applied to obtain the PSNR:

$$\text{PSNR} = 10 \cdot \log_{10}(\text{MAX}^2 / \text{MSE})$$

Where MAX is the average of the maximum values of each component of the image.

The description of the accepted parameters follows.

`stats_file, f`

If specified the filter will use the named file to save the PSNR of each individual frame.

The file printed if *stats\_file* is selected, contains a sequence of key/value pairs of the form *key:value* for each compared couple of frames.

A description of each shown parameter follows:

`n`

sequential number of the input frame, starting from 1

`mse_avg`

Mean Square Error pixel-by-pixel average difference of the compared frames, averaged over all the image components.

`mse_y, mse_u, mse_v, mse_r, mse_g, mse_g, mse_a`

Mean Square Error pixel-by-pixel average difference of the compared frames for the component specified by the suffix.

`psnr_y, psnr_u, psnr_v, psnr_r, psnr_g, psnr_b, psnr_a`

Peak Signal to Noise ratio of the compared frames for the component specified by the suffix.

For example:

```
movie=ref_movie.mpg, setpts=PTS-STARTPTS [main];  
[main][ref] psnr="stats_file=stats.log" [out]
```

On this example the input file being processed is compared with the reference file `ref_movie.mpg`. The PSNR of each individual frame is stored in `stats.log`.

## 37.90 pullup# TOC

Pulldown reversal (inverse telecine) filter, capable of handling mixed hard-telecine, 24000/1001 fps progressive, and 30000/1001 fps progressive content.

The pullup filter is designed to take advantage of future context in making its decisions. This filter is stateless in the sense that it does not lock onto a pattern to follow, but it instead looks forward to the following fields in order to identify matches and rebuild progressive frames.

To produce content with an even framerate, insert the fps filter after pullup, use  $\text{fps}=24000/1001$  if the input frame rate is 29.97fps,  $\text{fps}=24$  for 30fps and the (rare) telecined 25fps input.

The filter accepts the following options:

j1  
jr  
jt  
jb

These options set the amount of "junk" to ignore at the left, right, top, and bottom of the image, respectively. Left and right are in units of 8 pixels, while top and bottom are in units of 2 lines. The default is 8 pixels on each side.

sb

Set the strict breaks. Setting this option to 1 will reduce the chances of filter generating an occasional mismatched frame, but it may also cause an excessive number of frames to be dropped during high motion sequences. Conversely, setting it to -1 will make filter match fields more easily. This may help processing of video where there is slight blurring between the fields, but may also cause there to be interlaced frames in the output. Default value is 0.

mp

Set the metric plane to use. It accepts the following values:

'l'

Use luma plane.

'u'

Use chroma blue plane.

'v'

Use chroma red plane.

This option may be set to use chroma plane instead of the default luma plane for doing filter's computations. This may improve accuracy on very clean source material, but more likely will decrease accuracy, especially if there is chroma noise (rainbow effect) or any grayscale video. The main purpose of setting mp to a chroma plane is to reduce CPU load and make pullup usable in realtime on slow machines.

For best results (without duplicated frames in the output file) it is necessary to change the output frame rate. For example, to inverse telecine NTSC input:

```
ffmpeg -i input -vf pullup -r 24000/1001 ...
```

## 37.91 qp# TOC

Change video quantization parameters (QP).

The filter accepts the following option:

**qp**

Set expression for quantization parameter.

The expression is evaluated through the eval API and can contain, among others, the following constants:

*known*

1 if index is not 129, 0 otherwise.

*qp*

Sequential index starting from -129 to 128.

### 37.91.1 Examples# TOC

- Some equation like:

```
qp=2+2*sin(PI*qp)
```

## 37.92 random# TOC

Flush video frames from internal cache of frames into a random order. No frame is discarded. Inspired by frei0r nervous filter.

**frames**

Set size in number of frames of internal cache, in range from 2 to 512. Default is 30.

**seed**

Set seed for random number generator, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to less than 0, the filter will try to use a good random seed on a best effort basis.

## 37.93 removegrain# TOC

The removegrain filter is a spatial denoiser for progressive video.

m0

Set mode for the first plane.

m1

Set mode for the second plane.

m2

Set mode for the third plane.

m3

Set mode for the fourth plane.

Range of mode is from 0 to 24. Description of each mode follows:

0

Leave input plane unchanged. Default.

1

Clips the pixel with the minimum and maximum of the 8 neighbour pixels.

2

Clips the pixel with the second minimum and maximum of the 8 neighbour pixels.

3

Clips the pixel with the third minimum and maximum of the 8 neighbour pixels.

4

Clips the pixel with the fourth minimum and maximum of the 8 neighbour pixels. This is equivalent to a median filter.

5

Line-sensitive clipping giving the minimal change.

6

Line-sensitive clipping, intermediate.

7



Line-sensitive clipping, intermediate.

8

Line-sensitive clipping, intermediate.

9

Line-sensitive clipping on a line where the neighbours pixels are the closest.

10

Replaces the target pixel with the closest neighbour.

11

[1 2 1] horizontal and vertical kernel blur.

12

Same as mode 11.

13

Bob mode, interpolates top field from the line where the neighbours pixels are the closest.

14

Bob mode, interpolates bottom field from the line where the neighbours pixels are the closest.

15

Bob mode, interpolates top field. Same as 13 but with a more complicated interpolation formula.

16

Bob mode, interpolates bottom field. Same as 14 but with a more complicated interpolation formula.

17

Clips the pixel with the minimum and maximum of respectively the maximum and minimum of each pair of opposite neighbour pixels.

18

Line-sensitive clipping using opposite neighbours whose greatest distance from the current pixel is minimal.

19

Replaces the pixel with the average of its 8 neighbours.

20

Averages the 9 pixels ([1 1 1] horizontal and vertical blur).

21

Clips pixels using the averages of opposite neighbour.

22

Same as mode 21 but simpler and faster.

23

Small edge and halo removal, but reputed useless.

24

Similar as 23.

## 37.94 removelogo# TOC

Suppress a TV station logo, using an image file to determine which pixels comprise the logo. It works by filling in the pixels that comprise the logo with neighboring pixels.

The filter accepts the following options:

`filename, f`

Set the filter bitmap file, which can be any image format supported by libavformat. The width and height of the image file must match those of the video stream being processed.

Pixels in the provided bitmap image with a value of zero are not considered part of the logo, non-zero pixels are considered part of the logo. If you use white (255) for the logo and black (0) for the rest, you will be safe. For making the filter bitmap, it is recommended to take a screen capture of a black frame with the logo visible, and then using a threshold filter followed by the erode filter once or twice.

If needed, little splotches can be fixed manually. Remember that if logo pixels are not covered, the filter quality will be much reduced. Marking too many pixels as part of the logo does not hurt as much, but it will increase the amount of blurring needed to cover over the image and will destroy more information than necessary, and extra pixels will slow things down on a large logo.

## 37.95 repeatfields# TOC

This filter uses the repeat\_field flag from the Video ES headers and hard repeats fields based on its value.

## 37.96 reverse, areverse# TOC

Reverse a clip.

Warning: This filter requires memory to buffer the entire clip, so trimming is suggested.

### 37.96.1 Examples# TOC

- Take the first 5 seconds of a clip, and reverse it.

```
trim=end=5,reverse
```

## 37.97 rotate# TOC

Rotate video by an arbitrary angle expressed in radians.

The filter accepts the following options:

A description of the optional parameters follows.

angle, a

Set an expression for the angle by which to rotate the input video clockwise, expressed as a number of radians. A negative value will result in a counter-clockwise rotation. By default it is set to "0".

This expression is evaluated for each frame.

out\_w, ow

Set the output width expression, default value is "iw". This expression is evaluated just once during configuration.

out\_h, oh

Set the output height expression, default value is "ih". This expression is evaluated just once during configuration.

bilinear

Enable bilinear interpolation if set to 1, a value of 0 disables it. Default value is 1.

fillcolor, c

Set the color used to fill the output area not covered by the rotated image. For the general syntax of this option, check the "Color" section in the ffmpeg-utils manual. If the special value "none" is selected then no background is printed (useful for example if the background is never shown).

Default value is "black".

The expressions for the angle and the output size can contain the following constants and functions:

`n`

sequential number of the input frame, starting from 0. It is always NAN before the first frame is filtered.

`t`

time in seconds of the input frame, it is set to 0 when the filter is configured. It is always NAN before the first frame is filtered.

`hsub`

`vsub`

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

`in_w, iw`

`in_h, ih`

the input video width and height

`out_w, ow`

`out_h, oh`

the output width and height, that is the size of the padded area as specified by the *width* and *height* expressions

`rotw(a)`

`roth(a)`

the minimal width/height required for completely containing the input video rotated by *a* radians.

These are only available when computing the `out_w` and `out_h` expressions.

### 37.97.1 Examples# TOC

- Rotate the input by  $\text{PI}/6$  radians clockwise:

```
rotate=PI/6
```

- Rotate the input by  $\text{PI}/6$  radians counter-clockwise:

```
rotate=-PI/6
```

- Rotate the input by 45 degrees clockwise:

```
rotate=45*PI/180
```

- Apply a constant rotation with period T, starting from an angle of  $\pi/3$ :

```
rotate=PI/3+2*PI*t/T
```

- Make the input video rotation oscillating with a period of T seconds and an amplitude of A radians:

```
rotate=A*sin(2*PI/T*t)
```

- Rotate the video, output size is chosen so that the whole rotating input video is always completely contained in the output:

```
rotate=2*PI*t:ow=hypot(iw,ih):oh=ow'
```

- Rotate the video, reduce the output size so that no background is ever shown:

```
rotate=2*PI*t:ow='min(iw,ih)/sqrt(2)':oh=ow:c=none
```

### 37.97.2 Commands# TOC

The filter supports the following commands:

a, angle

Set the angle expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.98 sab# TOC

Apply Shape Adaptive Blur.

The filter accepts the following options:

luma\_radius, lr

Set luma blur filter strength, must be a value in range 0.1-4.0, default value is 1.0. A greater value will result in a more blurred image, and in slower processing.

luma\_pre\_filter\_radius, lpfr

Set luma pre-filter radius, must be a value in the 0.1-2.0 range, default value is 1.0.

luma\_strength, ls

Set luma maximum difference between pixels to still be considered, must be a value in the 0.1-100.0 range, default value is 1.0.

`chroma_radius, cr`

Set chroma blur filter strength, must be a value in range 0.1-4.0. A greater value will result in a more blurred image, and in slower processing.

`chroma_pre_filter_radius, cpfr`

Set chroma pre-filter radius, must be a value in the 0.1-2.0 range.

`chroma_strength, cs`

Set chroma maximum difference between pixels to still be considered, must be a value in the 0.1-100.0 range.

Each chroma option value, if not explicitly specified, is set to the corresponding luma option value.

## **37.99 scale# TOC**

Scale (resize) the input video, using the libswscale library.

The scale filter forces the output display aspect ratio to be the same of the input, by changing the output sample aspect ratio.

If the input image format is different from the format requested by the next filter, the scale filter will convert the input to the requested format.

### **37.99.1 Options# TOC**

The filter accepts the following options, or any of the options supported by the libswscale scaler.

See (ffmpeg-scaler)the ffmpeg-scaler manual for the complete list of scaler options.

`width, w`

`height, h`

Set the output video dimension expression. Default value is the input dimension.

If the value is 0, the input width is used for the output.

If one of the values is -1, the scale filter will use a value that maintains the aspect ratio of the input image, calculated from the other specified dimension. If both of them are -1, the input size is used

If one of the values is -n with  $n > 1$ , the scale filter will also use a value that maintains the aspect ratio of the input image, calculated from the other specified dimension. After that it will, however, make sure that the calculated dimension is divisible by n and adjust the value if necessary.

See below for the list of accepted constants for use in the dimension expression.

`interl`

Set the interlacing mode. It accepts the following values:

`'1'`

Force interlaced aware scaling.

`'0'`

Do not apply interlaced scaling.

`'-1'`

Select interlaced aware scaling depending on whether the source frames are flagged as interlaced or not.

Default value is `'0'`.

`flags`

Set libswscale scaling flags. See (ffmpeg-scaler)the ffmpeg-scaler manual for the complete list of values. If not explicitly specified the filter applies the default flags.

`size, s`

Set the video size. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

`in_color_matrix`

`out_color_matrix`

Set in/output YCbCr color space type.

This allows the autodetected value to be overridden as well as allows forcing a specific value used for the output and encoder.

If not specified, the color space type depends on the pixel format.

Possible values:

`'auto'`

Choose automatically.

`'bt709'`

Format conforming to International Telecommunication Union (ITU) Recommendation BT.709.

`'fcc'`

Set color space conforming to the United States Federal Communications Commission (FCC) Code of Federal Regulations (CFR) Title 47 (2003) 73.682 (a).

`'bt601'`

Set color space conforming to:

- ITU Radiocommunication Sector (ITU-R) Recommendation BT.601
- ITU-R Rec. BT.470-6 (1998) Systems B, B1, and G
- Society of Motion Picture and Television Engineers (SMPTE) ST 170:2004

`'smpte240m'`

Set color space conforming to SMPTE ST 240:1999.

`in_range`

`out_range`

Set in/output YCbCr sample range.

This allows the autodetected value to be overridden as well as allows forcing a specific value used for the output and encoder. If not specified, the range depends on the pixel format. Possible values:

`'auto'`

Choose automatically.

`'jpeg/full/pc'`

Set full range (0-255 in case of 8-bit luma).

`'mpeg/tv'`

Set "MPEG" range (16-235 in case of 8-bit luma).

`force_original_aspect_ratio`

Enable decreasing or increasing output video width or height if necessary to keep the original aspect ratio. Possible values:

`'disable'`

Scale the video as specified and disable this feature.

`'decrease'`



The output video dimensions will automatically be decreased if needed.

`'increase'`

The output video dimensions will automatically be increased if needed.

One useful instance of this option is that when you know a specific device's maximum allowed resolution, you can use this to limit the output video to that, while retaining the aspect ratio. For example, device A allows 1280x720 playback, and your video is 1920x800. Using this option (set it to decrease) and specifying 1280x720 to the command line makes the output 1280x533.

Please note that this is a different thing than specifying -1 for w or h, you still need to specify the output resolution for this option to work.

The values of the w and h options are expressions containing the following constants:

*in\_w*  
*in\_h*

The input width and height

*iw*  
*ih*

These are the same as *in\_w* and *in\_h*.

*out\_w*  
*out\_h*

The output (scaled) width and height

*ow*  
*oh*

These are the same as *out\_w* and *out\_h*

*a*

The same as *iw* / *ih*

*sar*

input sample aspect ratio

*dar*

The input display aspect ratio. Calculated from  $(iw / ih) * sar$ .

*hsub*  
*vsub*

horizontal and vertical input chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

*ohsub*  
*ovsub*

horizontal and vertical output chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

## 37.99.2 Examples# TOC

- Scale the input video to a size of 200x100

```
scale=w=200:h=100
```

This is equivalent to:

```
scale=200:100
```

or:

```
scale=200x100
```

- Specify a size abbreviation for the output size:

```
scale=qcif
```

which can also be written as:

```
scale=size=qcif
```

- Scale the input to 2x:

```
scale=w=2*iw:h=2*ih
```

- The above is the same as:

```
scale=2*in_w:2*in_h
```

- Scale the input to 2x with forced interlaced scaling:

```
scale=2*iw:2*ih:interl=1
```

- Scale the input to half size:

```
scale=w=iw/2:h=ih/2
```

- Increase the width, and set the height to the same size:

```
scale=3/2*iw:ow
```

- Seek Greek harmony:

```
scale=iw:1/PHI*iw  
scale=ih*PHI:ih
```

- Increase the height, and set the width to 3/2 of the height:

```
scale=w=3/2*oh:h=3/5*ih
```

- Increase the size, making the size a multiple of the chroma subsample values:

```
scale="trunc(3/2*iw/hsub)*hsub:trunc(3/2*ih/vsub)*vsub"
```

- Increase the width to a maximum of 500 pixels, keeping the same aspect ratio as the input:

```
scale=w='min(500\, iw*3/2):h=-1'
```

### 37.99.3 Commands# TOC

This filter supports the following commands:

```
width, w  
height, h
```

Set the output video dimension expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.100 scale2ref# TOC

Scale (resize) the input video, based on a reference video.

See the scale filter for available options, scale2ref supports the same but uses the reference video instead of the main input as basis.

#### 37.100.1 Examples# TOC

- Scale a subtitle stream to match the main video in size before overlaying

```
'scale2ref[b][a];[a][b]overlay'
```

### 37.101 separatefields# TOC

The `separatefields` takes a frame-based video input and splits each frame into its components fields, producing a new half height clip with twice the frame rate and twice the frame count.

This filter use field-dominance information in frame to decide which of each pair of fields to place first in the output. If it gets it wrong use `setfield` filter before `separatefields` filter.

## 37.102 `setdar`, `setsar`# TOC

The `setdar` filter sets the Display Aspect Ratio for the filter output video.

This is done by changing the specified Sample (aka Pixel) Aspect Ratio, according to the following equation:

$$DAR = HORIZONTAL\_RESOLUTION / VERTICAL\_RESOLUTION * SAR$$

Keep in mind that the `setdar` filter does not modify the pixel dimensions of the video frame. Also, the display aspect ratio set by this filter may be changed by later filters in the filterchain, e.g. in case of scaling or if another "setdar" or a "setsar" filter is applied.

The `setsar` filter sets the Sample (aka Pixel) Aspect Ratio for the filter output video.

Note that as a consequence of the application of this filter, the output display aspect ratio will change according to the equation above.

Keep in mind that the sample aspect ratio set by the `setsar` filter may be changed by later filters in the filterchain, e.g. if another "setsar" or a "setdar" filter is applied.

It accepts the following parameters:

`r`, `ratio`, `dar` (`setdar` only), `sar` (`setsar` only)

Set the aspect ratio used by the filter.

The parameter can be a floating point number string, an expression, or a string of the form *num:den*, where *num* and *den* are the numerator and denominator of the aspect ratio. If the parameter is not specified, it is assumed the value "0". In case the form "*num:den*" is used, the `:` character should be escaped.

`max`

Set the maximum integer value to use for expressing numerator and denominator when reducing the expressed aspect ratio to a rational. Default value is 100.

The parameter *sar* is an expression containing the following constants:

`E`, `PI`, `PHI`

These are approximated values for the mathematical constants *e* (Euler's number), *pi* (Greek pi), and *phi* (the golden ratio).

`w, h`

The input width and height.

`a`

These are the same as  $w / h$ .

`sar`

The input sample aspect ratio.

`dar`

The input display aspect ratio. It is the same as  $(w / h) * sar$ .

`hsub, vsub`

Horizontal and vertical chroma subsample values. For example, for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

### 37.102.1 Examples# TOC

- To change the display aspect ratio to 16:9, specify one of the following:

```
setdar=dar=1.77777
setdar=dar=16/9
setdar=dar=1.77777
```

- To change the sample aspect ratio to 10:11, specify:

```
setsar=sar=10/11
```

- To set a display aspect ratio of 16:9, and specify a maximum integer value of 1000 in the aspect ratio reduction, use the command:

```
setdar=ratio=16/9:max=1000
```

### 37.103 setfield# TOC

Force field for the output video frame.

The `setfield` filter marks the interlace type field for the output frames. It does not change the input frame, but only sets the corresponding property, which affects how the frame is treated by following filters (e.g. `fieldorder` or `yadif`).

The filter accepts the following options:

mode

Available values are:

‘auto’

Keep the same field property.

‘bff’

Mark the frame as bottom-field-first.

‘tff’

Mark the frame as top-field-first.

‘prog’

Mark the frame as progressive.

## 37.104 showinfo# TOC

Show a line containing various information for each input video frame. The input video is not modified.

The shown line contains a sequence of key/value pairs of the form *key:value*.

The following values are shown in the output:

n

The (sequential) number of the input frame, starting from 0.

pts

The Presentation TimeStamp of the input frame, expressed as a number of time base units. The time base unit depends on the filter input pad.

pts\_time

The Presentation TimeStamp of the input frame, expressed as a number of seconds.

pos

The position of the frame in the input stream, or -1 if this information is unavailable and/or meaningless (for example in case of synthetic video).

fmt

The pixel format name.

`sar`

The sample aspect ratio of the input frame, expressed in the form *num/den*.

`s`

The size of the input frame. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

`i`

The type of interlaced mode ("P" for "progressive", "T" for top field first, "B" for bottom field first).

`iskey`

This is 1 if the frame is a key frame, 0 otherwise.

`type`

The picture type of the input frame ("I" for an I-frame, "P" for a P-frame, "B" for a B-frame, or "?" for an unknown type). Also refer to the documentation of the `AVPictureType` enum and of the `av_get_picture_type_char` function defined in `libavutil/avutil.h`.

`checksum`

The Adler-32 checksum (printed in hexadecimal) of all the planes of the input frame.

`plane_checksum`

The Adler-32 checksum (printed in hexadecimal) of each plane of the input frame, expressed in the form "[*c0 c1 c2 c3*]".

## 37.105 showpalette# TOC

Displays the 256 colors palette of each frame. This filter is only relevant for *pal8* pixel format frames.

It accepts the following option:

`s`

Set the size of the box used to represent one palette color entry. Default is 30 (for a 30×30 pixel box).

## 37.106 shuffleplanes# TOC

Reorder and/or duplicate video planes.

It accepts the following parameters:

map0

The index of the input plane to be used as the first output plane.

map1

The index of the input plane to be used as the second output plane.

map2

The index of the input plane to be used as the third output plane.

map3

The index of the input plane to be used as the fourth output plane.

The first plane has the index 0. The default is to keep the input unchanged.

Swap the second and third planes of the input:

```
ffmpeg -i INPUT -vf shuffleplanes=0:2:1:3 OUTPUT
```

## 37.107 signalstats# TOC

Evaluate various visual metrics that assist in determining issues associated with the digitization of analog video media.

By default the filter will log these metadata values:

YMIN

Display the minimal Y value contained within the input frame. Expressed in range of [0-255].

YLOW

Display the Y value at the 10% percentile within the input frame. Expressed in range of [0-255].

YAVG

Display the average Y value within the input frame. Expressed in range of [0-255].

YHIGH



Display the Y value at the 90% percentile within the input frame. Expressed in range of [0-255].

YMAX

Display the maximum Y value contained within the input frame. Expressed in range of [0-255].

UMIN

Display the minimal U value contained within the input frame. Expressed in range of [0-255].

ULOW

Display the U value at the 10% percentile within the input frame. Expressed in range of [0-255].

UAVG

Display the average U value within the input frame. Expressed in range of [0-255].

UHIGH

Display the U value at the 90% percentile within the input frame. Expressed in range of [0-255].

UMAX

Display the maximum U value contained within the input frame. Expressed in range of [0-255].

VMIN

Display the minimal V value contained within the input frame. Expressed in range of [0-255].

VLOW

Display the V value at the 10% percentile within the input frame. Expressed in range of [0-255].

VAVG

Display the average V value within the input frame. Expressed in range of [0-255].

VHIGH

Display the V value at the 90% percentile within the input frame. Expressed in range of [0-255].

VMAX

Display the maximum V value contained within the input frame. Expressed in range of [0-255].

SATMIN

Display the minimal saturation value contained within the input frame. Expressed in range of [0~181.02].

SATLOW

Display the saturation value at the 10% percentile within the input frame. Expressed in range of [0~181.02].

SATAVG

Display the average saturation value within the input frame. Expressed in range of [0~181.02].

SATHIGH

Display the saturation value at the 90% percentile within the input frame. Expressed in range of [0~181.02].

SATMAX

Display the maximum saturation value contained within the input frame. Expressed in range of [0~181.02].

HUEMED

Display the median value for hue within the input frame. Expressed in range of [0-360].

HUEAVG

Display the average value for hue within the input frame. Expressed in range of [0-360].

YDIF

Display the average of sample value difference between all values of the Y plane in the current frame and corresponding values of the previous input frame. Expressed in range of [0-255].

UDIF

Display the average of sample value difference between all values of the U plane in the current frame and corresponding values of the previous input frame. Expressed in range of [0-255].

VDIF

Display the average of sample value difference between all values of the V plane in the current frame and corresponding values of the previous input frame. Expressed in range of [0-255].

The filter accepts the following options:

stat

out

stat specify an additional form of image analysis. out output video with the specified type of pixel highlighted.

Both options accept the following values:

‘tout’

Identify *temporal outliers* pixels. A *temporal outlier* is a pixel unlike the neighboring pixels of the same field. Examples of temporal outliers include the results of video dropouts, head clogs, or tape tracking issues.

‘vrep’

Identify *vertical line repetition*. Vertical line repetition includes similar rows of pixels within a frame. In born-digital video vertical line repetition is common, but this pattern is uncommon in video digitized from an analog source. When it occurs in video that results from the digitization of an analog source it can indicate concealment from a dropout compensator.

‘brng’

Identify pixels that fall outside of legal broadcast range.

color, c

Set the highlight color for the out option. The default color is yellow.

### 37.107.1 Examples# TOC

- Output data of various video metrics:

```
ffprobe -f lavfi movie=example.mov,signalstats="stat=tout+vrep+brng" -show_frames
```

- Output specific data about the minimum and maximum values of the Y plane per frame:

```
ffprobe -f lavfi movie=example.mov,signalstats -show_entries frame_tags=lavfi.signalstats.YMAX,lavfi.signalstats.YMIN
```

- Playback video while highlighting pixels that are outside of broadcast range in red.

```
ffplay example.mov -vf signalstats="out=brng:color=red"
```

- Playback video with signalstats metadata drawn over the frame.

```
ffplay example.mov -vf signalstats=stat=brng+vrep+tout,drawtext=fontfile=FreeSerif.ttf:textfile=signalstat_drawtext.txt
```

The contents of signalstat\_drawtext.txt used in the command are:

```
time %{pts:hms}  
Y (%{metadata:lavfi.signalstats.YMIN}-{metadata:lavfi.signalstats.YMAX})  
U (%{metadata:lavfi.signalstats.UMIN}-{metadata:lavfi.signalstats.UMAX})  
V (%{metadata:lavfi.signalstats.VMIN}-{metadata:lavfi.signalstats.VMAX})  
saturation maximum: %{metadata:lavfi.signalstats.SATMAX}
```

## 37.108 smartblur# TOC

Blur the input video without impacting the outlines.

It accepts the following options:

`luma_radius, lr`

Set the luma radius. The option value must be a float number in the range [0.1,5.0] that specifies the variance of the gaussian filter used to blur the image (slower if larger). Default value is 1.0.

`luma_strength, ls`

Set the luma strength. The option value must be a float number in the range [-1.0,1.0] that configures the blurring. A value included in [0.0,1.0] will blur the image whereas a value included in [-1.0,0.0] will sharpen the image. Default value is 1.0.

`luma_threshold, lt`

Set the luma threshold used as a coefficient to determine whether a pixel should be blurred or not. The option value must be an integer in the range [-30,30]. A value of 0 will filter all the image, a value included in [0,30] will filter flat areas and a value included in [-30,0] will filter edges. Default value is 0.

`chroma_radius, cr`

Set the chroma radius. The option value must be a float number in the range [0.1,5.0] that specifies the variance of the gaussian filter used to blur the image (slower if larger). Default value is 1.0.

`chroma_strength, cs`

Set the chroma strength. The option value must be a float number in the range [-1.0,1.0] that configures the blurring. A value included in [0.0,1.0] will blur the image whereas a value included in [-1.0,0.0] will sharpen the image. Default value is 1.0.

`chroma_threshold, ct`

Set the chroma threshold used as a coefficient to determine whether a pixel should be blurred or not. The option value must be an integer in the range [-30,30]. A value of 0 will filter all the image, a value included in [0,30] will filter flat areas and a value included in [-30,0] will filter edges. Default value is 0.

If a chroma option is not explicitly set, the corresponding luma value is set.

## 37.109 ssim# TOC

Obtain the SSIM (Structural SIMilarity Metric) between two input videos.

This filter takes in input two input videos, the first input is considered the "main" source and is passed unchanged to the output. The second input is used as a "reference" video for computing the SSIM.

Both video inputs must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The filter stores the calculated SSIM of each frame.

The description of the accepted parameters follows.

`stats_file, f`

If specified the filter will use the named file to save the SSIM of each individual frame.

The file printed if *stats\_file* is selected, contains a sequence of key/value pairs of the form *key:value* for each compared couple of frames.

A description of each shown parameter follows:

`n`

sequential number of the input frame, starting from 1

`Y, U, V, R, G, B`

SSIM of the compared frames for the component specified by the suffix.

`All`

SSIM of the compared frames for the whole frame.

`dB`

Same as above but in dB representation.

For example:

```
movie=ref_movie.mpg, setpts=PTS-STARTPTS [main];  
[main][ref] ssim="stats_file=stats.log" [out]
```

On this example the input file being processed is compared with the reference file `ref_movie.mpg`. The SSIM of each individual frame is stored in `stats.log`.

Another example with both psnr and ssim at same time:

```
ffmpeg -i main.mpg -i ref.mpg -lavfi "ssim:[0:v][1:v]psnr" -f null -
```

## 37.110 stereo3d# TOC

Convert between different stereoscopic image formats.

The filters accept the following options:

in

Set stereoscopic image format of input.

Available values for input image formats are:

‘sbsl’

side by side parallel (left eye left, right eye right)

‘sbsr’

side by side crosseye (right eye left, left eye right)

‘sbs2l’

side by side parallel with half width resolution (left eye left, right eye right)

‘sbs2r’

side by side crosseye with half width resolution (right eye left, left eye right)

‘abl’

above-below (left eye above, right eye below)

‘abr’

above-below (right eye above, left eye below)

‘ab2l’

above-below with half height resolution (left eye above, right eye below)

‘ab2r’

above-below with half height resolution (right eye above, left eye below)

‘al’

alternating frames (left eye first, right eye second)

‘ar’

alternating frames (right eye first, left eye second)

Default value is ‘sbsl’.

out

Set stereoscopic image format of output.

Available values for output image formats are all the input formats as well as:

‘arbg’

anaglyph red/blue gray (red filter on left eye, blue filter on right eye)

‘argg’

anaglyph red/green gray (red filter on left eye, green filter on right eye)

‘arcg’

anaglyph red/cyan gray (red filter on left eye, cyan filter on right eye)

‘arch’

anaglyph red/cyan half colored (red filter on left eye, cyan filter on right eye)

‘arcc’

anaglyph red/cyan color (red filter on left eye, cyan filter on right eye)

‘arcd’

anaglyph red/cyan color optimized with the least squares projection of dubois (red filter on left eye, cyan filter on right eye)

‘agmg’

anaglyph green/magenta gray (green filter on left eye, magenta filter on right eye)

‘agmh’

anaglyph green/magenta half colored (green filter on left eye, magenta filter on right eye)

‘agmc’

anaglyph green/magenta colored (green filter on left eye, magenta filter on right eye)

‘agmd’

anaglyph green/magenta color optimized with the least squares projection of dubois (green filter on left eye, magenta filter on right eye)

‘aybg’

anaglyph yellow/blue gray (yellow filter on left eye, blue filter on right eye)

‘aybh’

anaglyph yellow/blue half colored (yellow filter on left eye, blue filter on right eye)

‘aybc’

anaglyph yellow/blue colored (yellow filter on left eye, blue filter on right eye)

‘aybd’

anaglyph yellow/blue color optimized with the least squares projection of dubois (yellow filter on left eye, blue filter on right eye)

‘irl’

interleaved rows (left eye has top row, right eye starts on next row)

‘irr’

interleaved rows (right eye has top row, left eye starts on next row)

‘ml’

mono output (left eye only)

‘mr’

mono output (right eye only)

Default value is ‘arcd’.

### 37.110.1 Examples# TOC

- Convert input video from side by side parallel to anaglyph yellow/blue dubois:



```
stereo3d=sbsl:aybd
```

- Convert input video from above below (left eye above, right eye below) to side by side crosseye.

```
stereo3d=abl:sbsr
```

## 37.111 spp# TOC

Apply a simple postprocessing filter that compresses and decompresses the image at several (or - in the case of `quality` level 6 - all) shifts and average the results.

The filter accepts the following options:

`quality`

Set quality. This option defines the number of levels for averaging. It accepts an integer in the range 0-6. If set to 0, the filter will have no effect. A value of 6 means the higher quality. For each increment of that value the speed drops by a factor of approximately 2. Default value is 3.

`qp`

Force a constant quantization parameter. If not set, the filter will use the QP from the video stream (if available).

`mode`

Set thresholding mode. Available modes are:

`'hard'`

Set hard thresholding (default).

`'soft'`

Set soft thresholding (better de-ringing effect, but likely blurrier).

`use_bframe_qp`

Enable the use of the QP from the B-Frames if set to 1. Using this option may cause flicker since the B-Frames have often larger QP. Default is 0 (not enabled).

## 37.112 subtitles# TOC

Draw subtitles on top of input video using the libass library.

To enable compilation of this filter you need to configure FFmpeg with `--enable-libass`. This filter also requires a build with libavcodec and libavformat to convert the passed subtitles file to ASS (Advanced Substation Alpha) subtitles format.

The filter accepts the following options:

`filename, f`

Set the filename of the subtitle file to read. It must be specified.

`original_size`

Specify the size of the original video, the video for which the ASS file was composed. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Due to a misdesign in ASS aspect ratio arithmetic, this is necessary to correctly scale the fonts if the aspect ratio has been changed.

`fontsdir`

Set a directory path containing fonts that can be used by the filter. These fonts will be used in addition to whatever the font provider uses.

`charenc`

Set subtitles input character encoding. `subtitles` filter only. Only useful if not UTF-8.

`stream_index, si`

Set subtitles stream index. `subtitles` filter only.

`force_style`

Override default style or script info parameters of the subtitles. It accepts a string containing ASS style format `KEY=VALUE` couples separated by `",`.

If the first key is not specified, it is assumed that the first value specifies the `filename`.

For example, to render the file `sub.srt` on top of the input video, use the command:

```
subtitles=sub.srt
```

which is equivalent to:

```
subtitles=filename=sub.srt
```

To render the default subtitles stream from file `video.mkv`, use:

```
subtitles=video.mkv
```

To render the second subtitles stream from that file, use:

```
subtitles=video.mkv:si=1
```

To make the subtitles stream from `sub.srt` appear in transparent green DeJaVu Serif, use:

```
subtitles=sub.srt:force_style='FontName=DeJaVu Serif,PrimaryColour=&HAA00FF00'
```

### 37.113 super2xsai# TOC

Scale the input by 2x and smooth using the Super2xSaI (Scale and Interpolate) pixel art scaling algorithm.

Useful for enlarging pixel art images without reducing sharpness.

### 37.114 swapuv# TOC

Swap U & V plane.

### 37.115 telecine# TOC

Apply telecine process to the video.

This filter accepts the following options:

`first_field`

`'top, t'`

`top field first`

`'bottom, b'`

`bottom field first` The default value is `top`.

`pattern`

A string of numbers representing the pulldown pattern you wish to apply. The default value is 23.

Some typical patterns:

NTSC output (30i):

27.5p: 32222

24p: 23 (classic)

24p: 2332 (preferred)

20p: 33

18p: 334

16p: 3444

PAL output (25i):

27.5p: 12222

24p: 222222222223 ("Euro pulldown")

16.67p: 33

16p: 33333334

## 37.116 thumbnail# TOC

Select the most representative frame in a given sequence of consecutive frames.

The filter accepts the following options:

`n`

Set the frames batch size to analyze; in a set of  $n$  frames, the filter will pick one of them, and then handle the next batch of  $n$  frames until the end. Default is 100.

Since the filter keeps track of the whole frames sequence, a bigger  $n$  value will result in a higher memory usage, so a high value is not recommended.

### 37.116.1 Examples# TOC

- Extract one picture each 50 frames:

```
thumbnail=50
```

- Complete example of a thumbnail creation with `ffmpeg`:

```
ffmpeg -i in.avi -vf thumbnail,scale=300:200 -frames:v 1 out.png
```

## 37.117 tile# TOC

Tile several successive frames together.

The filter accepts the following options:

`layout`

Set the grid size (i.e. the number of lines and columns). For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

`nb_frames`

Set the maximum number of frames to render in the given area. It must be less than or equal to  $w \times h$ . The default value is 0, meaning all the area will be used.

`margin`

Set the outer border margin in pixels.

`padding`

Set the inner border thickness (i.e. the number of pixels between frames). For more advanced padding options (such as having different values for the edges), refer to the `pad` video filter.

color

Specify the color of the unused area. For the syntax of this option, check the "Color" section in the ffmpeg-utils manual. The default value of *color* is "black".

### 37.117.1 Examples# TOC

- Produce 8x8 PNG tiles of all keyframes (`-skip_frame nokey`) in a movie:

```
ffmpeg -skip_frame nokey -i file.avi -vf 'scale=128:72,tile=8x8' -an -vsync 0 keyframes%03d.png
```

The `-vsync 0` is necessary to prevent ffmpeg from duplicating each output frame to accommodate the originally detected frame rate.

- Display 5 pictures in an area of 3x2 frames, with 7 pixels between them, and 2 pixels of initial margin, using mixed flat and named options:

```
tile=3x2:nb_frames=5:padding=7:margin=2
```

### 37.118 tinterlace# TOC

Perform various types of temporal field interlacing.

Frames are counted starting from 1, so the first input frame is considered odd.

The filter accepts the following options:

mode

Specify the mode of the interlacing. This option can also be specified as a value alone. See below for a list of values for this option.

Available values are:

`'merge, 0'`

Move odd frames into the upper field, even into the lower field, generating a double height frame at half frame rate.

```
-----> time
Input:
Frame 1          Frame 2          Frame 3          Frame 4

11111            22222            33333            44444
11111            22222            33333            44444
11111            22222            33333            44444
11111            22222            33333            44444

Output:
11111            33333
22222            44444
11111            33333
```

22222	44444
11111	33333
22222	44444
11111	33333
22222	44444

‘drop\_odd, 1’

Only output even frames, odd frames are dropped, generating a frame with unchanged height at half frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444

Output:
                22222                44444
                22222                44444
                22222                44444
                22222                44444

```

‘drop\_even, 2’

Only output odd frames, even frames are dropped, generating a frame with unchanged height at half frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444

Output:
11111                33333
11111                33333
11111                33333
11111                33333

```

‘pad, 3’

Expand each frame to full height, but pad alternate lines with black, generating a frame with double height at the same input frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

```

11111	22222	33333	44444
11111	22222	33333	44444
11111	22222	33333	44444
11111	22222	33333	44444

Output:

11111	.....	33333	.....
.....	22222	.....	44444
11111	.....	33333	.....
.....	22222	.....	44444
11111	.....	33333	.....
.....	22222	.....	44444
11111	.....	33333	.....
.....	22222	.....	44444

‘interleave\_top, 4’

Interleave the upper field from odd frames with the lower field from even frames, generating a frame with unchanged height at half frame rate.

-----> time

Input:

Frame 1	Frame 2	Frame 3	Frame 4
11111<-	22222	33333<-	44444
11111	22222<-	33333	44444<-
11111<-	22222	33333<-	44444
11111	22222<-	33333	44444<-

Output:

11111	33333
22222	44444
11111	33333
22222	44444

‘interleave\_bottom, 5’

Interleave the lower field from odd frames with the upper field from even frames, generating a frame with unchanged height at half frame rate.

-----> time

Input:

Frame 1	Frame 2	Frame 3	Frame 4
11111	22222<-	33333	44444<-
11111<-	22222	33333<-	44444
11111	22222<-	33333	44444<-
11111<-	22222	33333<-	44444

Output:

22222	44444
11111	33333
22222	44444
11111	33333

`'interlacex2, 6'`

Double frame rate with unchanged height. Frames are inserted each containing the second temporal field from the previous input frame and the first temporal field from the next input frame. This mode relies on the `top_field_first` flag. Useful for interlaced video displays with no field synchronisation.

```
-----> time
Input:
Frame 1          Frame 2          Frame 3          Frame 4

11111           22222           33333           44444
 11111           22222           33333           44444
11111           22222           33333           44444
 11111           22222           33333           44444

Output:
11111  22222  22222  33333  33333  44444  44444
 11111  11111  22222  22222  33333  33333  44444
11111  22222  22222  33333  33333  44444  44444
 11111  11111  22222  22222  33333  33333  44444
```

Numeric values are deprecated but are accepted for backward compatibility reasons.

Default mode is merge.

flags

Specify flags influencing the filter process.

Available value for *flags* is:

`low_pass_filter, vlfp`

Enable vertical low-pass filtering in the filter. Vertical low-pass filtering is required when creating an interlaced destination from a progressive source which contains high-frequency vertical detail. Filtering will reduce interlace 'twitter' and Moire patterning.

Vertical low-pass filtering can only be enabled for mode *interleave\_top* and *interleave\_bottom*.

## 37.119 transpose# TOC

Transpose rows with columns in the input video and optionally flip it.

It accepts the following parameters:

`dir`

Specify the transposition direction.



Can assume the following values:

`'0, 4, cclock_flip'`

Rotate by 90 degrees counterclockwise and vertically flip (default), that is:

L.R		L.l
. .	->	. .
l.r		R.r

`'1, 5, clock'`

Rotate by 90 degrees clockwise, that is:

L.R		l.L
. .	->	. .
l.r		r.R

`'2, 6, cclock'`

Rotate by 90 degrees counterclockwise, that is:

L.R		R.r
. .	->	. .
l.r		L.l

`'3, 7, clock_flip'`

Rotate by 90 degrees clockwise and vertically flip, that is:

L.R		r.R
. .	->	. .
l.r		l.L

For values between 4-7, the transposition is only done if the input video geometry is portrait and not landscape. These values are deprecated, the `passthrough` option should be used instead.

Numerical values are deprecated, and should be dropped in favor of symbolic constants.

`passthrough`

Do not apply the transposition if the input geometry matches the one specified by the specified value. It accepts the following values:

`'none'`

Always apply transposition.

`'portrait'`

Preserve portrait geometry (when *height*  $\geq$  *width*).

‘landscape’

Preserve landscape geometry (when *width*  $\geq$  *height*).

Default value is none.

For example to rotate by 90 degrees clockwise and preserve portrait layout:

```
transpose=dir=1:passthrough=portrait
```

The command above can also be specified as:

```
transpose=1:portrait
```

## 37.120 trim# TOC

Trim the input so that the output contains one continuous subpart of the input.

It accepts the following parameters:

*start*

Specify the time of the start of the kept section, i.e. the frame with the timestamp *start* will be the first frame in the output.

*end*

Specify the time of the first frame that will be dropped, i.e. the frame immediately preceding the one with the timestamp *end* will be the last frame in the output.

*start\_pts*

This is the same as *start*, except this option sets the start timestamp in timebase units instead of seconds.

*end\_pts*

This is the same as *end*, except this option sets the end timestamp in timebase units instead of seconds.

*duration*

The maximum duration of the output in seconds.

*start\_frame*

The number of the first frame that should be passed to the output.

`end_frame`

The number of the first frame that should be dropped.

`start`, `end`, and `duration` are expressed as time duration specifications; see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax.

Note that the first two sets of the `start/end` options and the `duration` option look at the frame timestamp, while the `_frame` variants simply count the frames that pass through the filter. Also note that this filter does not modify the timestamps. If you wish for the output timestamps to start at zero, insert a `setpts` filter after the `trim` filter.

If multiple `start` or `end` options are set, this filter tries to be greedy and keep all the frames that match at least one of the specified constraints. To keep only the part that matches all the constraints at once, chain multiple `trim` filters.

The defaults are such that all the input is kept. So it is possible to set e.g. just the end values to keep everything before the specified time.

Examples:

- Drop everything except the second minute of input:

```
ffmpeg -i INPUT -vf trim=60:120
```

- Keep only the first second:

```
ffmpeg -i INPUT -vf trim=duration=1
```

## 37.121 unsharp# TOC

Sharpen or blur the input video.

It accepts the following parameters:

`luma_msize_x`, `lx`

Set the luma matrix horizontal size. It must be an odd integer between 3 and 63. The default value is 5.

`luma_msize_y`, `ly`

Set the luma matrix vertical size. It must be an odd integer between 3 and 63. The default value is 5.

`luma_amount`, `la`

Set the luma effect strength. It must be a floating point number, reasonable values lay between -1.5 and 1.5.

Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

Default value is 1.0.

`chroma_msize_x, cx`

Set the chroma matrix horizontal size. It must be an odd integer between 3 and 63. The default value is 5.

`chroma_msize_y, cy`

Set the chroma matrix vertical size. It must be an odd integer between 3 and 63. The default value is 5.

`chroma_amount, ca`

Set the chroma effect strength. It must be a floating point number, reasonable values lay between -1.5 and 1.5.

Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

Default value is 0.0.

`opencl`

If set to 1, specify using OpenCL capabilities, only available if FFmpeg was configured with `--enable-opencl`. Default value is 0.

All parameters are optional and default to the equivalent of the string '5:5:1.0:5:5:0.0'.

### 37.121.1 Examples# TOC

- Apply strong luma sharpen effect:

```
unsharp=luma_msize_x=7:luma_msize_y=7:luma_amount=2.5
```

- Apply a strong blur of both luma and chroma parameters:

```
unsharp=7:7:-2:7:7:-2
```

## 37.122 uspp# TOC

Apply ultra slow/simple postprocessing filter that compresses and decompresses the image at several (or - in the case of `quality` level 8 - all) shifts and average the results.

The way this differs from the behavior of `spp` is that `uspp` actually encodes & decodes each case with libavcodec Snow, whereas `spp` uses a simplified intra only 8x8 DCT similar to MJPEG.

The filter accepts the following options:

`quality`

Set quality. This option defines the number of levels for averaging. It accepts an integer in the range 0-8. If set to 0, the filter will have no effect. A value of 8 means the higher quality. For each increment of that value the speed drops by a factor of approximately 2. Default value is 3.

`qp`

Force a constant quantization parameter. If not set, the filter will use the QP from the video stream (if available).

## 37.123 vectorscope# TOC

Display 2 color component values in the two dimensional graph (which is called a vectorscope).

This filter accepts the following options:

`mode, m`

Set vectorscope mode.

It accepts the following values:

`'gray'`

Gray values are displayed on graph, higher brightness means more pixels have same component color value on location in graph. This is the default mode.

`'color'`

Gray values are displayed on graph. Surrounding pixels values which are not present in video frame are drawn in gradient of 2 color components which are set by option `x` and `y`.

`'color2'`

Actual color components values present in video frame are displayed on graph.

`'color3'`

Similar as color2 but higher frequency of same values x and y on graph increases value of another color component, which is luminance by default values of x and y.

`'color4'`

Actual colors present in video frame are displayed on graph. If two different colors map to same position on graph then color with higher value of component not present in graph is picked.

x

Set which color component will be represented on X-axis. Default is 1.

y

Set which color component will be represented on Y-axis. Default is 2.

intensity, i

Set intensity, used by modes: gray, color and color3 for increasing brightness of color component which represents frequency of (X, Y) location in graph.

envelope, e

`'none'`

No envelope, this is default.

`'instant'`

Instant envelope, even darkest single pixel will be clearly highlighted.

`'peak'`

Hold maximum and minimum values presented in graph over time. This way you can still spot out of range values without constantly looking at vectorscope.

`'peak+instant'`

Peak and instant envelope combined together.

## 37.124 vidstabdetect# TOC

Analyze video stabilization/deshaking. Perform pass 1 of 2, see vidstabtransform for pass 2.

This filter generates a file with relative translation and rotation transform information about subsequent frames, which is then used by the vidstabtransform filter.

To enable compilation of this filter you need to configure FFmpeg with `--enable-libvidstab`.

This filter accepts the following options:

`result`

Set the path to the file used to write the transforms information. Default value is `transforms.trf`.

`shakiness`

Set how shaky the video is and how quick the camera is. It accepts an integer in the range 1-10, a value of 1 means little shakiness, a value of 10 means strong shakiness. Default value is 5.

`accuracy`

Set the accuracy of the detection process. It must be a value in the range 1-15. A value of 1 means low accuracy, a value of 15 means high accuracy. Default value is 15.

`stepsize`

Set stepsize of the search process. The region around minimum is scanned with 1 pixel resolution. Default value is 6.

`mincontrast`

Set minimum contrast. Below this value a local measurement field is discarded. Must be a floating point value in the range 0-1. Default value is 0.3.

`tripod`

Set reference frame number for tripod mode.

If enabled, the motion of the frames is compared to a reference frame in the filtered stream, identified by the specified number. The idea is to compensate all movements in a more-or-less static scene and keep the camera view absolutely still.

If set to 0, it is disabled. The frames are counted starting from 1.

`show`

Show fields and transforms in the resulting frames. It accepts an integer in the range 0-2. Default value is 0, which disables any visualization.

### 37.124.1 Examples# TOC

- Use default values:

`vidstabdetect`

- Analyze strongly shaky movie and put the results in file `mytransforms.trf`:

```
vidstabdetect=shakiness=10:accuracy=15:result="mytransforms.trf"
```

- Visualize the result of internal transformations in the resulting video:

```
vidstabdetect=show=1
```

- Analyze a video with medium shakiness using `ffmpeg`:

```
ffmpeg -i input -vf vidstabdetect=shakiness=5:show=1 dummy.avi
```

## 37.125 vidstabtransform# TOC

Video stabilization/deshaking: pass 2 of 2, see `vidstabdetect` for pass 1.

Read a file with transform information for each frame and apply/compensate them. Together with the `vidstabdetect` filter this can be used to deshake videos. See also <http://public.hronopik.de/vid.stab>. It is important to also use the `unsharp` filter, see below.

To enable compilation of this filter you need to configure FFmpeg with `--enable-libvidstab`.

### 37.125.1 Options# TOC

`input`

Set path to the file used to read the transforms. Default value is `transforms.trf`.

`smoothing`

Set the number of frames ( $\text{value} * 2 + 1$ ) used for lowpass filtering the camera movements. Default value is 10.

For example a number of 10 means that 21 frames are used (10 in the past and 10 in the future) to smoothen the motion in the video. A larger value leads to a smoother video, but limits the acceleration of the camera (pan/tilt movements). 0 is a special case where a static camera is simulated.

`optalgo`

Set the camera path optimization algorithm.

Accepted values are:

`'gauss'`



gaussian kernel low-pass filter on camera motion (default)

‘avg’

averaging on transformations

maxshift

Set maximal number of pixels to translate frames. Default value is -1, meaning no limit.

maxangle

Set maximal angle in radians ( $\text{degree} \cdot \pi / 180$ ) to rotate frames. Default value is -1, meaning no limit.

crop

Specify how to deal with borders that may be visible due to movement compensation.

Available values are:

‘keep’

keep image information from previous frame (default)

‘black’

fill the border black

invert

Invert transforms if set to 1. Default value is 0.

relative

Consider transforms as relative to previous frame if set to 1, absolute if set to 0. Default value is 0.

zoom

Set percentage to zoom. A positive value will result in a zoom-in effect, a negative value in a zoom-out effect. Default value is 0 (no zoom).

optzoom

Set optimal zooming to avoid borders.

Accepted values are:

‘0’

disabled

‘1’

optimal static zoom value is determined (only very strong movements will lead to visible borders) (default)

‘2’

optimal adaptive zoom value is determined (no borders will be visible), see zoomspeed

Note that the value given at zoom is added to the one calculated here.

zoomspeed

Set percent to zoom maximally each frame (enabled when optzoom is set to 2). Range is from 0 to 5, default value is 0.25.

interpol

Specify type of interpolation.

Available values are:

‘no’

no interpolation

‘linear’

linear only horizontal

‘bilinear’

linear in both directions (default)

‘bicubic’

cubic in both directions (slow)

tripod

Enable virtual tripod mode if set to 1, which is equivalent to relative=0:smoothing=0. Default value is 0.

Use also tripod option of vidstabdetect.

debug

Increase log verbosity if set to 1. Also the detected global motions are written to the temporary file `global_motions.trf`. Default value is 0.

### 37.125.2 Examples# TOC

- Use `ffmpeg` for a typical stabilization with default values:

```
ffmpeg -i inp.mpeg -vf vidstabtransform,unsharp=5:5:0.8:3:3:0.4 inp_stabilized.mpeg
```

Note the use of the `unsharp` filter which is always recommended.

- Zoom in a bit more and load transform data from a given file:

```
vidstabtransform=zoom=5:input="mytransforms.trf"
```

- Smoothen the video even more:

```
vidstabtransform=smoothing=30
```

### 37.126 vflip# TOC

Flip the input video vertically.

For example, to vertically flip a video with `ffmpeg`:

```
ffmpeg -i in.avi -vf "vflip" out.avi
```

### 37.127 vignette# TOC

Make or reverse a natural vignetting effect.

The filter accepts the following options:

`angle`, `a`

Set lens angle expression as a number of radians.

The value is clipped in the  $[0, \pi/2]$  range.

Default value: `" $\pi/5$ "`

`x0`

`y0`

Set center coordinates expressions. Respectively `"w/2"` and `"h/2"` by default.

`mode`

Set forward/backward mode.

Available modes are:

‘forward’

The larger the distance from the central point, the darker the image becomes.

‘backward’

The larger the distance from the central point, the brighter the image becomes. This can be used to reverse a vignette effect, though there is no automatic detection to extract the lens angle and other settings (yet). It can also be used to create a burning effect.

Default value is ‘forward’.

eval

Set evaluation mode for the expressions (angle, x0, y0).

It accepts the following values:

‘init’

Evaluate expressions only once during the filter initialization.

‘frame’

Evaluate expressions for each incoming frame. This is way slower than the ‘init’ mode since it requires all the scalars to be re-computed, but it allows advanced dynamic expressions.

Default value is ‘init’.

dither

Set dithering to reduce the circular banding effects. Default is 1 (enabled).

aspect

Set vignette aspect. This setting allows one to adjust the shape of the vignette. Setting this value to the SAR of the input will make a rectangular vignetting following the dimensions of the video.

Default is 1/1.

### **37.127.1 Expressions# TOC**

The alpha, x0 and y0 expressions can contain the following parameters.

w  
h

input width and height

n

the number of input frame, starting from 0

pts

the PTS (Presentation TimeStamp) time of the filtered video frame, expressed in *TB* units, NAN if undefined

r

frame rate of the input video, NAN if the input frame rate is unknown

t

the PTS (Presentation TimeStamp) of the filtered video frame, expressed in seconds, NAN if undefined

tb

time base of the input video

### 37.127.2 Examples# TOC

- Apply simple strong vignetting effect:

```
vignette=PI/4
```

- Make a flickering vignetting:

```
vignette='PI/4+random(1)*PI/50':eval=frame
```

### 37.128 vstack# TOC

Stack input videos vertically.

All streams must be of same pixel format and of same width.

Note that this filter is faster than using overlay and pad filter to create same output.

The filter accept the following option:

nb\_inputs

Set number of input streams. Default is 2.

## 37.129 w3fdif# TOC

Deinterlace the input video ("w3fdif" stands for "Weston 3 Field Deinterlacing Filter").

Based on the process described by Martin Weston for BBC R&D, and implemented based on the de-interlace algorithm written by Jim Easterbrook for BBC R&D, the Weston 3 field deinterlacing filter uses filter coefficients calculated by BBC R&D.

There are two sets of filter coefficients, so called "simple": and "complex". Which set of filter coefficients is used can be set by passing an optional parameter:

`filter`

Set the interlacing filter coefficients. Accepts one of the following values:

`'simple'`

Simple filter coefficient set.

`'complex'`

More-complex filter coefficient set.

Default value is `'complex'`.

`deint`

Specify which frames to deinterlace. Accept one of the following values:

`'all'`

Deinterlace all frames,

`'interlaced'`

Only deinterlace frames marked as interlaced.

Default value is `'all'`.

## 37.130 waveform# TOC

Video waveform monitor.

The waveform monitor plots color component intensity. By default luminance only. Each column of the waveform corresponds to a column of pixels in the source video.

It accepts the following options:

`mode, m`

Can be either `row`, or `column`. Default is `column`. In row mode, the graph on the left side represents color component value 0 and the right side represents value = 255. In column mode, the top side represents color component value = 0 and bottom side represents value = 255.

`intensity, i`

Set intensity. Smaller values are useful to find out how many values of the same luminance are distributed across input rows/columns. Default value is 0.04. Allowed range is [0, 1].

`mirror, r`

Set mirroring mode. 0 means unmirrored, 1 means mirrored. In mirrored mode, higher values will be represented on the left side for `row` mode and at the top for `column` mode. Default is 1 (mirrored).

`display, d`

Set display mode. It accepts the following values:

`'overlay'`

Presents information identical to that in the `parade`, except that the graphs representing color components are superimposed directly over one another.

This display mode makes it easier to spot relative differences or similarities in overlapping areas of the color components that are supposed to be identical, such as neutral whites, grays, or blacks.

`'parade'`

Display separate graph for the color components side by side in `row` mode or one below the other in `column` mode.

Using this display mode makes it easy to spot color casts in the highlights and shadows of an image, by comparing the contours of the top and the bottom graphs of each waveform. Since whites, grays, and blacks are characterized by exactly equal amounts of red, green, and blue, neutral areas of the picture should display three waveforms of roughly equal width/height. If not, the correction is easy to perform by making level adjustments the three waveforms.

Default is `parade`.

`components, c`

Set which color components to display. Default is 1, which means only luminance or red color component if input is in RGB colorspace. If is set for example to 7 it will display all 3 (if) available color components.

envelope, e  
    'none'

No envelope, this is default.

    'instant'

Instant envelope, minimum and maximum values presented in graph will be easily visible even with small step value.

    'peak'

Hold minimum and maximum values presented in graph across time. This way you can still spot out of range values without constantly looking at waveforms.

    'peak+instant'

Peak and instant envelope combined together.

filter, f  
    'lowpass'

No filtering, this is default.

    'flat'

Luma and chroma combined together.

    'aflat'

Similar as above, but shows difference between blue and red chroma.

    'chroma'

Displays only chroma.

    'achroma'

Similar as above, but shows difference between blue and red chroma.

    'color'

Displays actual color value on waveform.

## 37.131 xbr# TOC

Apply the xBR high-quality magnification filter which is designed for pixel art. It follows a set of edge-detection rules, see <http://www.libretro.com/forums/viewtopic.php?f=6&t=134>.



It accepts the following option:

`n`

Set the scaling dimension: 2 for 2xBR, 3 for 3xBR and 4 for 4xBR. Default is 3.

### **37.132 yadif# TOC**

Deinterlace the input video ("yadif" means "yet another deinterlacing filter").

It accepts the following parameters:

`mode`

The interlacing mode to adopt. It accepts one of the following values:

`0, send_frame`

Output one frame for each frame.

`1, send_field`

Output one frame for each field.

`2, send_frame_nospatial`

Like `send_frame`, but it skips the spatial interlacing check.

`3, send_field_nospatial`

Like `send_field`, but it skips the spatial interlacing check.

The default value is `send_frame`.

`parity`

The picture field parity assumed for the input interlaced video. It accepts one of the following values:

`0, tff`

Assume the top field is first.

`1, bff`

Assume the bottom field is first.

`-1, auto`

Enable automatic detection of field parity.

The default value is `auto`. If the interlacing is unknown or the decoder does not export this information, top field first will be assumed.

`deint`

Specify which frames to deinterlace. Accept one of the following values:

`0, all`

Deinterlace all frames.

`1, interlaced`

Only deinterlace frames marked as interlaced.

The default value is `all`.

### 37.133 zoompan# TOC

Apply Zoom & Pan effect.

This filter accepts the following options:

`zoom, z`

Set the zoom expression. Default is 1.

`x`

`y`

Set the x and y expression. Default is 0.

`d`

Set the duration expression in number of frames. This sets for how many number of frames effect will last for single input image.

`s`

Set the output image size, default is 'hd720'.

Each expression can contain the following constants:

`in_w, iw`

Input width.

in\_h, ih

Input height.

out\_w, ow

Output width.

out\_h, oh

Output height.

in

Input frame count.

on

Output frame count.

x

y

Last calculated 'x' and 'y' position from 'x' and 'y' expression for current input frame.

px

py

'x' and 'y' of last output frame of previous input frame or 0 when there was not yet such frame (first input frame).

zoom

Last calculated zoom from 'z' expression for current input frame.

pzoom

Last calculated zoom of last output frame of previous input frame.

duration

Number of output frames for current input frame. Calculated from 'd' expression for each input frame.

pduration

number of output frames created for previous input frame

a

Rational number: input width / input height

sar

sample aspect ratio

dar

display aspect ratio

### 37.133.1 Examples# TOC

- Zoom-in up to 1.5 and pan at same time to some spot near center of picture:

```
zoompan=z='min(zoom+0.0015,1.5)':d=700:x='if(gte(zoom,1.5),x,x+1/a)':y='if(gte(zoom,1.5),y,y+1)':s=640x360
```

- Zoom-in up to 1.5 and pan always at center of picture:

```
zoompan=z='min(zoom+0.0015,1.5)':d=700:x='iw/2-(iw/zoom/2)':y='ih/2-(ih/zoom/2)'
```

## 38 Video Sources# TOC

Below is a description of the currently available video sources.

### 38.1 buffer# TOC

Buffer video frames, and make them available to the filter chain.

This source is mainly intended for a programmatic use, in particular through the interface defined in `libavfilter/vsrc_buffer.h`.

It accepts the following parameters:

video\_size

Specify the size (width and height) of the buffered video frames. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

width

The input video width.

height

The input video height.

`pix_fmt`

A string representing the pixel format of the buffered video frames. It may be a number corresponding to a pixel format, or a pixel format name.

`time_base`

Specify the timebase assumed by the timestamps of the buffered frames.

`frame_rate`

Specify the frame rate expected for the video stream.

`pixel_aspect, sar`

The sample (pixel) aspect ratio of the input video.

`sws_param`

Specify the optional parameters to be used for the scale filter which is automatically inserted when an input change is detected in the input size or format.

For example:

```
buffer=width=320:height=240:pix_fmt=yuv410p:time_base=1/24:sar=1
```

will instruct the source to accept video frames with size 320x240 and with format "yuv410p", assuming 1/24 as the timestamps timebase and square pixels (1:1 sample aspect ratio). Since the pixel format with name "yuv410p" corresponds to the number 6 (check the enum `AVPixelFormat` definition in `libavutil/pixfmt.h`), this example corresponds to:

```
buffer=size=320x240:pixfmt=6:time_base=1/24:pixel_aspect=1/1
```

Alternatively, the options can be specified as a flat string, but this syntax is deprecated:

```
width:height:pix_fmt:time_base.num:time_base.den:pixel_aspect.num:pixel_aspect.den[:sws_param]
```

## 38.2 cellauto# TOC

Create a pattern generated by an elementary cellular automaton.

The initial state of the cellular automaton can be defined through the `filename`, and `pattern` options. If such options are not specified an initial state is created randomly.

At each new frame a new row in the video is filled with the result of the cellular automaton next generation. The behavior when the whole frame is filled is defined by the `scroll` option.

This source accepts the following options:

`filename, f`

Read the initial cellular automaton state, i.e. the starting row, from the specified file. In the file, each non-whitespace character is considered an alive cell, a newline will terminate the row, and further characters in the file will be ignored.

`pattern, p`

Read the initial cellular automaton state, i.e. the starting row, from the specified string.

Each non-whitespace character in the string is considered an alive cell, a newline will terminate the row, and further characters in the string will be ignored.

`rate, r`

Set the video rate, that is the number of frames generated per second. Default is 25.

`random_fill_ratio, ratio`

Set the random fill ratio for the initial cellular automaton row. It is a floating point number value ranging from 0 to 1, defaults to 1/PHI.

This option is ignored when a file or a pattern is specified.

`random_seed, seed`

Set the seed for filling randomly the initial row, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to -1, the filter will try to use a good random seed on a best effort basis.

`rule`

Set the cellular automaton rule, it is a number ranging from 0 to 255. Default value is 110.

`size, s`

Set the size of the output video. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

If `filename` or `pattern` is specified, the size is set by default to the width of the specified initial state row, and the height is set to *width* \* PHI.

If `size` is set, it must contain the width of the specified pattern string, and the specified pattern will be centered in the larger row.

If a filename or a pattern string is not specified, the size value defaults to "320x518" (used for a randomly generated initial state).

`scroll`

If set to 1, scroll the output upward when all the rows in the output have been already filled. If set to 0, the new generated row will be written over the top row just after the bottom row is filled. Defaults to 1.

`start_full, full`

If set to 1, completely fill the output with generated rows before outputting the first frame. This is the default behavior, for disabling set the value to 0.

`stitch`

If set to 1, stitch the left and right row edges together. This is the default behavior, for disabling set the value to 0.

### 38.2.1 Examples# TOC

- Read the initial state from `pattern`, and specify an output of size 200x400.

```
cellauto=f=pattern:s=200x400
```

- Generate a random initial row with a width of 200 cells, with a fill ratio of 2/3:

```
cellauto=ratio=2/3:s=200x200
```

- Create a pattern generated by rule 18 starting by a single alive cell centered on an initial row with width 100:

```
cellauto=p=@:s=100x400:full=0:rule=18
```

- Specify a more elaborated initial pattern:

```
cellauto=p='@@ @ @@':s=100x400:full=0:rule=18
```

### 38.3 mandelbrot# TOC

Generate a Mandelbrot set fractal, and progressively zoom towards the point specified with *start\_x* and *start\_y*.

This source accepts the following options:

`end_pts`

Set the terminal pts value. Default value is 400.

`end_scale`

Set the terminal scale value. Must be a floating point value. Default value is 0.3.

`inner`

Set the inner coloring mode, that is the algorithm used to draw the Mandelbrot fractal internal region.

It shall assume one of the following values:

`black`

Set black mode.

`convergence`

Show time until convergence.

`mincol`

Set color based on point closest to the origin of the iterations.

`period`

Set period mode.

Default value is *mincol*.

`bailout`

Set the bailout value. Default value is 10.0.

`maxiter`

Set the maximum of iterations performed by the rendering algorithm. Default value is 7189.

`outer`

Set outer coloring mode. It shall assume one of following values:

`iteration_count`

Set iteration count mode.

`normalized_iteration_count`

set normalized iteration count mode.



Default value is *normalized\_iteration\_count*.

`rate, r`

Set frame rate, expressed as number of frames per second. Default value is "25".

`size, s`

Set frame size. For the syntax of this option, check the "Video size" section in the ffmpeg-utils manual. Default value is "640x480".

`start_scale`

Set the initial scale value. Default value is 3.0.

`start_x`

Set the initial x position. Must be a floating point value between -100 and 100. Default value is -0.743643887037158704752191506114774.

`start_y`

Set the initial y position. Must be a floating point value between -100 and 100. Default value is -0.131825904205311970493132056385139.

## 38.4 mptestsrc# TOC

Generate various test patterns, as generated by the MPlayer test filter.

The size of the generated video is fixed, and is 256x256. This source is useful in particular for testing encoding features.

This source accepts the following options:

`rate, r`

Specify the frame rate of the sourced video, as the number of frames generated per second. It has to be a string in the format *frame\_rate\_num/frame\_rate\_den*, an integer number, a floating point number or a valid video frame rate abbreviation. The default value is "25".

`duration, d`

Set the duration of the sourced video. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax.

If not specified, or the expressed duration is negative, the video is supposed to be generated forever.

`test, t`

Set the number or the name of the test to perform. Supported tests are:

dc\_luma  
dc\_chroma  
freq\_luma  
freq\_chroma  
amp\_luma  
amp\_chroma  
cbp  
mv  
ring1  
ring2  
all

Default value is "all", which will cycle through the list of all tests.

Some examples:

```
mptestsrc=t=dc_luma
```

will generate a "dc\_luma" test pattern.

## 38.5 frei0r\_src# TOC

Provide a frei0r source.

To enable compilation of this filter you need to install the frei0r header and configure FFmpeg with `--enable-frei0r`.

This source accepts the following parameters:

`size`

The size of the video to generate. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

`framerate`

The framerate of the generated video. It may be a string of the form *num/den* or a frame rate abbreviation.

`filter_name`

The name to the frei0r source to load. For more information regarding frei0r and how to set the parameters, read the frei0r section in the video filters documentation.

`filter_params`

A `'`-separated list of parameters to pass to the `frei0r` source.

For example, to generate a `frei0r` `partik01` source with size 200x200 and frame rate 10 which is overlaid on the overlay filter main input:

```
frei0r_src=size=200x200:framerate=10:filter_name=partik01:filter_params=1234 [overlay]; [in][overlay] overlay
```

## 38.6 life# TOC

Generate a life pattern.

This source is based on a generalization of John Conway's life game.

The sourced input represents a life grid, each pixel represents a cell which can be in one of two possible states, alive or dead. Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent.

At each interaction the grid evolves according to the adopted rule, which specifies the number of neighbor alive cells which will make a cell stay alive or born. The `rule` option allows one to specify the rule to adopt.

This source accepts the following options:

`filename, f`

Set the file from which to read the initial grid state. In the file, each non-whitespace character is considered an alive cell, and newline is used to delimit the end of each row.

If this option is not specified, the initial grid is generated randomly.

`rate, r`

Set the video rate, that is the number of frames generated per second. Default is 25.

`random_fill_ratio, ratio`

Set the random fill ratio for the initial random grid. It is a floating point number value ranging from 0 to 1, defaults to 1/PHI. It is ignored when a file is specified.

`random_seed, seed`

Set the seed for filling the initial random grid, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to -1, the filter will try to use a good random seed on a best effort basis.

`rule`

Set the life rule.

A rule can be specified with a code of the kind "SNS/BNB", where *NS* and *NB* are sequences of numbers in the range 0-8, *NS* specifies the number of alive neighbor cells which make a live cell stay alive, and *NB* the number of alive neighbor cells which make a dead cell to become alive (i.e. to "born"). "s" and "b" can be used in place of "S" and "B", respectively.

Alternatively a rule can be specified by an 18-bits integer. The 9 high order bits are used to encode the next cell state if it is alive for each number of neighbor alive cells, the low order bits specify the rule for "borning" new cells. Higher order bits encode for an higher number of neighbor cells. For example the number 6153 = ( 12<<9 ) +9 specifies a stay alive rule of 12 and a born rule of 9, which corresponds to "S23/B03".

Default value is "S23/B3", which is the original Conway's game of life rule, and will keep a cell alive if it has 2 or 3 neighbor alive cells, and will born a new cell if there are three alive cells around a dead cell.

size, s

Set the size of the output video. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

If filename is specified, the size is set by default to the same size of the input file. If size is set, it must contain the size specified in the input file, and the initial grid defined in that file is centered in the larger resulting area.

If a filename is not specified, the size value defaults to "320x240" (used for a randomly generated initial grid).

stitch

If set to 1, stitch the left and right grid edges together, and the top and bottom edges also. Defaults to 1.

mold

Set cell mold speed. If set, a dead cell will go from death\_color to mold\_color with a step of mold. mold can have a value from 0 to 255.

life\_color

Set the color of living (or new born) cells.

death\_color

Set the color of dead cells. If mold is set, this is the first color used to represent a dead cell.

mold\_color

Set mold color, for definitely dead and moldy cells.

For the syntax of these 3 color options, check the "Color" section in the ffmpeg-utils manual.

### 38.6.1 Examples# TOC

- Read a grid from `pattern`, and center it on a grid of size 300x300 pixels:

```
life=f=pattern:s=300x300
```

- Generate a random grid of size 200x200, with a fill ratio of 2/3:

```
life=ratio=2/3:s=200x200
```

- Specify a custom rule for evolving a randomly generated grid:

```
life=rule=S14/B34
```

- Full example with slow death effect (mold) using `ffplay`:

```
ffplay -f lavfi life=s=300x200:mold=10:r=60:ratio=0.1:death_color=#C83232:life_color=#00ff00,scale=1200:800:flags=16
```

## 38.7 allrgb, allyuv, color, haldclutsrc, nullsrc, rgbtestsrc, smptebars, smptehtbars, testsrc# TOC

The `allrgb` source returns frames of size 4096x4096 of all rgb colors.

The `allyuv` source returns frames of size 4096x4096 of all yuv colors.

The `color` source provides an uniformly colored input.

The `haldclutsrc` source provides an identity Hald CLUT. See also `haldclut` filter.

The `nullsrc` source returns unprocessed video frames. It is mainly useful to be employed in analysis / debugging tools, or as the source for filters which ignore the input data.

The `rgbtestsrc` source generates an RGB test pattern useful for detecting RGB vs BGR issues. You should see a red, green and blue stripe from top to bottom.

The `smptebars` source generates a color bars pattern, based on the SMPTE Engineering Guideline EG 1-1990.

The `smptehtbars` source generates a color bars pattern, based on the SMPTE RP 219-2002.

The `testsrc` source generates a test video pattern, showing a color pattern, a scrolling gradient and a timestamp. This is mainly intended for testing purposes.

The sources accept the following parameters:

`color, c`

Specify the color of the source, only available in the `color` source. For the syntax of this option, check the "Color" section in the `ffmpeg-utils` manual.

`level`

Specify the level of the Hald CLUT, only available in the `haldclutsrc` source. A level of  $N$  generates a picture of  $N*N*N$  by  $N*N*N$  pixels to be used as identity matrix for 3D lookup tables. Each component is coded on a  $1 / (N*N)$  scale.

`size, s`

Specify the size of the sourced video. For the syntax of this option, check the (`ffmpeg-utils`)"Video size" section in the `ffmpeg-utils` manual. The default value is 320x240.

This option is not available with the `haldclutsrc` filter.

`rate, r`

Specify the frame rate of the sourced video, as the number of frames generated per second. It has to be a string in the format *frame\_rate\_num/frame\_rate\_den*, an integer number, a floating point number or a valid video frame rate abbreviation. The default value is "25".

`sar`

Set the sample aspect ratio of the sourced video.

`duration, d`

Set the duration of the sourced video. See (`ffmpeg-utils`)the Time duration section in the `ffmpeg-utils(1)` manual for the accepted syntax.

If not specified, or the expressed duration is negative, the video is supposed to be generated forever.

`decimals, n`

Set the number of decimals to show in the timestamp, only available in the `testsrc` source.

The displayed timestamp value will correspond to the original timestamp value multiplied by the power of 10 of the specified value. Default value is 0.

For example the following:

```
testsrc=duration=5.3:size=qcif:rate=10
```

will generate a video with a duration of 5.3 seconds, with size 176x144 and a frame rate of 10 frames per second.

The following graph description will generate a red source with an opacity of 0.2, with size "qcif" and a frame rate of 10 frames per second.

```
color=c=red@0.2:s=qcif:r=10
```

If the input content is to be ignored, `nullsrc` can be used. The following command generates noise in the luminance plane by employing the `geq` filter:

```
nullsrc=s=256x256, geq=random(1)*255:128:128
```

### **38.7.1 Commands# TOC**

The `color` source supports the following commands:

`c, color`

Set the color of the created image. Accepts the same syntax of the corresponding `color` option.

## **39 Video Sinks# TOC**

Below is a description of the currently available video sinks.

### **39.1 buffersink# TOC**

Buffer video frames, and make them available to the end of the filter graph.

This sink is mainly intended for programmatic use, in particular through the interface defined in `libavfilter/buffersink.h` or the options system.

It accepts a pointer to an `AVBufferSinkContext` structure, which defines the incoming buffers' formats, to be passed as the opaque parameter to `avfilter_init_filter` for initialization.

### **39.2 nullsink# TOC**

Null video sink: do absolutely nothing with the input video. It is mainly useful as a template and for use in analysis / debugging tools.

## **40 Multimedia Filters# TOC**

Below is a description of the currently available multimedia filters.

### **40.1 aphasemeter# TOC**

Convert input audio to a video output, displaying the audio phase.

The filter accepts the following options:

`rate, r`

Set the output frame rate. Default value is 25.

`size, s`

Set the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 800×400.

`rc`

`gc`

`bc`

Specify the red, green, blue contrast. Default values are 2, 7 and 1. Allowed range is [ 0 , 255 ].

`mpc`

Set color which will be used for drawing median phase. If color is none which is default, no median phase value will be drawn.

The filter also exports the frame metadata `lavfi.aphasemeter.phase` which represents mean phase of current audio frame. Value is in range [ -1 , 1 ]. The -1 means left and right channels are completely out of phase and 1 means channels are in phase.

## 40.2 avectorscope# TOC

Convert input audio to a video output, representing the audio vector scope.

The filter is used to measure the difference between channels of stereo audio stream. A monoaural signal, consisting of identical left and right signal, results in straight vertical line. Any stereo separation is visible as a deviation from this line, creating a Lissajous figure. If the straight (or deviation from it) but horizontal line appears this indicates that the left and right channels are out of phase.

The filter accepts the following options:

`mode, m`

Set the vectorscope mode.

Available values are:

`'lissajous'`

Lissajous rotated by 45 degrees.



`'lissajous_xy'`

Same as above but not rotated.

`'polar'`

Shape resembling half of circle.

Default value is `'lissajous'`.

`size, s`

Set the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 400×400.

`rate, r`

Set the output frame rate. Default value is 25.

`rc`

`gc`

`bc`

`ac`

Specify the red, green, blue and alpha contrast. Default values are 40, 160, 80 and 255. Allowed range is [0, 255].

`rf`

`gf`

`bf`

`af`

Specify the red, green, blue and alpha fade. Default values are 15, 10, 5 and 5. Allowed range is [0, 255].

`zoom`

Set the zoom factor. Default value is 1. Allowed range is [1, 10].

### 40.2.1 Examples# TOC

- Complete example using `ffplay`:

```
ffplay -f lavfi 'amovie=input.mp3, asplit [a][out1];  
[a] avectorscope=zoom=1.3:rc=2:gc=200:bc=10:rf=1:gf=8:bf=7 [out0]'
```

## 40.3 concat# TOC

Concatenate audio and video streams, joining them together one after the other.

The filter works on segments of synchronized video and audio streams. All segments must have the same number of streams of each type, and that will also be the number of streams at output.

The filter accepts the following options:

`n`

Set the number of segments. Default is 2.

`v`

Set the number of output video streams, that is also the number of video streams in each segment. Default is 1.

`a`

Set the number of output audio streams, that is also the number of audio streams in each segment. Default is 0.

`unsafe`

Activate unsafe mode: do not fail if segments have a different format.

The filter has  $v+a$  outputs: first  $v$  video outputs, then  $a$  audio outputs.

There are  $nx(v+a)$  inputs: first the inputs for the first segment, in the same order as the outputs, then the inputs for the second segment, etc.

Related streams do not always have exactly the same duration, for various reasons including codec frame size or sloppy authoring. For that reason, related synchronized streams (e.g. a video and its audio track) should be concatenated at once. The concat filter will use the duration of the longest stream in each segment (except the last one), and if necessary pad shorter audio streams with silence.

For this filter to work correctly, all segments must start at timestamp 0.

All corresponding streams must have the same parameters in all segments; the filtering system will automatically select a common pixel format for video streams, and a common sample format, sample rate and channel layout for audio streams, but other settings, such as resolution, must be converted explicitly by the user.

Different frame rates are acceptable but will result in variable frame rate at output; be sure to configure the output file to handle it.

### 40.3.1 Examples# TOC

- Concatenate an opening, an episode and an ending, all in bilingual version (video in stream 0, audio in streams 1 and 2):

```
ffmpeg -i opening.mkv -i episode.mkv -i ending.mkv -filter_complex \
'[0:0] [0:1] [0:2] [1:0] [1:1] [1:2] [2:0] [2:1] [2:2]
concat=n=3:v=1:a=2 [v] [a1] [a2]' \
-map '[v]' -map '[a1]' -map '[a2]' output.mkv
```

- Concatenate two parts, handling audio and video separately, using the (a)movie sources, and adjusting the resolution:

```
movie=part1.mp4, scale=512:288 [v1] ; amovie=part1.mp4 [a1] ;
movie=part2.mp4, scale=512:288 [v2] ; amovie=part2.mp4 [a2] ;
[v1] [v2] concat [outv] ; [a1] [a2] concat=v=0:a=1 [outa]
```

Note that a desync will happen at the stitch if the audio and video streams do not have exactly the same duration in the first file.

## 40.4 ebur128# TOC

EBU R128 scanner filter. This filter takes an audio stream as input and outputs it unchanged. By default, it logs a message at a frequency of 10Hz with the Momentary loudness (identified by M), Short-term loudness (S), Integrated loudness (I) and Loudness Range (LRA).

The filter also has a video output (see the *video* option) with a real time graph to observe the loudness evolution. The graphic contains the logged message mentioned above, so it is not printed anymore when this option is set, unless the verbose logging is set. The main graphing area contains the short-term loudness (3 seconds of analysis), and the gauge on the right is for the momentary loudness (400 milliseconds).

More information about the Loudness Recommendation EBU R128 on <http://tech.ebu.ch/loudness>.

The filter accepts the following options:

**video**

Activate the video output. The audio stream is passed unchanged whether this option is set or no. The video stream will be the first output stream if activated. Default is 0.

**size**

Set the video size. This option is for video only. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default and minimum resolution is 640x480.

**meter**

Set the EBU scale meter. Default is 9. Common values are 9 and 18, respectively for EBU scale meter +9 and EBU scale meter +18. Any other integer value between this range is allowed.

## metadata

Set metadata injection. If set to 1, the audio input will be segmented into 100ms output frames, each of them containing various loudness information in metadata. All the metadata keys are prefixed with `lavfi.r128..`

Default is 0.

## framelog

Force the frame logging level.

Available values are:

‘info’

information logging level

‘verbose’

verbose logging level

By default, the logging level is set to *info*. If the `video` or the `metadata` options are set, it switches to *verbose*.

## peak

Set peak mode(s).

Available modes can be cumulated (the option is a `flag` type). Possible values are:

‘none’

Disable any peak mode (default).

‘sample’

Enable sample-peak mode.

Simple peak mode looking for the higher sample value. It logs a message for sample-peak (identified by SPK).

‘true’

Enable true-peak mode.

If enabled, the peak lookup is done on an over-sampled version of the input stream for better peak accuracy. It logs a message for true-peak. (identified by TPK) and true-peak per frame (identified by FTPK). This mode requires a build with `libswresample`.

#### 40.4.1 Examples# TOC

- Real-time graph using `ffplay`, with a EBU scale meter +18:

```
ffplay -f lavfi -i "amovie=input.mp3,eburl28=video=1:meter=18 [out0][out1]"
```

- Run an analysis with `ffmpeg`:

```
ffmpeg -nostats -i input.mp3 -filter_complex eburl28 -f null -
```

### 40.5 interleave, ainterleave# TOC

Temporally interleave frames from several inputs.

`interleave` works with video inputs, `ainterleave` with audio.

These filters read frames from several inputs and send the oldest queued frame to the output.

Input streams must have a well defined, monotonically increasing frame timestamp values.

In order to submit one frame to output, these filters need to enqueue at least one frame for each input, so they cannot work in case one input is not yet terminated and will not receive incoming frames.

For example consider the case when one input is a `select` filter which always drop input frames. The `interleave` filter will keep reading from that input, but it will never be able to send new frames to output until the input will send an end-of-stream signal.

Also, depending on inputs synchronization, the filters will drop frames in case one input receives more frames than the other ones, and the queue is already filled.

These filters accept the following options:

`nb_inputs, n`

Set the number of different inputs, it is 2 by default.

#### 40.5.1 Examples# TOC

- Interleave frames belonging to different streams using `ffmpeg`:

```
ffmpeg -i bambi.avi -i pr0n.mkv -filter_complex "[0:v][1:v] interleave" out.avi
```

- Add flickering blur effect:

```
select='if(gt(random(0), 0.2), 1, 2)':n=2 [tmp], boxblur=2:2, [tmp] interleave
```

## 40.6 perms, aperms# TOC

Set read/write permissions for the output frames.

These filters are mainly aimed at developers to test direct path in the following filter in the filtergraph.

The filters accept the following options:

mode

Select the permissions mode.

It accepts the following values:

‘none’

Do nothing. This is the default.

‘ro’

Set all the output frames read-only.

‘rw’

Set all the output frames directly writable.

‘toggle’

Make the frame read-only if writable, and writable if read-only.

‘random’

Set each output frame read-only or writable randomly.

seed

Set the seed for the *random* mode, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to `-1`, the filter will try to use a good random seed on a best effort basis.

Note: in case of auto-inserted filter between the permission filter and the following one, the permission might not be received as expected in that following filter. Inserting a `format` or `aformat` filter before the `perms/aperms` filter can avoid this problem.

## 40.7 select, aselect# TOC

Select frames to pass in output.

This filter accepts the following options:

`expr, e`

Set expression, which is evaluated for each input frame.

If the expression is evaluated to zero, the frame is discarded.

If the evaluation result is negative or NaN, the frame is sent to the first output; otherwise it is sent to the output with index `ceil(val)-1`, assuming that the input index starts from 0.

For example a value of `1.2` corresponds to the output with index `ceil(1.2)-1 = 2-1 = 1`, that is the second output.

`outputs, n`

Set the number of outputs. The output to which to send the selected frame is based on the result of the evaluation. Default value is 1.

The expression can contain the following constants:

`n`

The (sequential) number of the filtered frame, starting from 0.

`selected_n`

The (sequential) number of the selected frame, starting from 0.

`prev_selected_n`

The sequential number of the last selected frame. It's NAN if undefined.

`TB`

The timebase of the input timestamps.

`pts`

The PTS (Presentation TimeStamp) of the filtered video frame, expressed in *TB* units. It's NAN if undefined.

`t`

The PTS of the filtered video frame, expressed in seconds. It's NAN if undefined.

`prev_pts`

The PTS of the previously filtered video frame. It's NAN if undefined.

`prev_selected_pts`

The PTS of the last previously filtered video frame. It's NAN if undefined.

`prev_selected_t`

The PTS of the last previously selected video frame. It's NAN if undefined.

`start_pts`

The PTS of the first video frame in the video. It's NAN if undefined.

`start_t`

The time of the first video frame in the video. It's NAN if undefined.

`pict_type (video only)`

The type of the filtered frame. It can assume one of the following values:

I

P

B

S

SI

SP

BI

`interlace_type (video only)`

The frame interlace type. It can assume one of the following values:

PROGRESSIVE

The frame is progressive (not interlaced).

TOPFIRST

The frame is top-field-first.

BOTTOMFIRST

The frame is bottom-field-first.



`consumed_sample_n (audio only)`

the number of selected samples before the current frame

`samples_n (audio only)`

the number of samples in the current frame

`sample_rate (audio only)`

the input sample rate

`key`

This is 1 if the filtered frame is a key-frame, 0 otherwise.

`pos`

the position in the file of the filtered frame, -1 if the information is not available (e.g. for synthetic video)

`scene (video only)`

value between 0 and 1 to indicate a new scene; a low value reflects a low probability for the current frame to introduce a new scene, while a higher value means the current frame is more likely to be one (see the example below)

The default value of the select expression is "1".

### 40.7.1 Examples# TOC

- Select all frames in input:

```
select
```

The example above is the same as:

```
select=1
```

- Skip all frames:

```
select=0
```

- Select only I-frames:

```
select='eq(pict_type\,I)'
```

- Select one frame every 100:

```
select='not(mod(n\,100))'
```

- Select only frames contained in the 10-20 time interval:

```
select=between(t\,10\,20)
```

- Select only I frames contained in the 10-20 time interval:

```
select=between(t\,10\,20)*eq(pict_type\,I)
```

- Select frames with a minimum distance of 10 seconds:

```
select='isnan(prev_selected_t)+gte(t-prev_selected_t\,10)'
```

- Use aselect to select only audio frames with samples number > 100:

```
aselect='gt(samples_n\,100)'
```

- Create a mosaic of the first scenes:

```
ffmpeg -i video.avi -vf select='gt(scene\,0.4)',scale=160:120,tile -frames:v 1 preview.png
```

Comparing *scene* against a value between 0.3 and 0.5 is generally a sane choice.

- Send even and odd frames to separate outputs, and compose them:

```
select=n=2:e='mod(n, 2)+1' [odd][even]; [odd] pad=h=2*ih [tmp]; [tmp][even] overlay=y=h
```

## 40.8 sendcmd, asendcmd# TOC

Send commands to filters in the filtergraph.

These filters read commands to be sent to other filters in the filtergraph.

sendcmd must be inserted between two video filters, asendcmd must be inserted between two audio filters, but apart from that they act the same way.

The specification of commands can be provided in the filter arguments with the *commands* option, or in a file specified by the *filename* option.

These filters accept the following options:

**commands, c**

Set the commands to be read and sent to the other filters.

**filename, f**

Set the filename of the commands to be read and sent to the other filters.

## 40.8.1 Commands syntax# TOC

A commands description consists of a sequence of interval specifications, comprising a list of commands to be executed when a particular event related to that interval occurs. The occurring event is typically the current frame time entering or leaving a given time interval.

An interval is specified by the following syntax:

```
START [ -END ] COMMANDS ;
```

The time interval is specified by the *START* and *END* times. *END* is optional and defaults to the maximum time.

The current frame time is considered within the specified interval if it is included in the interval [*START*, *END*), that is when the time is greater or equal to *START* and is lesser than *END*.

*COMMANDS* consists of a sequence of one or more command specifications, separated by ",", relating to that interval. The syntax of a command specification is given by:

```
[FLAGS] TARGET COMMAND ARG
```

*FLAGS* is optional and specifies the type of events relating to the time interval which enable sending the specified command, and must be a non-null sequence of identifier flags separated by "+" or "|" and enclosed between "[" and "]".

The following flags are recognized:

`enter`

The command is sent when the current frame timestamp enters the specified interval. In other words, the command is sent when the previous frame timestamp was not in the given interval, and the current is.

`leave`

The command is sent when the current frame timestamp leaves the specified interval. In other words, the command is sent when the previous frame timestamp was in the given interval, and the current is not.

If *FLAGS* is not specified, a default value of [`enter`] is assumed.

*TARGET* specifies the target of the command, usually the name of the filter class or a specific filter instance name.

*COMMAND* specifies the name of the command for the target filter.

*ARG* is optional and specifies the optional list of argument for the given *COMMAND*.

Between one interval specification and another, whitespaces, or sequences of characters starting with # until the end of line, are ignored and can be used to annotate comments.

A simplified BNF description of the commands specification syntax follows:

```
COMMAND_FLAG  ::= "enter" | "leave"
COMMAND_FLAGS ::= COMMAND_FLAG [(+|"")COMMAND_FLAG]
COMMAND       ::= [" " COMMAND_FLAGS ""] TARGET COMMAND [ARG]
COMMANDS      ::= COMMAND [ ,COMMANDS]
INTERVAL      ::= START[-END] COMMANDS
INTERVALS     ::= INTERVAL[ ;INTERVALS]
```

## 40.8.2 Examples# TOC

- Specify audio tempo change at second 4:

```
asendcmd=c='4.0 atempo tempo 1.5',atempo
```

- Specify a list of drawtext and hue commands in a file.

```
# show text in the interval 5-10
5.0-10.0 [enter] drawtext reinit 'fontfile=FreeSerif.ttf:text=hello world',
        [leave] drawtext reinit 'fontfile=FreeSerif.ttf:text=';

# desaturate the image in the interval 15-20
15.0-20.0 [enter] hue s 0,
        [enter] drawtext reinit 'fontfile=FreeSerif.ttf:text=nocolor',
        [leave] hue s 1,
        [leave] drawtext reinit 'fontfile=FreeSerif.ttf:text=color';

# apply an exponential saturation fade-out effect, starting from time 25
25 [enter] hue s exp(25-t)
```

A filtergraph allowing to read and process the above command list stored in a file `test.cmd`, can be specified with:

```
sendcmd=f=test.cmd,drawtext=fontfile=FreeSerif.ttf:text='',hue
```

## 40.9 setpts, asetpts# TOC

Change the PTS (presentation timestamp) of the input frames.

`setpts` works on video frames, `asetpts` on audio frames.

This filter accepts the following options:

`expr`

The expression which is evaluated for each frame to construct its timestamp.

The expression is evaluated through the eval API and can contain the following constants:

FRAME\_RATE

frame rate, only defined for constant frame-rate video

PTS

The presentation timestamp in input

N

The count of the input frame for video or the number of consumed samples, not including the current frame for audio, starting from 0.

NB\_CONSUMED\_SAMPLES

The number of consumed samples, not including the current frame (only audio)

NB\_SAMPLES, S

The number of samples in the current frame (only audio)

SAMPLE\_RATE, SR

The audio sample rate.

STARTPTS

The PTS of the first frame.

STARTT

the time in seconds of the first frame

INTERLACED

State whether the current frame is interlaced.

T

the time in seconds of the current frame

POS

original position in the file of the frame, or undefined if undefined for the current frame

PREV\_INPTS

The previous input PTS.

PREV\_INT

previous input time in seconds

PREV\_OUTPTS

The previous output PTS.

PREV\_OUTT

previous output time in seconds

RTCTIME

The wallclock (RTC) time in microseconds. This is deprecated, use time(0) instead.

RTCSTART

The wallclock (RTC) time at the start of the movie in microseconds.

TB

The timebase of the input timestamps.

#### 40.9.1 Examples# TOC

- Start counting PTS from zero

```
setpts=PTS-STARTPTS
```

- Apply fast motion effect:

```
setpts=0.5*PTS
```

- Apply slow motion effect:

```
setpts=2.0*PTS
```

- Set fixed rate of 25 frames per second:

```
setpts=N/(25*Tb)
```

- Set fixed rate 25 fps with some jitter:

```
setpts='1/(25*Tb) * (N + 0.05 * sin(N*2*PI/25))'
```

- Apply an offset of 10 seconds to the input PTS:

```
setpts=PTS+10/TB
```

- Generate timestamps from a "live source" and rebase onto the current timebase:

```
setpts='(RTCTIME - RTCSTART) / (TB * 1000000)'
```

- Generate timestamps by counting samples:

```
asetpts=N/SR/TB
```

## 40.10 settb, asetb# TOC

Set the timebase to use for the output frames timestamps. It is mainly useful for testing timebase configuration.

It accepts the following parameters:

`expr`, `tb`

The expression which is evaluated into the output timebase.

The value for `tb` is an arithmetic expression representing a rational. The expression can contain the constants "AVTB" (the default timebase), "intb" (the input timebase) and "sr" (the sample rate, audio only). Default value is "intb".

### 40.10.1 Examples# TOC

- Set the timebase to 1/25:

```
settb=expr=1/25
```

- Set the timebase to 1/10:

```
settb=expr=0.1
```

- Set the timebase to 1001/1000:

```
settb=1+0.001
```

- Set the timebase to 2\*intb:

```
settb=2*intb
```

- Set the default timebase value:

```
settb=AVTB
```

## 40.11 showcqt# TOC

Convert input audio to a video output representing frequency spectrum logarithmically (using constant Q transform with Brown-Puckette algorithm), with musical tone scale, from E0 to D#10 (10 octaves).

The filter accepts the following options:

volume

Specify transform volume (multiplier) expression. The expression can contain variables:

frequency, freq, f

the frequency where transform is evaluated

timeclamp, tc

value of timeclamp option

and functions:

a\_weighting(f)

A-weighting of equal loudness

b\_weighting(f)

B-weighting of equal loudness

c\_weighting(f)

C-weighting of equal loudness

Default value is 16.

tlength

Specify transform length expression. The expression can contain variables:

frequency, freq, f

the frequency where transform is evaluated

timeclamp, tc

value of timeclamp option

Default value is  $384/f*tc/(384/f+tc)$ .



timeclamp

Specify the transform timeclamp. At low frequency, there is trade-off between accuracy in time domain and frequency domain. If timeclamp is lower, event in time domain is represented more accurately (such as fast bass drum), otherwise event in frequency domain is represented more accurately (such as bass guitar). Acceptable value is [0.1, 1.0]. Default value is 0.17.

coeffclamp

Specify the transform coeffclamp. If coeffclamp is lower, transform is more accurate, otherwise transform is faster. Acceptable value is [0.1, 10.0]. Default value is 1.0.

gamma

Specify gamma. Lower gamma makes the spectrum more contrast, higher gamma makes the spectrum having more range. Acceptable value is [1.0, 7.0]. Default value is 3.0.

gamma2

Specify gamma of bargraph. Acceptable value is [1.0, 7.0]. Default value is 1.0.

fontfile

Specify font file for use with freetype. If not specified, use embedded font.

fontcolor

Specify font color expression. This is arithmetic expression that should return integer value 0xRRGGBB. The expression can contain variables:

frequency, freq, f

the frequency where transform is evaluated

timeclamp, tc

value of timeclamp option

and functions:

midi(f)

midi number of frequency f, some midi numbers: E0(16), C1(24), C2(36), A4(69)

r(x), g(x), b(x)

red, green, and blue value of intensity x

Default value is `st(0, (midi(f)-59.5)/12); st(1, if(between(ld(0),0,1), 0.5-0.5*cos(2*PI*ld(0)), 0)); r(1-lld(1)) + b(lld(1))`

`fullhd`

If set to 1 (the default), the video size is 1920x1080 (full HD), if set to 0, the video size is 960x540. Use this option to make CPU usage lower.

`fps`

Specify video fps. Default value is 25.

`count`

Specify number of transform per frame, so there are `fps*count` transforms per second. Note that audio data rate must be divisible by `fps*count`. Default value is 6.

### 40.11.1 Examples# TOC

- Playing audio while showing the spectrum:

```
ffplay -f lavfi 'amovie=a.mp3, asplit [a][out1]; [a] showcqt [out0]'
```

- Same as above, but with frame rate 30 fps:

```
ffplay -f lavfi 'amovie=a.mp3, asplit [a][out1]; [a] showcqt=fps=30:count=5 [out0]'
```

- Playing at 960x540 and lower CPU usage:

```
ffplay -f lavfi 'amovie=a.mp3, asplit [a][out1]; [a] showcqt=fullhd=0:count=3 [out0]'
```

- A1 and its harmonics: A1, A2, (near)E3, A3:

```
ffplay -f lavfi 'aevalsrc=0.1*sin(2*PI*55*t)+0.1*sin(4*PI*55*t)+0.1*sin(6*PI*55*t)+0.1*sin(8*PI*55*t), asplit[a][out1]; [a] showcqt [out0]'
```

- Same as above, but with more accuracy in frequency domain (and slower):

```
ffplay -f lavfi 'aevalsrc=0.1*sin(2*PI*55*t)+0.1*sin(4*PI*55*t)+0.1*sin(6*PI*55*t)+0.1*sin(8*PI*55*t), asplit[a][out1]; [a] showcqt=timeclamp=0.5 [out0]'
```

- B-weighting of equal loudness

```
volume=16*b_weighting(f)
```

- Lower Q factor

```
tlength=100/f*tc/(100/f+tc)
```

- Custom fontcolor, C-note is colored green, others are colored blue

```
fontcolor='if(mod(floor(midi(f)+0.5),12), 0x0000FF, g(1))'
```

- Custom gamma, now spectrum is linear to the amplitude.

```
gamma=2:gamma2=2
```

## 40.12 showfreqs# TOC

Convert input audio to video output representing the audio power spectrum. Audio amplitude is on Y-axis while frequency is on X-axis.

The filter accepts the following options:

`size, s`

Specify size of video. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default is 1024x512.

`mode`

Set display mode. This set how each frequency bin will be represented.

It accepts the following values:

```
'line'  
'bar'  
'dot'
```

Default is bar.

`ascale`

Set amplitude scale.

It accepts the following values:

```
'lin'
```

Linear scale.

```
'sqrt'
```

Square root scale.

```
'cbrt'
```

Cubic root scale.

`'log'`

Logarithmic scale.

Default is `log`.

`fscale`

Set frequency scale.

It accepts the following values:

`'lin'`

Linear scale.

`'log'`

Logarithmic scale.

`'rlog'`

Reverse logarithmic scale.

Default is `lin`.

`win_size`

Set window size.

It accepts the following values:

`'w16'`

`'w32'`

`'w64'`

`'w128'`

`'w256'`

`'w512'`

`'w1024'`

`'w2048'`

`'w4096'`

`'w8192'`

`'w16384'`

`'w32768'`

`'w65536'`

Default is `w2048`

`win_func`

Set windowing function.

It accepts the following values:

`'rect'`  
`'bartlett'`  
`'hanning'`  
`'hamming'`  
`'blackman'`  
`'welch'`  
`'flattop'`  
`'bharris'`  
`'bnuttall'`  
`'bhann'`  
`'sine'`  
`'nutall'`

Default is `hanning`.

`overlap`

Set window overlap. In range `[ 0 , 1 ]`. Default is 1, which means optimal overlap for selected window function will be picked.

`averaging`

Set time averaging. Setting this to 0 will display current maximal peaks. Default is 1, which means time averaging is disabled.

`color`

Specify list of colors separated by space or by `'|'` which will be used to draw channel frequencies. Unrecognized or missing colors will be replaced by white color.

## **40.13 showspectrum# TOC**

Convert input audio to a video output, representing the audio frequency spectrum.

The filter accepts the following options:

`size, s`

Specify the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 640x512.

slide

Specify how the spectrum should slide along the window.

It accepts the following values:

‘replace’

the samples start again on the left when they reach the right

‘scroll’

the samples scroll from right to left

‘fullframe’

frames are only produced when the samples reach the right

Default value is `replace`.

mode

Specify display mode.

It accepts the following values:

‘combined’

all channels are displayed in the same row

‘separate’

all channels are displayed in separate rows

Default value is ‘combined’.

color

Specify display color mode.

It accepts the following values:

‘channel’

each channel is displayed in a separate color

‘intensity’

each channel is displayed using the same color scheme

Default value is 'channel'.

scale

Specify scale used for calculating intensity color values.

It accepts the following values:

'lin'

linear

'sqrt'

square root, default

'cbrt'

cubic root

'log'

logarithmic

Default value is 'sqrt'.

saturation

Set saturation modifier for displayed colors. Negative values provide alternative color scheme. 0 is no saturation at all. Saturation must be in [-10.0, 10.0] range. Default value is 1.

win\_func

Set window function.

It accepts the following values:

'none'

No samples pre-processing (do not expect this to be faster)

'hann'

Hann window

'hamming'

Hamming window

`'blackman'`

Blackman window

Default value is hann.

The usage is very similar to the showwaves filter; see the examples in that section.

### 40.13.1 Examples# TOC

- Large window with logarithmic color scaling:

```
showspectrum=s=1280x480:scale=log
```

- Complete example for a colored and sliding spectrum per channel using ffmpeg:

```
ffmpeg -f lavfi 'amovie=input.mp3, asplit [a][out1];  
[a] showspectrum=mode=separate:color=intensity:slide=1:scale=cbrt [out0]'
```

### 40.14 showvolume# TOC

Convert input audio volume to a video output.

The filter accepts the following options:

rate, r

Set video rate.

b

Set border width, allowed range is [0, 5]. Default is 1.

w

Set channel width, allowed range is [40, 1080]. Default is 400.

h

Set channel height, allowed range is [1, 100]. Default is 20.

f

Set fade, allowed range is [1, 255]. Default is 20.

c



Set volume color expression.

The expression can use the following variables:

VOLUME

Current max volume of channel in dB.

CHANNEL

Current channel number, starting from 0.

t

If set, displays channel names. Default is enabled.

## 40.15 showwaves# TOC

Convert input audio to a video output, representing the samples waves.

The filter accepts the following options:

size, s

Specify the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 600x240.

mode

Set display mode.

Available values are:

‘point’

Draw a point for each sample.

‘line’

Draw a vertical line for each sample.

‘p2p’

Draw a point for each sample and a line between them.

‘cline’

Draw a centered vertical line for each sample.

Default value is `point`.

`n`

Set the number of samples which are printed on the same column. A larger value will decrease the frame rate. Must be a positive integer. This option can be set only if the value for *rate* is not explicitly specified.

`rate, r`

Set the (approximate) output frame rate. This is done by setting the option *n*. Default value is "25".

`split_channels`

Set if channels should be drawn separately or overlap. Default value is 0.

### 40.15.1 Examples# TOC

- Output the input file audio and the corresponding video representation at the same time:

```
amovie=a.mp3,asplit[out0],showwaves[out1]
```

- Create a synthetic signal and show it with showwaves, forcing a frame rate of 30 frames per second:

```
aevalsrc=sin(1*2*PI*t)*sin(880*2*PI*t):cos(2*PI*200*t),asplit[out0],showwaves=r=30[out1]
```

### 40.16 showwavespic# TOC

Convert input audio to a single video frame, representing the samples waves.

The filter accepts the following options:

`size, s`

Specify the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 600x240.

`split_channels`

Set if channels should be drawn separately or overlap. Default value is 0.

### 40.16.1 Examples# TOC

- Extract a channel split representation of the wave form of a whole audio track in a 1024x800 picture using ffmpeg:

```
ffmpeg -i audio.flac -lavfi showwavespic=split_channels=1:s=1024x800 waveform.png
```

## 40.17 split, asplit# TOC

Split input into several identical outputs.

`asplit` works with audio input, `split` with video.

The filter accepts a single parameter which specifies the number of outputs. If unspecified, it defaults to 2.

### 40.17.1 Examples# TOC

- Create two separate outputs from the same input:

```
[in] split [out0][out1]
```

- To create 3 or more outputs, you need to specify the number of outputs, like in:

```
[in] asplit=3 [out0][out1][out2]
```

- Create two separate outputs from the same input, one cropped and one padded:

```
[in] split [splitout1][splitout2];  
[splitout1] crop=100:100:0:0 [cropout];  
[splitout2] pad=200:200:100:100 [padout];
```

- Create 5 copies of the input audio with `ffmpeg`:

```
ffmpeg -i INPUT -filter_complex asplit=5 OUTPUT
```

## 40.18 zmq, azmq# TOC

Receive commands sent through a libzmq client, and forward them to filters in the filtergraph.

`zmq` and `azmq` work as a pass-through filters. `zmq` must be inserted between two video filters, `azmq` between two audio filters.

To enable these filters you need to install the libzmq library and headers and configure FFmpeg with `--enable-libzmq`.

For more information about libzmq see: <http://www.zeromq.org/>

The `zmq` and `azmq` filters work as a libzmq server, which receives messages sent through a network interface defined by the `bind_address` option.

The received message must be in the form:

```
TARGET COMMAND [ARG]
```

*TARGET* specifies the target of the command, usually the name of the filter class or a specific filter instance name.

*COMMAND* specifies the name of the command for the target filter.

*ARG* is optional and specifies the optional argument list for the given *COMMAND*.

Upon reception, the message is processed and the corresponding command is injected into the filtergraph. Depending on the result, the filter will send a reply to the client, adopting the format:

```
ERROR_CODE ERROR_REASON  
MESSAGE
```

*MESSAGE* is optional.

## 40.18.1 Examples# TOC

Look at `tools/zmqsend` for an example of a zmq client which can be used to send commands processed by these filters.

Consider the following filtergraph generated by `ffplay`

```
ffplay -dumpgraph 1 -f lavfi "  
color=s=100x100:c=red [l];  
color=s=100x100:c=blue [r];  
nullsrc=s=200x100, zmq [bg];  
[bg][l] overlay [bg+l];  
[bg+l][r] overlay=x=100 "
```

To change the color of the left side of the video, the following command can be used:

```
echo Parsed_color_0 c yellow | tools/zmqsend
```

To change the right side:

```
echo Parsed_color_1 c pink | tools/zmqsend
```

## 41 Multimedia Sources# TOC

Below is a description of the currently available multimedia sources.

### 41.1 amovie# TOC

This is the same as movie source, except it selects an audio stream by default.

### 41.2 movie# TOC

Read audio and/or video stream(s) from a movie container.

It accepts the following parameters:

filename

The name of the resource to read (not necessarily a file; it can also be a device or a stream accessed through some protocol).

format\_name, f

Specifies the format assumed for the movie to read, and can be either the name of a container or an input device. If not specified, the format is guessed from *movie\_name* or by probing.

seek\_point, sp

Specifies the seek point in seconds. The frames will be output starting from this seek point. The parameter is evaluated with `av_strtod`, so the numerical value may be suffixed by an IS postfix. The default value is "0".

streams, s

Specifies the streams to read. Several streams can be specified, separated by "+". The source will then have as many outputs, in the same order. The syntax is explained in the "Stream specifiers" section in the ffmpeg manual. Two special names, "dv" and "da" specify respectively the default (best suited) video and audio stream. Default is "dv", or "da" if the filter is called as "amovie".

stream\_index, si

Specifies the index of the video stream to read. If the value is -1, the most suitable video stream will be automatically selected. The default value is "-1". Deprecated. If the filter is called "amovie", it will select audio instead of video.

loop

Specifies how many times to read the stream in sequence. If the value is less than 1, the stream will be read again and again. Default value is "1".

Note that when the movie is looped the source timestamps are not changed, so it will generate non monotonically increasing timestamps.

It allows overlaying a second video on top of the main input of a filtergraph, as shown in this graph:

```
input -----> deltapts0 --> overlay --> output
                        ^
                        |
movie --> scale--> deltapts1 -----+
```

### 41.2.1 Examples# TOC

- Skip 3.2 seconds from the start of the AVI file in.avi, and overlay it on top of the input labelled "in":

```
movie=in.avi:seek_point=3.2, scale=180:-1, setpts=PTS-STARTPTS [over];  
[in] setpts=PTS-STARTPTS [main];  
[main][over] overlay=16:16 [out]
```

- Read from a video4linux2 device, and overlay it on top of the input labelled "in":

```
movie=/dev/video0:f=video4linux2, scale=180:-1, setpts=PTS-STARTPTS [over];  
[in] setpts=PTS-STARTPTS [main];  
[main][over] overlay=16:16 [out]
```

- Read the first video stream and the audio stream with id 0x81 from dvd.vob; the video is connected to the pad named "video" and the audio is connected to the pad named "audio":

```
movie=dvd.vob:s=v:0+#0x81 [video] [audio]
```

## 42 See Also# TOC

ffmpeg, ffplay, ffprobe, ffserver, ffmpeg-utils, ffmpeg-scaler, ffmpeg-resampler, ffmpeg-codecs, ffmpeg-bitstream-filters, ffmpeg-formats, ffmpeg-devices, ffmpeg-protocols, ffmpeg-filters

## 43 Authors# TOC

The FFmpeg developers.

For details about the authorship, see the Git history of the project ([git://source.ffmpeg.org/ffmpeg](http://source.ffmpeg.org/ffmpeg)), e.g. by typing the command `git log` in the FFmpeg source directory, or browsing the online repository at <http://source.ffmpeg.org>.

Maintainers for the specific components are listed in the file MAINTAINERS in the source code tree.

This document was generated using *makeinfo*.

# ffmpeg Documentation

## Table of Contents

- 1 Synopsis
- 2 Description
- 3 Detailed description
  - 3.1 Filtering
    - 3.1.1 Simple filtergraphs
    - 3.1.2 Complex filtergraphs
  - 3.2 Stream copy
- 4 Stream selection
- 5 Options
  - 5.1 Stream specifiers
  - 5.2 Generic options
  - 5.3 AVOptions
  - 5.4 Main options
  - 5.5 Video Options
  - 5.6 Advanced Video options
  - 5.7 Audio Options
  - 5.8 Advanced Audio options
  - 5.9 Subtitle options
  - 5.10 Advanced Subtitle options
  - 5.11 Advanced options
  - 5.12 Preset files
    - 5.12.1 ffpreset files
    - 5.12.2 avpreset files
- 6 Examples
  - 6.1 Video and Audio grabbing
  - 6.2 X11 grabbing
  - 6.3 Video and Audio file format conversion
- 7 Syntax
  - 7.1 Quoting and escaping
    - 7.1.1 Examples
  - 7.2 Date
  - 7.3 Time duration
    - 7.3.1 Examples
  - 7.4 Video size
  - 7.5 Video rate
  - 7.6 Ratio
  - 7.7 Color
  - 7.8 Channel Layout
- 8 Expression Evaluation

- 9 OpenCL Options
- 10 Codec Options
- 11 Decoders
- 12 Video Decoders
  - 12.1 hevc
  - 12.2 rawvideo
    - 12.2.1 Options
- 13 Audio Decoders
  - 13.1 ac3
    - 13.1.1 AC-3 Decoder Options
  - 13.2 flac
    - 13.2.1 FLAC Decoder options
  - 13.3 ffwavesynth
  - 13.4 libcelt
  - 13.5 libgsm
  - 13.6 libilbc
    - 13.6.1 Options
  - 13.7 libopencore-amrnb
  - 13.8 libopencore-amrwb
  - 13.9 libopus
- 14 Subtitles Decoders
  - 14.1 dvbsub
    - 14.1.1 Options
  - 14.2 dvdsup
    - 14.2.1 Options
  - 14.3 libzybi-teletext
    - 14.3.1 Options
- 15 Encoders
- 16 Audio Encoders
  - 16.1 aac
    - 16.1.1 Options
  - 16.2 ac3 and ac3\_fixed
    - 16.2.1 AC-3 Metadata
      - 16.2.1.1 Metadata Control Options
      - 16.2.1.2 Downmix Levels
      - 16.2.1.3 Audio Production Information
      - 16.2.1.4 Other Metadata Options
    - 16.2.2 Extended Bitstream Information
      - 16.2.2.1 Extended Bitstream Information - Part 1
      - 16.2.2.2 Extended Bitstream Information - Part 2
    - 16.2.3 Other AC-3 Encoding Options
    - 16.2.4 Floating-Point-Only AC-3 Encoding Options
  - 16.3 flac



- 16.3.1 Options
- 16.4 libfaac
  - 16.4.1 Options
  - 16.4.2 Examples
- 16.5 libfdk\_aac
  - 16.5.1 Options
  - 16.5.2 Examples
- 16.6 libmp3lame
  - 16.6.1 Options
- 16.7 libopencore-amrnb
  - 16.7.1 Options
- 16.8 libshine
  - 16.8.1 Options
- 16.9 libtwolame
  - 16.9.1 Options
- 16.10 libvo-aacenc
  - 16.10.1 Options
- 16.11 libvo-amrwbenc
  - 16.11.1 Options
- 16.12 libopus
  - 16.12.1 Option Mapping
- 16.13 libvorbis
  - 16.13.1 Options
- 16.14 libwavpack
  - 16.14.1 Options
- 16.15 wavpack
  - 16.15.1 Options
    - 16.15.1.1 Shared options
    - 16.15.1.2 Private options
- 17 Video Encoders
  - 17.1 jpeg2000
    - 17.1.1 Options
  - 17.2 snow
    - 17.2.1 Options
  - 17.3 libtheora
    - 17.3.1 Options
    - 17.3.2 Examples
  - 17.4 libvpx
    - 17.4.1 Options
  - 17.5 libwebp
    - 17.5.1 Pixel Format
    - 17.5.2 Options
  - 17.6 libx264, libx264rgb

- 17.6.1 Supported Pixel Formats
  - 17.6.2 Options
- 17.7 libx265
  - 17.7.1 Options
- 17.8 libxvid
  - 17.8.1 Options
- 17.9 mpeg2
  - 17.9.1 Options
- 17.10 png
  - 17.10.1 Private options
- 17.11 ProRes
  - 17.11.1 Private Options for prores-ks
  - 17.11.2 Speed considerations
- 17.12 libkvazaar
  - 17.12.1 Options
- 18 Subtitles Encoders
  - 18.1 dvdsub
    - 18.1.1 Options
- 19 Bitstream Filters
  - 19.1 aac\_adtstoasc
  - 19.2 chomp
  - 19.3 dump\_extra
  - 19.4 h264\_mp4toannexb
  - 19.5 imxdump
  - 19.6 mjpeg2jpeg
  - 19.7 mjpega\_dump\_header
  - 19.8 movsub
  - 19.9 mp3\_header\_decompress
  - 19.10 mpeg4\_unpack\_bframes
  - 19.11 noise
  - 19.12 remove\_extra
- 20 Format Options
  - 20.1 Format stream specifiers
- 21 Demuxers
  - 21.1 aa
  - 21.2 applehttp
  - 21.3 apng
  - 21.4 asf
  - 21.5 concat
    - 21.5.1 Syntax
    - 21.5.2 Options
  - 21.6 flv
  - 21.7 libgme

- 21.8 libquvi
- 21.9 gif
- 21.10 image2
  - 21.10.1 Examples
- 21.11 mpegts
- 21.12 rawvideo
- 21.13 sbg
- 21.14 tedcaptions
- 22 Muxers
  - 22.1 aiff
    - 22.1.1 Options
  - 22.2 crc
    - 22.2.1 Examples
  - 22.3 framecrc
    - 22.3.1 Examples
  - 22.4 framemd5
    - 22.4.1 Examples
  - 22.5 gif
  - 22.6 hls
    - 22.6.1 Options
  - 22.7 ico
  - 22.8 image2
    - 22.8.1 Examples
    - 22.8.2 Options
  - 22.9 matroska
    - 22.9.1 Metadata
    - 22.9.2 Options
  - 22.10 md5
  - 22.11 mov, mp4, ismv
    - 22.11.1 Options
    - 22.11.2 Example
    - 22.11.3 Audible AAX
  - 22.12 mp3
  - 22.13 mpegts
    - 22.13.1 Options
    - 22.13.2 Example
  - 22.14 null
  - 22.15 nut
  - 22.16 ogg
  - 22.17 segment, stream\_segment, ssegment
    - 22.17.1 Options
    - 22.17.2 Examples
  - 22.18 smoothstreaming

- 22.19 tee
  - 22.19.1 Examples
- 22.20 webm\_dash\_manifest
  - 22.20.1 Options
  - 22.20.2 Example
- 22.21 webm\_chunk
  - 22.21.1 Options
  - 22.21.2 Example
- 23 Metadata
- 24 Protocols
  - 24.1 async
  - 24.2 bluray
  - 24.3 cache
  - 24.4 concat
  - 24.5 crypto
  - 24.6 data
  - 24.7 file
  - 24.8 ftp
  - 24.9 gopher
  - 24.10 hls
  - 24.11 http
    - 24.11.1 HTTP Cookies
  - 24.12 Icecast
  - 24.13 mmst
  - 24.14 mmsh
  - 24.15 md5
  - 24.16 pipe
  - 24.17 rtmp
  - 24.18 rtmpe
  - 24.19 rtmps
  - 24.20 rtmpt
  - 24.21 rtmpte
  - 24.22 rtmpts
  - 24.23 libsmbclient
  - 24.24 libssh
  - 24.25 librtmp rtmp, rtmpe, rtmps, rtmpt, rtmpte
  - 24.26 rtp
  - 24.27 rtsp
    - 24.27.1 Examples
  - 24.28 sap
    - 24.28.1 Muxer
    - 24.28.2 Demuxer
  - 24.29 sctp

- 24.30 srtp
- 24.31 subfile
- 24.32 tcp
- 24.33 tls
- 24.34 udp
  - 24.34.1 Examples
- 24.35 unix
- 25 Device Options
- 26 Input Devices
  - 26.1 alsa
    - 26.1.1 Options
  - 26.2 avfoundation
    - 26.2.1 Options
    - 26.2.2 Examples
  - 26.3 bktr
    - 26.3.1 Options
  - 26.4 decklink
    - 26.4.1 Options
    - 26.4.2 Examples
  - 26.5 dshow
    - 26.5.1 Options
    - 26.5.2 Examples
  - 26.6 dv1394
    - 26.6.1 Options
  - 26.7 fbdev
    - 26.7.1 Options
  - 26.8 gdigrab
    - 26.8.1 Options
  - 26.9 iec61883
    - 26.9.1 Options
    - 26.9.2 Examples
  - 26.10 jack
    - 26.10.1 Options
  - 26.11 lavfi
    - 26.11.1 Options
    - 26.11.2 Examples
  - 26.12 libcdio
    - 26.12.1 Options
  - 26.13 libdc1394
  - 26.14 openal
    - 26.14.1 Options
    - 26.14.2 Examples
  - 26.15 oss

- 26.15.1 Options
- 26.16 pulse
  - 26.16.1 Options
  - 26.16.2 Examples
- 26.17 qtkit
  - 26.17.1 Options
- 26.18 sndio
  - 26.18.1 Options
- 26.19 video4linux2, v4l2
  - 26.19.1 Options
- 26.20 vfwcap
  - 26.20.1 Options
- 26.21 x11grab
  - 26.21.1 Options
  - 26.21.2 *grab\_x grab\_y* AVOption
- 27 Output Devices
  - 27.1 alsa
    - 27.1.1 Examples
  - 27.2 caca
    - 27.2.1 Options
    - 27.2.2 Examples
  - 27.3 decklink
    - 27.3.1 Options
    - 27.3.2 Examples
  - 27.4 fbdev
    - 27.4.1 Options
    - 27.4.2 Examples
  - 27.5 opengl
    - 27.5.1 Options
    - 27.5.2 Examples
  - 27.6 oss
  - 27.7 pulse
    - 27.7.1 Options
    - 27.7.2 Examples
  - 27.8 sdl
    - 27.8.1 Options
    - 27.8.2 Interactive commands
    - 27.8.3 Examples
  - 27.9 sndio
  - 27.10 xv
    - 27.10.1 Options
    - 27.10.2 Examples
- 28 Resampler Options

- 29 Scaler Options
- 30 Filtering Introduction
- 31 graph2dot
- 32 Filtergraph description
  - 32.1 Filtergraph syntax
  - 32.2 Notes on filtergraph escaping
- 33 Timeline editing
- 34 Audio Filters
  - 34.1 acrossfade
    - 34.1.1 Examples
  - 34.2 adelay
    - 34.2.1 Examples
  - 34.3 aecho
    - 34.3.1 Examples
  - 34.4 aeval
    - 34.4.1 Examples
  - 34.5 afade
    - 34.5.1 Examples
  - 34.6 aformat
  - 34.7 allpass
  - 34.8 amerge
    - 34.8.1 Examples
  - 34.9 amix
  - 34.10 anull
  - 34.11 apad
    - 34.11.1 Examples
  - 34.12 aphasor
  - 34.13 aresample
    - 34.13.1 Examples
  - 34.14 asetnsamples
  - 34.15 asetrate
  - 34.16 ashowinfo
  - 34.17 astats
  - 34.18 astreamsync
    - 34.18.1 Examples
  - 34.19 asyncts
  - 34.20 atempo
    - 34.20.1 Examples
  - 34.21 atrim
  - 34.22 bandpass
  - 34.23 bandreject
  - 34.24 bass
  - 34.25 biquad

- 34.26 bs2b
- 34.27 channelmap
- 34.28 channelsplit
- 34.29 chorus
  - 34.29.1 Examples
- 34.30 compand
  - 34.30.1 Examples
- 34.31 dcsift
- 34.32 dynaudnorm
- 34.33 earwax
- 34.34 equalizer
  - 34.34.1 Examples
- 34.35 flanger
- 34.36 highpass
- 34.37 join
- 34.38 ladspa
  - 34.38.1 Examples
  - 34.38.2 Commands
- 34.39 lowpass
- 34.40 pan
  - 34.40.1 Mixing examples
  - 34.40.2 Remapping examples
- 34.41 replaygain
- 34.42 resample
- 34.43 sidechaincompress
  - 34.43.1 Examples
- 34.44 silencedetect
  - 34.44.1 Examples
- 34.45 silenceremove
  - 34.45.1 Examples
- 34.46 treble
- 34.47 volume
  - 34.47.1 Commands
  - 34.47.2 Examples
- 34.48 volumedetect
  - 34.48.1 Examples
- 35 Audio Sources
  - 35.1 abuffer
    - 35.1.1 Examples
  - 35.2 aevalsrc
    - 35.2.1 Examples
  - 35.3 anullsrc
    - 35.3.1 Examples



- 35.4 flite
  - 35.4.1 Examples
- 35.5 sine
  - 35.5.1 Examples
- 36 Audio Sinks
  - 36.1 abuffersink
  - 36.2 anullsink
- 37 Video Filters
  - 37.1 alphaextract
  - 37.2 alphamerge
  - 37.3 ass
  - 37.4 atadenoise
  - 37.5 bbox
  - 37.6 blackdetect
  - 37.7 blackframe
  - 37.8 blend, tblend
    - 37.8.1 Examples
  - 37.9 boxblur
    - 37.9.1 Examples
  - 37.10 codecview
    - 37.10.1 Examples
  - 37.11 colorbalance
    - 37.11.1 Examples
  - 37.12 colorkey
    - 37.12.1 Examples
  - 37.13 colorlevels
    - 37.13.1 Examples
  - 37.14 colorchannelmixer
    - 37.14.1 Examples
  - 37.15 colormatrix
  - 37.16 copy
  - 37.17 crop
    - 37.17.1 Examples
    - 37.17.2 Commands
  - 37.18 cropdetect
  - 37.19 curves
    - 37.19.1 Examples
  - 37.20 dctdnoiz
    - 37.20.1 Examples
  - 37.21 deband
  - 37.22 decimate
  - 37.23 deflate
  - 37.24 dejudder

- 37.25 delogo
  - 37.25.1 Examples
- 37.26 deshake
- 37.27 detelecine
- 37.28 dilation
- 37.29 drawbox
  - 37.29.1 Examples
- 37.30 drawgraph, adrawgraph
- 37.31 drawgrid
  - 37.31.1 Examples
- 37.32 drawtext
  - 37.32.1 Syntax
  - 37.32.2 Text expansion
  - 37.32.3 Examples
- 37.33 edgedetect
  - 37.33.1 Examples
- 37.34 eq
  - 37.34.1 Commands
- 37.35 erosion
- 37.36 extractplanes
  - 37.36.1 Examples
- 37.37 elbg
- 37.38 fade
  - 37.38.1 Examples
- 37.39 fftfilt
  - 37.39.1 Examples
- 37.40 field
- 37.41 fieldmatch
  - 37.41.1 p/c/n/u/b meaning
    - 37.41.1.1 p/c/n
    - 37.41.1.2 u/b
  - 37.41.2 Examples
- 37.42 fieldorder
- 37.43 fifo
- 37.44 find\_rect
  - 37.44.1 Examples
- 37.45 cover\_rect
  - 37.45.1 Examples
- 37.46 format
  - 37.46.1 Examples
- 37.47 fps
  - 37.47.1 Examples
- 37.48 framepack

- 37.49 framerate
- 37.50 framestep
- 37.51 frei0r
  - 37.51.1 Examples
- 37.52 fspp
- 37.53 geq
  - 37.53.1 Examples
- 37.54 gradfun
  - 37.54.1 Examples
- 37.55 haldclut
  - 37.55.1 Workflow examples
    - 37.55.1.1 Hald CLUT video stream
    - 37.55.1.2 Hald CLUT with preview
- 37.56 hflip
- 37.57 histeq
- 37.58 histogram
  - 37.58.1 Examples
- 37.59 hqdn3d
- 37.60 hqx
- 37.61 hstack
- 37.62 hue
  - 37.62.1 Examples
  - 37.62.2 Commands
- 37.63 idet
- 37.64 il
- 37.65 inflate
- 37.66 interlace
- 37.67 kerndeint
  - 37.67.1 Examples
- 37.68 lenscorrection
  - 37.68.1 Options
- 37.69 lut3d
- 37.70 lut, lutrgb, lutyuv
  - 37.70.1 Examples
- 37.71 mergeplanes
  - 37.71.1 Examples
- 37.72 mcdeint
- 37.73 mpdecimate
- 37.74 negate
- 37.75 noformat
  - 37.75.1 Examples
- 37.76 noise
  - 37.76.1 Examples

- 37.77 null
- 37.78 ocv
  - 37.78.1 dilate
  - 37.78.2 erode
  - 37.78.3 smooth
- 37.79 overlay
  - 37.79.1 Commands
  - 37.79.2 Examples
- 37.80 owdenoise
- 37.81 pad
  - 37.81.1 Examples
- 37.82 paletteen
- 37.82.1 Examples
- 37.83 paletteuse
  - 37.83.1 Examples
- 37.84 perspective
- 37.85 phase
- 37.86 pixdesctest
- 37.87 pp
  - 37.87.1 Examples
- 37.88 pp7
- 37.89 psnr
- 37.90 pullup
- 37.91 qp
  - 37.91.1 Examples
- 37.92 random
- 37.93 removegrain
- 37.94 removelogo
- 37.95 repeatfields
- 37.96 reverse, areverse
  - 37.96.1 Examples
- 37.97 rotate
  - 37.97.1 Examples
  - 37.97.2 Commands
- 37.98 sab
- 37.99 scale
  - 37.99.1 Options
  - 37.99.2 Examples
  - 37.99.3 Commands
- 37.100 scale2ref
  - 37.100.1 Examples
- 37.101 separatefields
- 37.102 setdar, setsar

- 37.102.1 Examples
- 37.103 setfield
- 37.104 showinfo
- 37.105 showpalette
- 37.106 shuffleplanes
- 37.107 signalstats
  - 37.107.1 Examples
- 37.108 smartblur
- 37.109 ssim
- 37.110 stereo3d
  - 37.110.1 Examples
- 37.111 spp
- 37.112 subtitles
- 37.113 super2xsai
- 37.114 swapuv
- 37.115 telecine
- 37.116 thumbnail
  - 37.116.1 Examples
- 37.117 tile
  - 37.117.1 Examples
- 37.118 tinterlace
- 37.119 transpose
- 37.120 trim
- 37.121 unsharp
  - 37.121.1 Examples
- 37.122 uspp
- 37.123 vectorscope
- 37.124 vidstabdetect
  - 37.124.1 Examples
- 37.125 vidstabtransform
  - 37.125.1 Options
  - 37.125.2 Examples
- 37.126 vflip
- 37.127 vignette
  - 37.127.1 Expressions
  - 37.127.2 Examples
- 37.128 vstack
- 37.129 w3dif
- 37.130 waveform
- 37.131 xbr
- 37.132 yadif
- 37.133 zoompan
  - 37.133.1 Examples

- 38 Video Sources
  - 38.1 buffer
  - 38.2 cellauto
    - 38.2.1 Examples
  - 38.3 mandelbrot
  - 38.4 mptestsrc
  - 38.5 frei0r\_src
  - 38.6 life
    - 38.6.1 Examples
  - 38.7 allrgb, allyuv, color, haldclutsrc, nullsrc, rgbtestsrc, smptebars, smptehtbars, testsrc
    - 38.7.1 Commands
- 39 Video Sinks
  - 39.1 buffersink
  - 39.2 nullsink
- 40 Multimedia Filters
  - 40.1 aphasemeter
  - 40.2 avectorscope
    - 40.2.1 Examples
  - 40.3 concat
    - 40.3.1 Examples
  - 40.4 ebur128
    - 40.4.1 Examples
  - 40.5 interleave, ainterleave
    - 40.5.1 Examples
  - 40.6 perms, aperms
  - 40.7 select, aselect
    - 40.7.1 Examples
  - 40.8 sendcmd, asendcmd
    - 40.8.1 Commands syntax
    - 40.8.2 Examples
  - 40.9 setpts, asetpts
    - 40.9.1 Examples
  - 40.10 settb, asettb
    - 40.10.1 Examples
  - 40.11 showcqt
    - 40.11.1 Examples
  - 40.12 showfreqs
  - 40.13 showspectrum
    - 40.13.1 Examples
  - 40.14 showvolume
  - 40.15 showwaves
    - 40.15.1 Examples
  - 40.16 showwavespic

- 40.16.1 Examples
- 40.17 split, asplit
  - 40.17.1 Examples
- 40.18 zmq, azmq
  - 40.18.1 Examples
- 41 Multimedia Sources
  - 41.1 amovie
  - 41.2 movie
    - 41.2.1 Examples
- 42 See Also
- 43 Authors

## 1 Synopsis# TOC

```
ffmpeg [global_options] {[input_file_options] -i input_file} ... {[output_file_options]
output_file} ...
```

## 2 Description# TOC

`ffmpeg` is a very fast video and audio converter that can also grab from a live audio/video source. It can also convert between arbitrary sample rates and resize video on the fly with a high quality polyphase filter.

`ffmpeg` reads from an arbitrary number of input "files" (which can be regular files, pipes, network streams, grabbing devices, etc.), specified by the `-i` option, and writes to an arbitrary number of output "files", which are specified by a plain output filename. Anything found on the command line which cannot be interpreted as an option is considered to be an output filename.

Each input or output file can, in principle, contain any number of streams of different types (video/audio/subtitle/attachment/data). The allowed number and/or types of streams may be limited by the container format. Selecting which streams from which inputs will go into which output is either done automatically or with the `-map` option (see the Stream selection chapter).

To refer to input files in options, you must use their indices (0-based). E.g. the first input file is 0, the second is 1, etc. Similarly, streams within a file are referred to by their indices. E.g. `2 : 3` refers to the fourth stream in the third input file. Also see the Stream specifiers chapter.

As a general rule, options are applied to the next specified file. Therefore, order is important, and you can have the same option on the command line multiple times. Each occurrence is then applied to the next input or output file. Exceptions from this rule are the global options (e.g. verbosity level), which should be specified first.

Do not mix input and output files – first specify all input files, then all output files. Also do not mix options which belong to different files. All options apply **ONLY** to the next input or output file and are reset between files.

- To set the video bitrate of the output file to 64 kbit/s:

```
ffmpeg -i input.avi -b:v 64k -bufsize 64k output.avi
```

- To force the frame rate of the output file to 24 fps:

```
ffmpeg -i input.avi -r 24 output.avi
```

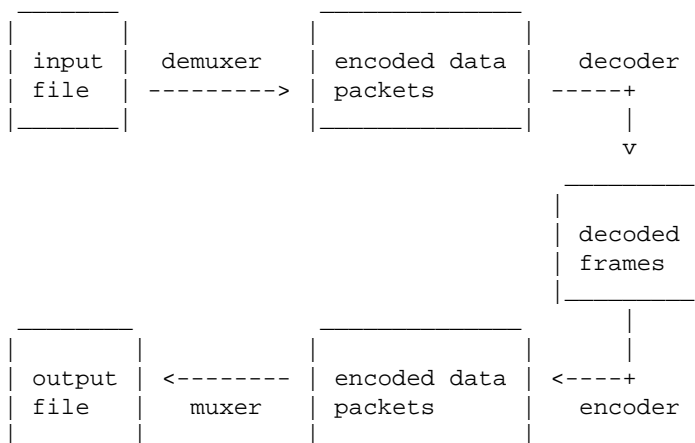
- To force the frame rate of the input file (valid for raw formats only) to 1 fps and the frame rate of the output file to 24 fps:

```
ffmpeg -r 1 -i input.m2v -r 24 output.avi
```

The format option may be needed for raw input files.

### 3 Detailed description# TOC

The transcoding process in `ffmpeg` for each output can be described by the following diagram:



`ffmpeg` calls the `libavformat` library (containing demuxers) to read input files and get packets containing encoded data from them. When there are multiple input files, `ffmpeg` tries to keep them synchronized by tracking lowest timestamp on any active input stream.

Encoded packets are then passed to the decoder (unless `streamcopy` is selected for the stream, see further for a description). The decoder produces uncompressed frames (raw video/PCM audio/...) which can be processed further by filtering (see next section). After filtering, the frames are passed to the encoder, which encodes them and outputs encoded packets. Finally those are passed to the muxer, which writes the encoded packets to the output file.



## 3.1 Filtering# TOC

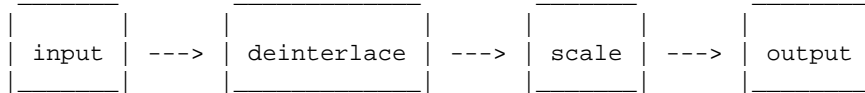
Before encoding, `ffmpeg` can process raw audio and video frames using filters from the `libavfilter` library. Several chained filters form a filter graph. `ffmpeg` distinguishes between two types of filtergraphs: simple and complex.

### 3.1.1 Simple filtergraphs# TOC

Simple filtergraphs are those that have exactly one input and output, both of the same type. In the above diagram they can be represented by simply inserting an additional step between decoding and encoding:



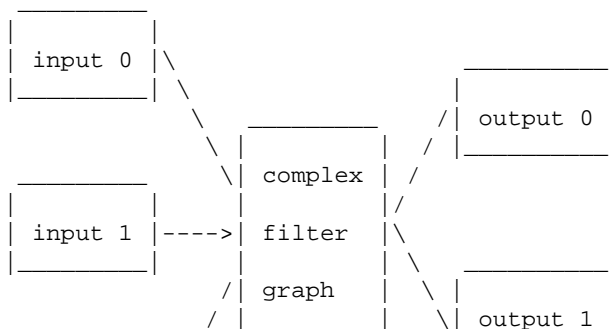
Simple filtergraphs are configured with the per-stream `-filter` option (with `-vf` and `-af` aliases for video and audio respectively). A simple filtergraph for video can look for example like this:



Note that some filters change frame properties but not frame contents. E.g. the `fps` filter in the example above changes number of frames, but does not touch the frame contents. Another example is the `setpts` filter, which only sets timestamps and otherwise passes the frames unchanged.

### 3.1.2 Complex filtergraphs# TOC

Complex filtergraphs are those which cannot be described as simply a linear processing chain applied to one stream. This is the case, for example, when the graph has more than one input and/or output, or when output stream type is different from input. They can be represented with the following diagram:





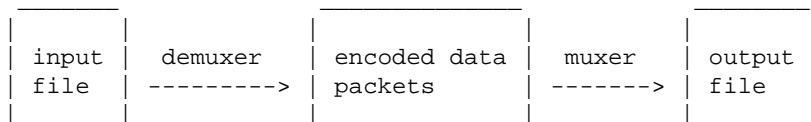
Complex filtergraphs are configured with the `-filter_complex` option. Note that this option is global, since a complex filtergraph, by its nature, cannot be unambiguously associated with a single stream or file.

The `-lavfi` option is equivalent to `-filter_complex`.

A trivial example of a complex filtergraph is the `overlay` filter, which has two video inputs and one video output, containing one video overlaid on top of the other. Its audio counterpart is the `amix` filter.

## 3.2 Stream copy# TOC

Stream copy is a mode selected by supplying the `copy` parameter to the `-codec` option. It makes `ffmpeg` omit the decoding and encoding step for the specified stream, so it does only demuxing and muxing. It is useful for changing the container format or modifying container-level metadata. The diagram above will, in this case, simplify to this:



Since there is no decoding or encoding, it is very fast and there is no quality loss. However, it might not work in some cases because of many factors. Applying filters is obviously also impossible, since filters work on uncompressed data.

## 4 Stream selection# TOC

By default, `ffmpeg` includes only one stream of each type (video, audio, subtitle) present in the input files and adds them to each output file. It picks the "best" of each based upon the following criteria: for video, it is the stream with the highest resolution, for audio, it is the stream with the most channels, for subtitles, it is the first subtitle stream. In the case where several streams of the same type rate equally, the stream with the lowest index is chosen.

You can disable some of those defaults by using the `-vn`/`-an`/`-sn` options. For full manual control, use the `-map` option, which disables the defaults just described.

## 5 Options# TOC

All the numerical options, if not specified otherwise, accept a string representing a number as input, which may be followed by one of the SI unit prefixes, for example: 'K', 'M', or 'G'.

If 'i' is appended to the SI unit prefix, the complete prefix will be interpreted as a unit prefix for binary multiples, which are based on powers of 1024 instead of powers of 1000. Appending 'B' to the SI unit prefix multiplies the value by 8. This allows using, for example: 'KB', 'MiB', 'G' and 'B' as number suffixes.

Options which do not take arguments are boolean options, and set the corresponding value to true. They can be set to false by prefixing the option name with "no". For example using "-nofoo" will set the boolean option with name "foo" to false.

## 5.1 Stream specifiers# TOC

Some options are applied per-stream, e.g. bitrate or codec. Stream specifiers are used to precisely specify which stream(s) a given option belongs to.

A stream specifier is a string generally appended to the option name and separated from it by a colon. E.g. `-codec:a:1 ac3` contains the `a:1` stream specifier, which matches the second audio stream. Therefore, it would select the ac3 codec for the second audio stream.

A stream specifier can match several streams, so that the option is applied to all of them. E.g. the stream specifier in `-b:a 128k` matches all audio streams.

An empty stream specifier matches all streams. For example, `-codec copy` or `-codec: copy` would copy all the streams without reencoding.

Possible forms of stream specifiers are:

*stream\_index*

Matches the stream with this index. E.g. `-threads:1 4` would set the thread count for the second stream to 4.

*stream\_type[:stream\_index]*

*stream\_type* is one of following: 'v' or 'V' for video, 'a' for audio, 's' for subtitle, 'd' for data, and 't' for attachments. 'v' matches all video streams, 'V' only matches video streams which are not attached pictures, video thumbnails or cover arts. If *stream\_index* is given, then it matches stream number *stream\_index* of this type. Otherwise, it matches all streams of this type.

*p:program\_id[:stream\_index]*

If *stream\_index* is given, then it matches the stream with number *stream\_index* in the program with the id *program\_id*. Otherwise, it matches all streams in the program.

*#stream\_id* or *i:stream\_id*

Match the stream by stream id (e.g. PID in MPEG-TS container).

`m:key[:value]`

Matches streams with the metadata tag *key* having the specified value. If *value* is not given, matches streams that contain the given tag with any value.

`u`

Matches streams with usable configuration, the codec must be defined and the essential information such as video dimension or audio sample rate must be present.

Note that in `ffmpeg`, matching by metadata will only work properly for input files.

## 5.2 Generic options# TOC

These options are shared amongst the `ff*` tools.

`-L`

Show license.

`-h, -?, -help, --help [arg]`

Show help. An optional parameter may be specified to print help about a specific item. If no argument is specified, only basic (non advanced) tool options are shown.

Possible values of *arg* are:

`long`

Print advanced tool options in addition to the basic tool options.

`full`

Print complete list of options, including shared and private options for encoders, decoders, demuxers, muxers, filters, etc.

`decoder=decoder_name`

Print detailed information about the decoder named *decoder\_name*. Use the `-decoders` option to get a list of all decoders.

`encoder=encoder_name`

Print detailed information about the encoder named *encoder\_name*. Use the `-encoders` option to get a list of all encoders.

`demuxer=demuxer_name`

Print detailed information about the demuxer named *demuxer\_name*. Use the `-formats` option to get a list of all demuxers and muxers.

`muxer=muxer_name`

Print detailed information about the muxer named *muxer\_name*. Use the `-formats` option to get a list of all muxers and demuxers.

`filter=filter_name`

Print detailed information about the filter name *filter\_name*. Use the `-filters` option to get a list of all filters.

`-version`

Show version.

`-formats`

Show available formats (including devices).

`-devices`

Show available devices.

`-codecs`

Show all codecs known to libavcodec.

Note that the term 'codec' is used throughout this documentation as a shortcut for what is more correctly called a media bitstream format.

`-decoders`

Show available decoders.

`-encoders`

Show all available encoders.

`-bsfs`

Show available bitstream filters.

`-protocols`

Show available protocols.

`-filters`

Show available libavfilter filters.

`-pix_fmts`

Show available pixel formats.

`-sample_fmts`

Show available sample formats.

`-layouts`

Show channel names and standard channel layouts.

`-colors`

Show recognized color names.

`-sources device[,opt1=val1[,opt2=val2]...]`

Show autodetected sources of the input device. Some devices may provide system-dependent source names that cannot be autodetected. The returned list cannot be assumed to be always complete.

```
ffmpeg -sources pulse,server=192.168.0.4
```

`-sinks device[,opt1=val1[,opt2=val2]...]`

Show autodetected sinks of the output device. Some devices may provide system-dependent sink names that cannot be autodetected. The returned list cannot be assumed to be always complete.

```
ffmpeg -sinks pulse,server=192.168.0.4
```

`-loglevel [repeat+]loglevel | -v [repeat+]loglevel`

Set the logging level used by the library. Adding "repeat+" indicates that repeated log output should not be compressed to the first line and the "Last message repeated n times" line will be omitted. "repeat" can also be used alone. If "repeat" is used alone, and with no prior loglevel set, the default loglevel will be used. If multiple loglevel parameters are given, using 'repeat' will not change the loglevel. *loglevel* is a string or a number containing one of the following values:

`'quiet, -8'`

Show nothing at all; be silent.

`'panic, 0'`

Only show fatal errors which could lead the process to crash, such as and assert failure. This is not currently used for anything.

`'fatal, 8'`

Only show fatal errors. These are errors after which the process absolutely cannot continue after.

`'error, 16'`

Show all errors, including ones which can be recovered from.

`'warning, 24'`

Show all warnings and errors. Any message related to possibly incorrect or unexpected events will be shown.

`'info, 32'`

Show informative messages during processing. This is in addition to warnings and errors. This is the default value.

`'verbose, 40'`

Same as `info`, except more verbose.

`'debug, 48'`

Show everything, including debugging information.

`'trace, 56'`

By default the program logs to `stderr`, if coloring is supported by the terminal, colors are used to mark errors and warnings. Log coloring can be disabled setting the environment variable `AV_LOG_FORCE_NOCOLOR` or `NO_COLOR`, or can be forced setting the environment variable `AV_LOG_FORCE_COLOR`. The use of the environment variable `NO_COLOR` is deprecated and will be dropped in a following FFmpeg version.

`-report`

Dump full command line and console output to a file named `program-YYYYMMDD-HHMMSS.log` in the current directory. This file can be useful for bug reports. It also implies `-loglevel verbose`.

Setting the environment variable `FFREPORT` to any value has the same effect. If the value is a `':'`-separated key=value sequence, these options will affect the report; option values must be escaped if they contain special characters or the options delimiter `':'` (see the “Quoting and escaping” section in the `ffmpeg-utils` manual).

The following options are recognized:

`file`

set the file name to use for the report; `%p` is expanded to the name of the program, `%t` is expanded to a timestamp, `%%` is expanded to a plain `%`

`level`

set the log verbosity level using a numerical value (see `-loglevel`).

For example, to output a report to a file named `ffreport.log` using a log level of 32 (alias for log level info):

```
FFREPORT=file=ffreport.log:level=32 ffmpeg -i input output
```

Errors in parsing the environment variable are not fatal, and will not appear in the report.

`-hide_banner`

Suppress printing banner.

All FFmpeg tools will normally show a copyright notice, build options and library versions. This option can be used to suppress printing this information.

`-cpuflags flags (global)`

Allows setting and clearing cpu flags. This option is intended for testing. Do not use it unless you know what you're doing.

```
ffmpeg -cpuflags -sse+mmx ...
ffmpeg -cpuflags mmx ...
ffmpeg -cpuflags 0 ...
```

Possible flags for this option are:

```
'x86'
  'mmx'
  'mmxext'
  'sse'
  'sse2'
  'sse2slow'
  'sse3'
  'sse3slow'
  'ssse3'
  'atom'
  'sse4.1'
  'sse4.2'
```



```

    'avx'
    'avx2'
    'xop'
    'fma3'
    'fma4'
    '3dnow'
    '3dnowext'
    'bmi1'
    'bmi2'
    'cmov'
'ARM'
    'armv5te'
    'armv6'
    'armv6t2'
    'vfp'
    'vfpv3'
    'neon'
    'setend'
'AArch64'
    'armv8'
    'vfp'
    'neon'
'PowerPC'
    'altivec'
'Specific Processors'
    'pentium2'
    'pentium3'
    'pentium4'
    'k6'
    'k62'
    'athlon'
    'athlonxp'
    'k8'
-opengl_bench

```

This option is used to benchmark all available OpenCL devices and print the results. This option is only available when FFmpeg has been compiled with `--enable-opengl`.

When FFmpeg is configured with `--enable-opengl`, the options for the global OpenCL context are set via `-opengl_options`. See the "OpenCL Options" section in the `ffmpeg-utils` manual for the complete list of supported options. Amongst others, these options include the ability to select a specific platform and device to run the OpenCL code on. By default, FFmpeg will run on the first device of the first platform. While the options for the global OpenCL context provide flexibility to the user in selecting the OpenCL device of their choice, most users would probably want to select the fastest OpenCL device for their system.

This option assists the selection of the most efficient configuration by identifying the appropriate device for the user's system. The built-in benchmark is run on all the OpenCL devices and the performance is measured for each device. The devices in the results list are sorted based on their performance with the fastest device listed first. The user can subsequently invoke `ffmpeg` using the device deemed most appropriate via `-opengl_options` to obtain the best performance for the OpenCL accelerated code.

Typical usage to use the fastest OpenCL device involve the following steps.

Run the command:

```
ffmpeg -opengl_bench
```

Note down the platform ID (*pidx*) and device ID (*didx*) of the first i.e. fastest device in the list. Select the platform and device using the command:

```
ffmpeg -opengl_options platform_idx=pidx:device_idx=didx ...
```

`-opengl_options options (global)`

Set OpenCL environment options. This option is only available when FFmpeg has been compiled with `--enable-opengl`.

*options* must be a list of *key=value* option pairs separated by `;`. See the “OpenCL Options” section in the `ffmpeg-utils` manual for the list of supported options.

## 5.3 AVOptions# TOC

These options are provided directly by the `libavformat`, `libavdevice` and `libavcodec` libraries. To see the list of available AVOptions, use the `-help` option. They are separated into two categories:

`generic`

These options can be set for any container, codec or device. Generic options are listed under `AVFormatContext` options for containers/devices and under `AVCodecContext` options for codecs.

`private`

These options are specific to the given container, device or codec. Private options are listed under their corresponding containers/devices/codecs.

For example to write an ID3v2.3 header instead of a default ID3v2.4 to an MP3 file, use the `id3v2_version` private option of the MP3 muxer:

```
ffmpeg -i input.flac -id3v2_version 3 out.mp3
```

All codec AVOptions are per-stream, and thus a stream specifier should be attached to them.

Note: the `-nooption` syntax cannot be used for boolean AVOptions, use `-option 0/-option 1`.

Note: the old undocumented way of specifying per-stream AVOptions by prepending `v/a/s` to the options name is now obsolete and will be removed soon.

## 5.4 Main options# TOC

`-f fmt (input/output)`

Force input or output file format. The format is normally auto detected for input files and guessed from the file extension for output files, so this option is not needed in most cases.

`-i filename (input)`

input file name

`-y (global)`

Overwrite output files without asking.

`-n (global)`

Do not overwrite output files, and exit immediately if a specified output file already exists.

`-c[:stream_specifier] codec (input/output,per-stream)`

`-codec[:stream_specifier] codec (input/output,per-stream)`

Select an encoder (when used before an output file) or a decoder (when used before an input file) for one or more streams. *codec* is the name of a decoder/encoder or a special value `copy` (output only) to indicate that the stream is not to be re-encoded.

For example

```
ffmpeg -i INPUT -map 0 -c:v libx264 -c:a copy OUTPUT
```

encodes all video streams with libx264 and copies all audio streams.

For each stream, the last matching `c` option is applied, so

```
ffmpeg -i INPUT -map 0 -c copy -c:v:1 libx264 -c:a:137 libvorbis OUTPUT
```

will copy all the streams except the second video, which will be encoded with libx264, and the 138th audio, which will be encoded with libvorbis.

`-t duration (input/output)`

When used as an input option (before `-i`), limit the *duration* of data read from the input file.

When used as an output option (before an output filename), stop writing the output after its duration reaches *duration*.

*duration* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

-to and -t are mutually exclusive and -t has priority.

-to *position* (*output*)

Stop writing the output at *position*. *position* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

-to and -t are mutually exclusive and -t has priority.

-fs *limit\_size* (*output*)

Set the file size limit, expressed in bytes.

-ss *position* (*input/output*)

When used as an input option (before -i), seeks in this input file to *position*. Note that in most formats it is not possible to seek exactly, so ffmpeg will seek to the closest seek point before *position*. When transcoding and -accurate\_seek is enabled (the default), this extra segment between the seek point and *position* will be decoded and discarded. When doing stream copy or when -noaccurate\_seek is used, it will be preserved.

When used as an output option (before an output filename), decodes but discards input until the timestamps reach *position*.

*position* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

-sseof *position* (*input/output*)

Like the -ss option but relative to the "end of file". That is negative values are earlier in the file, 0 is at EOF.

-itsoffset *offset* (*input*)

Set the input time offset.

*offset* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

The offset is added to the timestamps of the input files. Specifying a positive offset means that the corresponding streams are delayed by the time duration specified in *offset*.

`-timestamp date (output)`

Set the recording timestamp in the container.

*date* must be a date specification, see (ffmpeg-utils)the Date section in the ffmpeg-utils(1) manual.

`-metadata[:metadata_specifier] key=value (output,per-metadata)`

Set a metadata key/value pair.

An optional *metadata\_specifier* may be given to set metadata on streams or chapters. See `-map_metadata` documentation for details.

This option overrides metadata set with `-map_metadata`. It is also possible to delete metadata by using an empty value.

For example, for setting the title in the output file:

```
ffmpeg -i in.avi -metadata title="my title" out.flv
```

To set the language of the first audio stream:

```
ffmpeg -i INPUT -metadata:s:a:0 language=eng OUTPUT
```

`-target type (output)`

Specify target file type (vcd, svcd, dvd, dv, dv50). *type* may be prefixed with `pal-`, `ntsc-` or `film-` to use the corresponding standard. All the format options (bitrate, codecs, buffer sizes) are then set automatically. You can just type:

```
ffmpeg -i myfile.avi -target vcd /tmp/vcd.mpg
```

Nevertheless you can specify additional options as long as you know they do not conflict with the standard, as in:

```
ffmpeg -i myfile.avi -target vcd -bf 2 /tmp/vcd.mpg
```

`-dframes number (output)`

Set the number of data frames to output. This is an alias for `-frames:d`.

`-frames[:stream_specifier] framecount (output,per-stream)`

Stop writing to the stream after *framecount* frames.

`-q[:stream_specifier] q (output,per-stream)`

`-qscale[:stream_specifier] q (output,per-stream)`

Use fixed quality scale (VBR). The meaning of *q/qscale* is codec-dependent. If *qscale* is used without a *stream\_specifier* then it applies only to the video stream, this is to maintain compatibility with previous behavior and as specifying the same codec specific value to 2 different codecs that is audio

and video generally is not what is intended when no `stream_specifier` is used.

`-filter[:stream_specifier] filtergraph (output,per-stream)`

Create the filtergraph specified by *filtergraph* and use it to filter the stream.

*filtergraph* is a description of the filtergraph to apply to the stream, and must have a single input and a single output of the same type of the stream. In the filtergraph, the input is associated to the label `in`, and the output to the label `out`. See the `ffmpeg-filters` manual for more information about the filtergraph syntax.

See the `-filter_complex` option if you want to create filtergraphs with multiple inputs and/or outputs.

`-filter_script[:stream_specifier] filename (output,per-stream)`

This option is similar to `-filter`, the only difference is that its argument is the name of the file from which a filtergraph description is to be read.

`-pre[:stream_specifier] preset_name (output,per-stream)`

Specify the preset for matching stream(s).

`-stats (global)`

Print encoding progress/statistics. It is on by default, to explicitly disable it you need to specify `-nostats`.

`-progress url (global)`

Send program-friendly progress information to *url*.

Progress information is written approximately every second and at the end of the encoding process. It is made of "*key=value*" lines. *key* consists of only alphanumeric characters. The last key of a sequence of progress information is always "progress".

`-stdin`

Enable interaction on standard input. On by default unless standard input is used as an input. To explicitly disable interaction you need to specify `-nostdin`.

Disabling interaction on standard input is useful, for example, if `ffmpeg` is in the background process group. Roughly the same result can be achieved with `ffmpeg . . . < /dev/null` but it requires a shell.

`-debug_ts (global)`

Print timestamp information. It is off by default. This option is mostly useful for testing and debugging purposes, and the output format may change from one version to another, so it should not be employed by portable scripts.

See also the option `-fdebug ts`.

`-attach filename (output)`

Add an attachment to the output file. This is supported by a few formats like Matroska for e.g. fonts used in rendering subtitles. Attachments are implemented as a specific type of stream, so this option will add a new stream to the file. It is then possible to use per-stream options on this stream in the usual way. Attachment streams created with this option will be created after all the other streams (i.e. those created with `-map` or automatic mappings).

Note that for Matroska you also have to set the `mimetype` metadata tag:

```
ffmpeg -i INPUT -attach DejaVuSans.ttf -metadata:s:2 mimetype=application/x-truetype-font out.mkv
```

(assuming that the attachment stream will be third in the output file).

`-dump_attachment[:stream_specifier] filename (input,per-stream)`

Extract the matching attachment stream into a file named *filename*. If *filename* is empty, then the value of the `filename` metadata tag will be used.

E.g. to extract the first attachment to a file named 'out.ttf':

```
ffmpeg -dump_attachment:t:0 out.ttf -i INPUT
```

To extract all attachments to files determined by the `filename` tag:

```
ffmpeg -dump_attachment:t "" -i INPUT
```

Technical note – attachments are implemented as codec extradata, so this option can actually be used to extract extradata from any stream, not just attachments.

`-noautorotate`

Disable automatically rotating video based on file metadata.

## 5.5 Video Options# TOC

`-vframes number (output)`

Set the number of video frames to output. This is an alias for `-frames:v`.

`-r[:stream_specifier] fps (input/output,per-stream)`

Set frame rate (Hz value, fraction or abbreviation).

As an input option, ignore any timestamps stored in the file and instead generate timestamps assuming constant frame rate *fps*. This is not the same as the `-framerate` option used for some input formats like `image2` or `v4l2` (it used to be the same in older versions of FFmpeg). If in doubt use `-framerate` instead of the input option `-r`.

As an output option, duplicate or drop input frames to achieve constant output frame rate *fps*.

`-s[:stream_specifier] size (input/output,per-stream)`

Set frame size.

As an input option, this is a shortcut for the `video_size` private option, recognized by some demuxers for which the frame size is either not stored in the file or is configurable – e.g. raw video or video grabbers.

As an output option, this inserts the `scale` video filter to the *end* of the corresponding filtergraph. Please use the `scale` filter directly to insert it at the beginning or some other place.

The format is 'wxh' (default - same as source).

`-aspect[:stream_specifier] aspect (output,per-stream)`

Set the video display aspect ratio specified by *aspect*.

*aspect* can be a floating point number string, or a string of the form *num:den*, where *num* and *den* are the numerator and denominator of the aspect ratio. For example "4:3", "16:9", "1.3333", and "1.7777" are valid argument values.

If used together with `-vcodec copy`, it will affect the aspect ratio stored at container level, but not the aspect ratio stored in encoded frames, if it exists.

`-vn (output)`

Disable video recording.

`-vcodec codec (output)`

Set the video codec. This is an alias for `-codec:v`.

`-pass[:stream_specifier] n (output,per-stream)`

Select the pass number (1 or 2). It is used to do two-pass video encoding. The statistics of the video are recorded in the first pass into a log file (see also the option `-passlogfile`), and in the second pass that log file is used to generate the video at the exact requested bitrate. On pass 1, you may just deactivate audio and set output to null, examples for Windows and Unix:

```
ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y NUL
ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y /dev/null
```

`-passlogfile[:stream_specifier] prefix (output,per-stream)`

Set two-pass log file name prefix to *prefix*, the default file name prefix is "ffmpeg2pass". The complete file name will be `PREFIX-N.log`, where N is a number specific to the output stream



`-vf filtergraph (output)`

Create the filtergraph specified by *filtergraph* and use it to filter the stream.

This is an alias for `-filter:v`, see the `-filter` option.

## 5.6 Advanced Video options# TOC

`-pix_fmt[:stream_specifier] format (input/output,per-stream)`

Set pixel format. Use `-pix_fmts` to show all the supported pixel formats. If the selected pixel format can not be selected, ffmpeg will print a warning and select the best pixel format supported by the encoder. If *pix\_fmt* is prefixed by a +, ffmpeg will exit with an error if the requested pixel format can not be selected, and automatic conversions inside filtergraphs are disabled. If *pix\_fmt* is a single +, ffmpeg selects the same pixel format as the input (or graph output) and automatic conversions are disabled.

`-sws_flags flags (input/output)`

Set SwScaler flags.

`-vdt n`

Discard threshold.

`-rc_override[:stream_specifier] override (output,per-stream)`

Rate control override for specific intervals, formatted as "int,int,int" list separated with slashes. Two first values are the beginning and end frame numbers, last one is quantizer to use if positive, or quality factor if negative.

`-ilme`

Force interlacing support in encoder (MPEG-2 and MPEG-4 only). Use this option if your input file is interlaced and you want to keep the interlaced format for minimum losses. The alternative is to deinterlace the input stream with `-deinterlace`, but deinterlacing introduces losses.

`-psnr`

Calculate PSNR of compressed frames.

`-vstats`

Dump video coding statistics to `vstats_HHMMSS.log`.

`-vstats_file file`

Dump video coding statistics to *file*.

`-top[:stream_specifier] n (output,per-stream)`

top=1/bottom=0/auto=-1 field first

`-dc precision`

Intra\_dc\_precision.

`-vtag fourcc/tag (output)`

Force video tag/fourcc. This is an alias for `-tag:v`.

`-qphist (global)`

Show QP histogram

`-vbsf bitstream_filter`

Deprecated see `-bsf`

`-force_key_frames[:stream_specifier] time[,time...] (output,per-stream)`

`-force_key_frames[:stream_specifier] expr:expr (output,per-stream)`

Force key frames at the specified timestamps, more precisely at the first frames after each specified time.

If the argument is prefixed with `expr:`, the string *expr* is interpreted like an expression and is evaluated for each frame. A key frame is forced in case the evaluation is non-zero.

If one of the times is "chapters[*delta*]", it is expanded into the time of the beginning of all chapters in the file, shifted by *delta*, expressed as a time in seconds. This option can be useful to ensure that a seek point is present at a chapter mark or any other designated place in the output file.

For example, to insert a key frame at 5 minutes, plus key frames 0.1 second before the beginning of every chapter:

`-force_key_frames 0:05:00,chapters-0.1`

The expression in *expr* can contain the following constants:

*n*

the number of current processed frame, starting from 0

*n\_forced*

the number of forced frames

`prev_forced_n`

the number of the previous forced frame, it is NAN when no keyframe was forced yet

`prev_forced_t`

the time of the previous forced frame, it is NAN when no keyframe was forced yet

`t`

the time of the current processed frame

For example to force a key frame every 5 seconds, you can specify:

```
-force_key_frames expr:gte(t,n_forced*5)
```

To force a key frame 5 seconds after the time of the last forced one, starting from second 13:

```
-force_key_frames expr:if(isnan(prev_forced_t),gte(t,13),gte(t,prev_forced_t+5))
```

Note that forcing too many keyframes is very harmful for the lookahead algorithms of certain encoders: using fixed-GOP options or similar would be more efficient.

```
-copyinkf[:stream_specifier] (output,per-stream)
```

When doing stream copy, copy also non-key frames found at the beginning.

```
-hwaccel[:stream_specifier] hwaccel (input,per-stream)
```

Use hardware acceleration to decode the matching stream(s). The allowed values of *hwaccel* are:

`none`

Do not use any hardware acceleration (the default).

`auto`

Automatically select the hardware acceleration method.

`vda`

Use Apple VDA hardware acceleration.

`vdpa`

Use VDPAU (Video Decode and Presentation API for Unix) hardware acceleration.

dxva2

Use DXVA2 (DirectX Video Acceleration) hardware acceleration.

This option has no effect if the selected hwaccel is not available or not supported by the chosen decoder.

Note that most acceleration methods are intended for playback and will not be faster than software decoding on modern CPUs. Additionally, `ffmpeg` will usually need to copy the decoded frames from the GPU memory into the system memory, resulting in further performance loss. This option is thus mainly useful for testing.

`-hwaccel_device[:stream_specifier] hwaccel_device (input,per-stream)`

Select a device to use for hardware acceleration.

This option only makes sense when the `-hwaccel` option is also specified. Its exact meaning depends on the specific hardware acceleration method chosen.

vdpau

For VDPAU, this option specifies the X11 display/screen to use. If this option is not specified, the value of the `DISPLAY` environment variable is used

dxva2

For DXVA2, this option should contain the number of the display adapter to use. If this option is not specified, the default adapter is used.

`-hwaccels`

List all hardware acceleration methods supported in this build of `ffmpeg`.

## 5.7 Audio Options# TOC

`-aframes number (output)`

Set the number of audio frames to output. This is an alias for `-frames:a`.

`-ar[:stream_specifier] freq (input/output,per-stream)`

Set the audio sampling frequency. For output streams it is set by default to the frequency of the corresponding input stream. For input streams this option only makes sense for audio grabbing devices and raw demuxers and is mapped to the corresponding demuxer options.

`-aq q (output)`

Set the audio quality (codec-specific, VBR). This is an alias for `-q:a`.

`-ac[:stream_specifier] channels (input/output,per-stream)`

Set the number of audio channels. For output streams it is set by default to the number of input audio channels. For input streams this option only makes sense for audio grabbing devices and raw demuxers and is mapped to the corresponding demuxer options.

`-an (output)`

Disable audio recording.

`-acodec codec (input/output)`

Set the audio codec. This is an alias for `-codec:a`.

`-sample_fmt[:stream_specifier] sample_fmt (output,per-stream)`

Set the audio sample format. Use `-sample_fmts` to get a list of supported sample formats.

`-af filtergraph (output)`

Create the filtergraph specified by *filtergraph* and use it to filter the stream.

This is an alias for `-filter:a`, see the `-filter` option.

## 5.8 Advanced Audio options# TOC

`-atag fourcc/tag (output)`

Force audio tag/fourcc. This is an alias for `-tag:a`.

`-absf bitstream_filter`

Deprecated, see `-bsf`

`-guess_layout_max channels (input,per-stream)`

If some input channel layout is not known, try to guess only if it corresponds to at most the specified number of channels. For example, 2 tells to `ffmpeg` to recognize 1 channel as mono and 2 channels as stereo but not 6 channels as 5.1. The default is to always try to guess. Use 0 to disable all guessing.

## 5.9 Subtitle options# TOC

`-scodec codec (input/output)`

Set the subtitle codec. This is an alias for `-codec:s`.

`-sn (output)`

Disable subtitle recording.

`-sbsf bitstream_filter`

Deprecated, see `-bsf`

## 5.10 Advanced Subtitle options# TOC

`-fix_sub_duration`

Fix subtitles durations. For each subtitle, wait for the next packet in the same stream and adjust the duration of the first to avoid overlap. This is necessary with some subtitles codecs, especially DVB subtitles, because the duration in the original packet is only a rough estimate and the end is actually marked by an empty subtitle frame. Failing to use this option when necessary can result in exaggerated durations or muxing failures due to non-monotonic timestamps.

Note that this option will delay the output of all data until the next subtitle packet is decoded: it may increase memory consumption and latency a lot.

`-canvas_size size`

Set the size of the canvas used to render subtitles.

## 5.11 Advanced options# TOC

`-map`

`[ - ]input_file_id[:stream_specifier][,sync_file_id[:stream_specifier]] | [linklabel] (output)`

Designate one or more input streams as a source for the output file. Each input stream is identified by the input file index *input\_file\_id* and the input stream index *input\_stream\_id* within the input file. Both indices start at 0. If specified, *sync\_file\_id:stream\_specifier* sets which input stream is used as a presentation sync reference.

The first `-map` option on the command line specifies the source for output stream 0, the second `-map` option specifies the source for output stream 1, etc.

A `-` character before the stream identifier creates a "negative" mapping. It disables matching streams from already created mappings.

An alternative *[linklabel]* form will map outputs from complex filter graphs (see the `-filter_complex` option) to the output file. *linklabel* must correspond to a defined output link label in the graph.

For example, to map ALL streams from the first input file to output

```
ffmpeg -i INPUT -map 0 output
```

For example, if you have two audio streams in the first input file, these streams are identified by "0:0" and "0:1". You can use `-map` to select which streams to place in an output file. For example:

```
ffmpeg -i INPUT -map 0:1 out.wav
```

will map the input stream in `INPUT` identified by "0:1" to the (single) output stream in `out.wav`.

For example, to select the stream with index 2 from input file `a.mov` (specified by the identifier "0:2"), and stream with index 6 from input `b.mov` (specified by the identifier "1:6"), and copy them to the output file `out.mov`:

```
ffmpeg -i a.mov -i b.mov -c copy -map 0:2 -map 1:6 out.mov
```

To select all video and the third audio stream from an input file:

```
ffmpeg -i INPUT -map 0:v -map 0:a:2 OUTPUT
```

To map all the streams except the second audio, use negative mappings

```
ffmpeg -i INPUT -map 0 -map -0:a:1 OUTPUT
```

To pick the English audio stream:

```
ffmpeg -i INPUT -map 0:m:language:eng OUTPUT
```

Note that using this option disables the default mappings for this output file.

`-ignore_unknown`

Ignore input streams with unknown type instead of failing if copying such streams is attempted.

`-copy_unknown`

Allow input streams with unknown type to be copied instead of failing if copying such streams is attempted.

`-map_channel`

`[input_file_id.stream_specifier.channel_id|-1][:output_file_id.stream_specifier]`

Map an audio channel from a given input to an output. If `output_file_id.stream_specifier` is not set, the audio channel will be mapped on all the audio streams.

Using "-1" instead of `input_file_id.stream_specifier.channel_id` will map a muted channel.

For example, assuming `INPUT` is a stereo audio file, you can switch the two audio channels with the following command:

```
ffmpeg -i INPUT -map_channel 0.0.1 -map_channel 0.0.0 OUTPUT
```

If you want to mute the first channel and keep the second:

```
ffmpeg -i INPUT -map_channel -1 -map_channel 0.0.1 OUTPUT
```

The order of the "-map\_channel" option specifies the order of the channels in the output stream. The output channel layout is guessed from the number of channels mapped (mono if one "-map\_channel", stereo if two, etc.). Using "-ac" in combination of "-map\_channel" makes the channel gain levels to be updated if input and output channel layouts don't match (for instance two "-map\_channel" options and "-ac 6").

You can also extract each channel of an input to specific outputs; the following command extracts two channels of the *INPUT* audio stream (file 0, stream 0) to the respective *OUTPUT\_CH0* and *OUTPUT\_CH1* outputs:

```
ffmpeg -i INPUT -map_channel 0.0.0 OUTPUT_CH0 -map_channel 0.0.1 OUTPUT_CH1
```

The following example splits the channels of a stereo input into two separate streams, which are put into the same output file:

```
ffmpeg -i stereo.wav -map 0:0 -map 0:0 -map_channel 0.0.0:0.0 -map_channel 0.0.1:0.1 -y out.ogg
```

Note that currently each output stream can only contain channels from a single input stream; you can't for example use "-map\_channel" to pick multiple input audio channels contained in different streams (from the same or different files) and merge them into a single output stream. It is therefore not currently possible, for example, to turn two separate mono streams into a single stereo stream. However splitting a stereo stream into two single channel mono streams is possible.

If you need this feature, a possible workaround is to use the *amerge* filter. For example, if you need to merge a media (here *input.mkv*) with 2 mono audio streams into one single stereo channel audio stream (and keep the video stream), you can use the following command:

```
ffmpeg -i input.mkv -filter_complex "[0:1] [0:2] amerge" -c:a pcm_s16le -c:v copy output.mkv
```

```
-map_metadata[:metadata_spec_out] infile[:metadata_spec_in]  
(output,per-metadata)
```

Set metadata information of the next output file from *infile*. Note that those are file indices (zero-based), not filenames. Optional *metadata\_spec\_in/out* parameters specify, which metadata to copy. A metadata specifier can have the following forms:

*g*

global metadata, i.e. metadata that applies to the whole file

*s[:stream\_spec]*



per-stream metadata. *stream\_spec* is a stream specifier as described in the Stream specifiers chapter. In an input metadata specifier, the first matching stream is copied from. In an output metadata specifier, all matching streams are copied to.

*c:chapter\_index*

per-chapter metadata. *chapter\_index* is the zero-based chapter index.

*p:program\_index*

per-program metadata. *program\_index* is the zero-based program index.

If metadata specifier is omitted, it defaults to global.

By default, global metadata is copied from the first input file, per-stream and per-chapter metadata is copied along with streams/chapters. These default mappings are disabled by creating any mapping of the relevant type. A negative file index can be used to create a dummy mapping that just disables automatic copying.

For example to copy metadata from the first stream of the input file to global metadata of the output file:

```
ffmpeg -i in.ogg -map_metadata 0:s:0 out.mp3
```

To do the reverse, i.e. copy global metadata to all audio streams:

```
ffmpeg -i in.mkv -map_metadata:s:a 0:g out.mkv
```

Note that simple 0 would work as well in this example, since global metadata is assumed by default.

`-map_chapters input_file_index (output)`

Copy chapters from input file with index *input\_file\_index* to the next output file. If no chapter mapping is specified, then chapters are copied from the first input file with at least one chapter. Use a negative file index to disable any chapter copying.

`-benchmark (global)`

Show benchmarking information at the end of an encode. Shows CPU time used and maximum memory consumption. Maximum memory consumption is not supported on all systems, it will usually display as 0 if not supported.

`-benchmark_all (global)`

Show benchmarking information during the encode. Shows CPU time used in various steps (audio/video encode/decode).

`-timelimit duration (global)`

Exit after ffmpeg has been running for *duration* seconds.

`-dump (global)`

Dump each input packet to stderr.

`-hex (global)`

When dumping packets, also dump the payload.

`-re (input)`

Read input at native frame rate. Mainly used to simulate a grab device. or live input stream (e.g. when reading from a file). Should not be used with actual grab devices or live input streams (where it can cause packet loss). By default `ffmpeg` attempts to read the input(s) as fast as possible. This option will slow down the reading of the input(s) to the native frame rate of the input(s). It is useful for real-time output (e.g. live streaming).

`-loop_input`

Loop over the input stream. Currently it works only for image streams. This option is used for automatic FFserver testing. This option is deprecated, use `-loop 1`.

`-loop_output number_of_times`

Repeatedly loop output for formats that support looping such as animated GIF (0 will loop the output infinitely). This option is deprecated, use `-loop`.

`-vsync parameter`

Video sync method. For compatibility reasons old values can be specified as numbers. Newly added values will have to be specified as strings always.

0, `passthrough`

Each frame is passed with its timestamp from the demuxer to the muxer.

1, `cfr`

Frames will be duplicated and dropped to achieve exactly the requested constant frame rate.

2, `vfr`

Frames are passed through with their timestamp or dropped so as to prevent 2 frames from having the same timestamp.

`drop`

As passthrough but destroys all timestamps, making the muxer generate fresh timestamps based on frame-rate.

`-1, auto`

Chooses between 1 and 2 depending on muxer capabilities. This is the default method.

Note that the timestamps may be further modified by the muxer, after this. For example, in the case that the format option `avoid_negative_ts` is enabled.

With `-map` you can select from which stream the timestamps should be taken. You can leave either video or audio unchanged and sync the remaining stream(s) to the unchanged one.

`-frame_drop_threshold parameter`

Frame drop threshold, which specifies how much behind video frames can be before they are dropped. In frame rate units, so 1.0 is one frame. The default is -1.1. One possible usecase is to avoid framedrops in case of noisy timestamps or to increase frame drop precision in case of exact timestamps.

`-async samples_per_second`

Audio sync method. "Stretches/squeezes" the audio stream to match the timestamps, the parameter is the maximum samples per second by which the audio is changed. `-async 1` is a special case where only the start of the audio stream is corrected without any later correction.

Note that the timestamps may be further modified by the muxer, after this. For example, in the case that the format option `avoid_negative_ts` is enabled.

This option has been deprecated. Use the `aresample` audio filter instead.

`-copyts`

Do not process input timestamps, but keep their values without trying to sanitize them. In particular, do not remove the initial start time offset value.

Note that, depending on the `vsync` option or on specific muxer processing (e.g. in case the format option `avoid_negative_ts` is enabled) the output timestamps may mismatch with the input timestamps even when this option is selected.

`-start_at_zero`

When used with `copyts`, shift input timestamps so they start at zero.

This means that using e.g. `-ss 50` will make output timestamps start at 50 seconds, regardless of what timestamp the input file started at.

`-copytb mode`

Specify how to set the encoder timebase when stream copying. *mode* is an integer numeric value, and can assume one of the following values:

1

Use the demuxer timebase.

The time base is copied to the output encoder from the corresponding input demuxer. This is sometimes required to avoid non monotonically increasing timestamps when copying video streams with variable frame rate.

0

Use the decoder timebase.

The time base is copied to the output encoder from the corresponding input decoder.

-1

Try to make the choice automatically, in order to generate a sane output.

Default value is -1.

`-shortest (output)`

Finish encoding when the shortest input stream ends.

`-dts_delta_threshold`

Timestamp discontinuity delta threshold.

`-muxdelay seconds (input)`

Set the maximum demux-decode delay.

`-muxpreload seconds (input)`

Set the initial demux-decode delay.

`-streamid output-stream-index:new-value (output)`

Assign a new stream-id value to an output stream. This option should be specified prior to the output filename to which it applies. For the situation where multiple output files exist, a streamid may be reassigned to a different value.

For example, to set the stream 0 PID to 33 and the stream 1 PID to 36 for an output mpegts file:

```
ffmpeg -i infile -streamid 0:33 -streamid 1:36 out.ts
```

`-bsf[:stream_specifier] bitstream_filters (output,per-stream)`

Set bitstream filters for matching streams. *bitstream\_filters* is a comma-separated list of bitstream filters. Use the `-bsfs` option to get the list of bitstream filters.

```
ffmpeg -i h264.mp4 -c:v copy -bsf:v h264_mp4toannexb -an out.h264
```

```
ffmpeg -i file.mov -an -vn -bsf:s mov2textsub -c:s copy -f rawvideo sub.txt
```

`-tag[:stream_specifier] codec_tag (input/output,per-stream)`

Force a tag/fourcc for matching streams.

`-timecode hh:mm:ssSEPff`

Specify Timecode for writing. *SEP* is ':' for non drop timecode and ';' (or '.') for drop.

```
ffmpeg -i input.mpg -timecode 01:02:03.04 -r 30000/1001 -s ntsc output.mpg
```

`-filter_complex filtergraph (global)`

Define a complex filtergraph, i.e. one with arbitrary number of inputs and/or outputs. For simple graphs – those with one input and one output of the same type – see the `-filter` options. *filtergraph* is a description of the filtergraph, as described in the “Filtergraph syntax” section of the `ffmpeg-filters` manual.

Input link labels must refer to input streams using the `[file_index:stream_specifier]` syntax (i.e. the same as `-map` uses). If *stream\_specifier* matches multiple streams, the first one will be used. An unlabeled input will be connected to the first unused input stream of the matching type.

Output link labels are referred to with `-map`. Unlabeled outputs are added to the first output file.

Note that with this option it is possible to use only `lavfi` sources without normal input files.

For example, to overlay an image over video

```
ffmpeg -i video.mkv -i image.png -filter_complex '[0:v][1:v]overlay[out]' -map '[out]' out.mkv
```

Here `[0:v]` refers to the first video stream in the first input file, which is linked to the first (main) input of the overlay filter. Similarly the first video stream in the second input is linked to the second (overlay) input of overlay.

Assuming there is only one video stream in each input file, we can omit input labels, so the above is equivalent to

```
ffmpeg -i video.mkv -i image.png -filter_complex 'overlay[out]' -map '[out]' out.mkv
```

Furthermore we can omit the output label and the single output from the filter graph will be added to the output file automatically, so we can simply write

```
ffmpeg -i video.mkv -i image.png -filter_complex 'overlay' out.mkv
```

To generate 5 seconds of pure red video using lavfi color source:

```
ffmpeg -filter_complex 'color=c=red' -t 5 out.mkv
```

`-lavfi filtergraph (global)`

Define a complex filtergraph, i.e. one with arbitrary number of inputs and/or outputs. Equivalent to `-filter_complex`.

`-filter_complex_script filename (global)`

This option is similar to `-filter_complex`, the only difference is that its argument is the name of the file from which a complex filtergraph description is to be read.

`-accurate_seek (input)`

This option enables or disables accurate seeking in input files with the `-ss` option. It is enabled by default, so seeking is accurate when transcoding. Use `-noaccurate_seek` to disable it, which may be useful e.g. when copying some streams and transcoding the others.

`-seek_timestamp (input)`

This option enables or disables seeking by timestamp in input files with the `-ss` option. It is disabled by default. If enabled, the argument to the `-ss` option is considered an actual timestamp, and is not offset by the start time of the file. This matters only for files which do not start from timestamp 0, such as transport streams.

`-thread_queue_size size (input)`

This option sets the maximum number of queued packets when reading from the file or device. With low latency / high rate live streams, packets may be discarded if they are not read in a timely manner; raising this value can avoid it.

`-override_ffserver (global)`

Overrides the input specifications from `ffserver`. Using this option you can map any input stream to `ffserver` and control many aspects of the encoding from `ffmpeg`. Without this option `ffmpeg` will transmit to `ffserver` what is requested by `ffserver`.

The option is intended for cases where features are needed that cannot be specified to `ffserver` but can be to `ffmpeg`.

`-sdp_file file (global)`

Print sdp information to *file*. This allows dumping sdp information when at least one output isn't an rtp stream.

`-discard (input)`

Allows discarding specific streams or frames of streams at the demuxer. Not all demuxers support this.

`none`

Discard no frame.

`default`

Default, which discards no frames.

`noref`

Discard all non-reference frames.

`bidir`

Discard all bidirectional frames.

`nokey`

Discard all frames excepts keyframes.

`all`

Discard all frames.

`-xerror (global)`

Stop and exit on error

As a special exception, you can use a bitmap subtitle stream as input: it will be converted into a video with the same size as the largest video in the file, or 720x576 if no video is present. Note that this is an experimental and temporary solution. It will be removed once libavfilter has proper support for subtitles.

For example, to hardcode subtitles on top of a DVB-T recording stored in MPEG-TS format, delaying the subtitles by 1 second:

```
ffmpeg -i input.ts -filter_complex \
'[#0x2ef] setpts=PTS+1/TB [sub] ; [#0x2d0] [sub] overlay' \
-sn -map '#0x2dc' output.mkv
```

(0x2d0, 0x2dc and 0x2ef are the MPEG-TS PIDs of respectively the video, audio and subtitles streams; 0:0, 0:3 and 0:7 would have worked too)

## 5.12 Preset files# TOC

A preset file contains a sequence of *option=value* pairs, one for each line, specifying a sequence of options which would be awkward to specify on the command line. Lines starting with the hash ('#') character are ignored and are used to provide comments. Check the `presets` directory in the FFmpeg source tree for examples.

There are two types of preset files: `ffpreset` and `avpreset` files.

### 5.12.1 ffpreset files# TOC

`ffpreset` files are specified with the `vpre`, `apre`, `spre`, and `fpre` options. The `fpre` option takes the filename of the preset instead of a preset name as input and can be used for any kind of codec. For the `vpre`, `apre`, and `spre` options, the options specified in a preset file are applied to the currently selected codec of the same type as the preset option.

The argument passed to the `vpre`, `apre`, and `spre` preset options identifies the preset file to use according to the following rules:

First `ffmpeg` searches for a file named *arg*.`ffpreset` in the directories `$FFMPEG_DATADIR` (if set), and `$HOME/.ffmpeg`, and in the `datadir` defined at configuration time (usually `PREFIX/share/ffmpeg`) or in a `ffpresets` folder along the executable on win32, in that order. For example, if the argument is `libvpx-1080p`, it will search for the file `libvpx-1080p.ffpreset`.

If no such file is found, then `ffmpeg` will search for a file named *codec\_name-arg*.`ffpreset` in the above-mentioned directories, where *codec\_name* is the name of the codec to which the preset file options will be applied. For example, if you select the video codec with `-vcodec libvpx` and use `-vpre 1080p`, then it will search for the file `libvpx-1080p.ffpreset`.

### 5.12.2 avpreset files# TOC

`avpreset` files are specified with the `pre` option. They work similar to `ffpreset` files, but they only allow encoder-specific options. Therefore, an *option=value* pair specifying an encoder cannot be used.

When the `pre` option is specified, `ffmpeg` will look for files with the suffix `.avpreset` in the directories `$AVCONV_DATADIR` (if set), and `$HOME/.avconv`, and in the `datadir` defined at configuration time (usually `PREFIX/share/ffmpeg`), in that order.

First `ffmpeg` searches for a file named *codec\_name-arg*.`avpreset` in the above-mentioned directories, where *codec\_name* is the name of the codec to which the preset file options will be applied. For example, if you select the video codec with `-vcodec libvpx` and use `-pre 1080p`, then it will search for the file `libvpx-1080p.avpreset`.

If no such file is found, then `ffmpeg` will search for a file named *arg*.`avpreset` in the same directories.



## 6 Examples# TOC

### 6.1 Video and Audio grabbing# TOC

If you specify the input format and device then ffmpeg can grab video and audio directly.

```
ffmpeg -f oss -i /dev/dsp -f video4linux2 -i /dev/video0 /tmp/out.mpg
```

Or with an ALSA audio source (mono input, card id 1) instead of OSS:

```
ffmpeg -f alsa -ac 1 -i hw:1 -f video4linux2 -i /dev/video0 /tmp/out.mpg
```

Note that you must activate the right video source and channel before launching ffmpeg with any TV viewer such as xawtv by Gerd Knorr. You also have to set the audio recording levels correctly with a standard mixer.

### 6.2 X11 grabbing# TOC

Grab the X11 display with ffmpeg via

```
ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0 /tmp/out.mpg
```

0.0 is display.screen number of your X11 server, same as the DISPLAY environment variable.

```
ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0+10,20 /tmp/out.mpg
```

0.0 is display.screen number of your X11 server, same as the DISPLAY environment variable. 10 is the x-offset and 20 the y-offset for the grabbing.

### 6.3 Video and Audio file format conversion# TOC

Any supported file format and protocol can serve as input to ffmpeg:

Examples:

- You can use YUV files as input:

```
ffmpeg -i /tmp/test%d.Y /tmp/out.mpg
```

It will use the files:

```
/tmp/test0.Y, /tmp/test0.U, /tmp/test0.V,  
/tmp/test1.Y, /tmp/test1.U, /tmp/test1.V, etc...
```

The Y files use twice the resolution of the U and V files. They are raw files, without header. They can be generated by all decent video decoders. You must specify the size of the image with the `-s` option if ffmpeg cannot guess it.

- You can input from a raw YUV420P file:

```
ffmpeg -i /tmp/test.yuv /tmp/out.avi
```

test.yuv is a file containing raw YUV planar data. Each frame is composed of the Y plane followed by the U and V planes at half vertical and horizontal resolution.

- You can output to a raw YUV420P file:

```
ffmpeg -i mydivx.avi hugefile.yuv
```

- You can set several input files and output files:

```
ffmpeg -i /tmp/a.wav -s 640x480 -i /tmp/a.yuv /tmp/a.mpg
```

Converts the audio file a.wav and the raw YUV video file a.yuv to MPEG file a.mpg.

- You can also do audio and video conversions at the same time:

```
ffmpeg -i /tmp/a.wav -ar 22050 /tmp/a.mp2
```

Converts a.wav to MPEG audio at 22050 Hz sample rate.

- You can encode to several formats at the same time and define a mapping from input stream to output streams:

```
ffmpeg -i /tmp/a.wav -map 0:a -b:a 64k /tmp/a.mp2 -map 0:a -b:a 128k /tmp/b.mp2
```

Converts a.wav to a.mp2 at 64 kbits and to b.mp2 at 128 kbits. '-map file:index' specifies which input stream is used for each output stream, in the order of the definition of output streams.

- You can transcode decrypted VOBs:

```
ffmpeg -i snatch_1.vob -f avi -c:v mpeg4 -b:v 800k -g 300 -bf 2 -c:a libmp3lame -b:a 128k snatch.avi
```

This is a typical DVD ripping example; the input is a VOB file, the output an AVI file with MPEG-4 video and MP3 audio. Note that in this command we use B-frames so the MPEG-4 stream is DivX5 compatible, and GOP size is 300 which means one intra frame every 10 seconds for 29.97fps input video. Furthermore, the audio stream is MP3-encoded so you need to enable LAME support by passing --enable-libmp3lame to configure. The mapping is particularly useful for DVD transcoding to get the desired audio language.

NOTE: To see the supported input formats, use `ffmpeg -formats`.

- You can extract images from a video, or create a video from many images:

For extracting images from a video:

```
ffmpeg -i foo.avi -r 1 -s WxH -f image2 foo-%03d.jpeg
```

This will extract one video frame per second from the video and will output them in files named `foo-001.jpeg`, `foo-002.jpeg`, etc. Images will be rescaled to fit the new WxH values.

If you want to extract just a limited number of frames, you can use the above command in combination with the `-vframes` or `-t` option, or in combination with `-ss` to start extracting from a certain point in time.

For creating a video from many images:

```
ffmpeg -f image2 -framerate 12 -i foo-%03d.jpeg -s WxH foo.avi
```

The syntax `foo-%03d.jpeg` specifies to use a decimal number composed of three digits padded with zeroes to express the sequence number. It is the same syntax supported by the C `printf` function, but only formats accepting a normal integer are suitable.

When importing an image sequence, `-i` also supports expanding shell-like wildcard patterns (globbing) internally, by selecting the image2-specific `-pattern_type glob` option.

For example, for creating a video from filenames matching the glob pattern `foo-*.jpeg`:

```
ffmpeg -f image2 -pattern_type glob -framerate 12 -i 'foo-*.jpeg' -s WxH foo.avi
```

- You can put many streams of the same type in the output:

```
ffmpeg -i test1.avi -i test2.avi -map 1:1 -map 1:0 -map 0:1 -map 0:0 -c copy -y test12.nut
```

The resulting output file `test12.nut` will contain the first four streams from the input files in reverse order.

- To force CBR video output:

```
ffmpeg -i myfile.avi -b 4000k -minrate 4000k -maxrate 4000k -bufsize 1835k out.m2v
```

- The four options `lmin`, `lmax`, `mblmin` and `mblmax` use 'lambda' units, but you may use the `QP2LAMBDA` constant to easily convert from 'q' units:

```
ffmpeg -i src.ext -lmax 21*QP2LAMBDA dst.ext
```

## 7 Syntax# TOC

This section documents the syntax and formats employed by the FFmpeg libraries and tools.

### 7.1 Quoting and escaping# TOC

FFmpeg adopts the following quoting and escaping mechanism, unless explicitly specified. The following rules are applied:

- ‘’ and ‘\’ are special characters (respectively used for quoting and escaping). In addition to them, there might be other special characters depending on the specific syntax where the escaping and quoting are employed.
- A special character is escaped by prefixing it with a ‘\’.
- All characters enclosed between ‘ ’’ are included literally in the parsed string. The quote character ‘’ itself cannot be quoted, so you may need to close the quote and escape it.
- Leading and trailing whitespaces, unless escaped or quoted, are removed from the parsed string.

Note that you may need to add a second level of escaping when using the command line or a script, which depends on the syntax of the adopted shell language.

The function `av_get_token` defined in `libavutil/avstring.h` can be used to parse a token quoted or escaped according to the rules defined above.

The tool `tools/ffescape` in the FFmpeg source tree can be used to automatically quote or escape a string in a script.

### 7.1.1 Examples# TOC

- Escape the string `Crime d'Amour` containing the ‘ ’ special character:

```
Crime d\'Amour
```

- The string above contains a quote, so the ‘ ’ needs to be escaped when quoting it:

```
'Crime d\'\'Amour'
```

- Include leading or trailing whitespaces using quoting:

```
' this string starts and ends with whitespaces '
```

- Escaping and quoting can be mixed together:

```
' The string \'string\' is a string '
```

- To include a literal ‘\’ you can use either escaping or quoting:

```
'c:\foo' can be written as c:\\foo
```

## 7.2 Date# TOC

The accepted syntax is:

```
[ (YYYY-MM-DD|YYYYMMDD) [T|t| ] ( (HH:MM:SS[.m...]) | (HHMMSS[.m...]) ) [Z]
now
```

If the value is "now" it takes the current time.

Time is local time unless Z is appended, in which case it is interpreted as UTC. If the year-month-day part is not specified it takes the current year-month-day.

## 7.3 Time duration# TOC

There are two accepted syntaxes for expressing time duration.

`[ - ][HH:]MM:SS[.m...]`

*HH* expresses the number of hours, *MM* the number of minutes for a maximum of 2 digits, and *SS* the number of seconds for a maximum of 2 digits. The *m* at the end expresses decimal value for *SS*.

or

`[ - ]S+[.m...]`

*S* expresses the number of seconds, with the optional decimal part *m*.

In both expressions, the optional ‘-’ indicates negative duration.

### 7.3.1 Examples# TOC

The following examples are all valid time duration:

‘55’

55 seconds

‘12:03:45’

12 hours, 03 minutes and 45 seconds

‘23.189’

23.189 seconds

## 7.4 Video size# TOC

Specify the size of the sourced video, it may be a string of the form *widthxheight*, or the name of a size abbreviation.

The following abbreviations are recognized:

‘ntsc’

720x480

‘pal’

720x576

‘qntsc’

352x240

‘qpal’

352x288

‘sntsc’

640x480

‘spal’

768x576

‘film’

352x240

‘ntsc-film’

352x240

‘sqcif’

128x96

‘qcif’

176x144

‘cif’

352x288

‘4cif’

704x576

‘16cif’

1408x1152

‘qqvga’

160x120

‘qvga’

320x240

‘vga’

640x480

‘svga’

800x600

‘xga’

1024x768

‘uxga’

1600x1200

‘qxga’

2048x1536

‘sxga’

1280x1024

‘qsxga’

2560x2048

‘hsxga’

5120x4096

‘wvga’

852x480

‘wxga’

1366x768

‘wsxga’

1600x1024

‘wuxga’

1920x1200

‘woxga’

2560x1600

‘wqsxga’

3200x2048

‘wquxga’

3840x2400

‘whsxga’

6400x4096

‘whuxga’

7680x4800

‘cga’

320x200

‘ega’

640x350

‘hd480’

852x480

‘hd720’

1280x720

‘hd1080’

1920x1080

‘2k’



2048x1080

'2kflat'

1998x1080

'2kscope'

2048x858

'4k'

4096x2160

'4kflat'

3996x2160

'4kscope'

4096x1716

'nhd'

640x360

'hqvga'

240x160

'wqvga'

400x240

'fwqvga'

432x240

'hvga'

480x320

'qhd'

960x540

'2kdc1'

2048x1080

‘4kdc1’

4096x2160

‘uhd2160’

3840x2160

‘uhd4320’

7680x4320

## 7.5 Video rate# TOC

Specify the frame rate of a video, expressed as the number of frames generated per second. It has to be a string in the format *frame\_rate\_num/frame\_rate\_den*, an integer number, a float number or a valid video frame rate abbreviation.

The following abbreviations are recognized:

‘ntsc’

30000/1001

‘pal’

25/1

‘qntsc’

30000/1001

‘qp1’

25/1

‘sntsc’

30000/1001

‘sp1’

25/1

‘film’

24/1

`'ntsc-film'`

24000/1001

## 7.6 Ratio# TOC

A ratio can be expressed as an expression, or in the form *numerator:denominator*.

Note that a ratio with infinite (1/0) or negative value is considered valid, so you should check on the returned value if you want to exclude those values.

The undefined value can be expressed using the "0:0" string.

## 7.7 Color# TOC

It can be the name of a color as defined below (case insensitive match) or a `[ 0x | # ]RRGGBB[ AA ]` sequence, possibly followed by `@` and a string representing the alpha component.

The alpha component may be a string composed by "0x" followed by an hexadecimal number or a decimal number between 0.0 and 1.0, which represents the opacity value (`'0x00'` or `'0.0'` means completely transparent, `'0xff'` or `'1.0'` completely opaque). If the alpha component is not specified then `'0xff'` is assumed.

The string `'random'` will result in a random color.

The following names of colors are recognized:

`'AliceBlue'`

`0xF0F8FF`

`'AntiqueWhite'`

`0xFAEBD7`

`'Aqua'`

`0x00FFFF`

`'Aquamarine'`

`0x7FFFD4`

`'Azure'`

0xF0FFFF

‘Beige’

0xF5F5DC

‘Bisque’

0xFFE4C4

‘Black’

0x000000

‘BlanchedAlmond’

0xFFEBCD

‘Blue’

0x0000FF

‘BlueViolet’

0x8A2BE2

‘Brown’

0xA52A2A

‘BurlyWood’

0xDEB887

‘CadetBlue’

0x5F9EA0

‘Chartreuse’

0x7FFF00

‘Chocolate’

0xD2691E

‘Coral’

0xFF7F50

‘CornflowerBlue’

0x6495ED

‘Cornsilk’

0xFFFF8DC

‘Crimson’

0xDC143C

‘Cyan’

0x00FFFF

‘DarkBlue’

0x00008B

‘DarkCyan’

0x008B8B

‘DarkGoldenRod’

0xB8860B

‘DarkGray’

0xA9A9A9

‘DarkGreen’

0x006400

‘DarkKhaki’

0xBDB76B

‘DarkMagenta’

0x8B008B

‘DarkOliveGreen’

0x556B2F

‘Darkorange’

0xFF8C00

‘DarkOrchid’

0x9932CC

‘DarkRed’

0x8B0000

‘DarkSalmon’

0xE9967A

‘DarkSeaGreen’

0x8FBC8F

‘DarkSlateBlue’

0x483D8B

‘DarkSlateGray’

0x2F4F4F

‘DarkTurquoise’

0x00CED1

‘DarkViolet’

0x9400D3

‘DeepPink’

0xFF1493

‘DeepSkyBlue’

0x00BFFF

‘DimGray’

0x696969

'DodgerBlue'

0x1E90FF

'FireBrick'

0xB22222

'FloralWhite'

0xFFFAF0

'ForestGreen'

0x228B22

'Fuchsia'

0xFF00FF

'Gainsboro'

0xDCDCDC

'GhostWhite'

0xF8F8FF

'Gold'

0xFFD700

'GoldenRod'

0xDAA520

'Gray'

0x808080

'Green'

0x008000

'GreenYellow'

0xADFF2F

‘HoneyDew’

0xF0FFF0

‘HotPink’

0xFF69B4

‘IndianRed’

0xCD5C5C

‘Indigo’

0x4B0082

‘Ivory’

0xFFFFF0

‘Khaki’

0xF0E68C

‘Lavender’

0xE6E6FA

‘LavenderBlush’

0xFFF0F5

‘LawnGreen’

0x7CFC00

‘LemonChiffon’

0xFFFACD

‘LightBlue’

0xADD8E6

‘LightCoral’



0xF08080

'LightCyan'

0xE0FFFF

'LightGoldenRodYellow'

0xFAFAD2

'LightGreen'

0x90EE90

'LightGrey'

0xD3D3D3

'LightPink'

0xFFB6C1

'LightSalmon'

0xFFA07A

'LightSeaGreen'

0x20B2AA

'LightSkyBlue'

0x87CEFA

'LightSlateGray'

0x778899

'LightSteelBlue'

0xB0C4DE

'LightYellow'

0xFFFFE0

'Lime'

0x00FF00

'LimeGreen'

0x32CD32

'Linen'

0xFAF0E6

'Magenta'

0xFF00FF

'Maroon'

0x800000

'MediumAquaMarine'

0x66CDAA

'MediumBlue'

0x0000CD

'MediumOrchid'

0xBA55D3

'MediumPurple'

0x9370D8

'MediumSeaGreen'

0x3CB371

'MediumSlateBlue'

0x7B68EE

'MediumSpringGreen'

0x00FA9A

'MediumTurquoise'

0x48D1CC

‘MediumVioletRed’

0xC71585

‘MidnightBlue’

0x191970

‘MintCream’

0xF5FFFA

‘MistyRose’

0xFFE4E1

‘Moccasin’

0xFFE4B5

‘NavajoWhite’

0xFFDEAD

‘Navy’

0x000080

‘OldLace’

0xFDF5E6

‘Olive’

0x808000

‘OliveDrab’

0x6B8E23

‘Orange’

0xFFA500

‘OrangeRed’

0xFF4500

‘Orchid’

0xDA70D6

‘PaleGoldenRod’

0xEEE8AA

‘PaleGreen’

0x98FB98

‘PaleTurquoise’

0xAFEEEE

‘PaleVioletRed’

0xD87093

‘PapayaWhip’

0xFFEFD5

‘PeachPuff’

0xFFDAB9

‘Peru’

0xCD853F

‘Pink’

0xFFC0CB

‘Plum’

0xDDA0DD

‘PowderBlue’

0xB0E0E6

‘Purple’

0x800080

‘Red’

0xFF0000

‘RosyBrown’

0xBC8F8F

‘RoyalBlue’

0x4169E1

‘SaddleBrown’

0x8B4513

‘Salmon’

0xFA8072

‘SandyBrown’

0xF4A460

‘SeaGreen’

0x2E8B57

‘SeaShell’

0xFFF5EE

‘Sienna’

0xA0522D

‘Silver’

0xC0C0C0

‘SkyBlue’

0x87CEEB

‘SlateBlue’

0x6A5ACD

‘SlateGray’

0x708090

‘Snow’

0xFFFFFA

‘SpringGreen’

0x00FF7F

‘SteelBlue’

0x4682B4

‘Tan’

0xD2B48C

‘Teal’

0x008080

‘Thistle’

0xD8BFD8

‘Tomato’

0xFF6347

‘Turquoise’

0x40E0D0

‘Violet’

0xEE82EE

‘Wheat’

0xF5DEB3

‘White’

0xFFFFFFFF

‘WhiteSmoke’

0xF5F5F5

‘Yellow’

0xFFFF00

‘YellowGreen’

0x9ACD32

## 7.8 Channel Layout# TOC

A channel layout specifies the spatial disposition of the channels in a multi-channel audio stream. To specify a channel layout, FFmpeg makes use of a special syntax.

Individual channels are identified by an id, as given by the table below:

‘FL’

front left

‘FR’

front right

‘FC’

front center

‘LFE’

low frequency

‘BL’

back left

‘BR’

back right

‘FLC’

front left-of-center

‘FRC’

front right-of-center

‘BC’

back center

‘SL’

side left

‘SR’

side right

‘TC’

top center

‘TFL’

top front left

‘TFC’

top front center

‘TFR’

top front right

‘TBL’

top back left

‘TBC’

top back center

‘TBR’

top back right

‘DL’

downmix left

‘DR’



downmix right

‘WL’

wide left

‘WR’

wide right

‘SDL’

surround direct left

‘SDR’

surround direct right

‘LFE2’

low frequency 2

Standard channel layout compositions can be specified by using the following identifiers:

‘mono’

FC

‘stereo’

FL+FR

‘2.1’

FL+FR+LFE

‘3.0’

FL+FR+FC

‘3.0(back)’

FL+FR+BC

‘4.0’

FL+FR+FC+BC

‘quad’

FL+FR+BL+BR

‘quad(side)’

FL+FR+SL+SR

‘3.1’

FL+FR+FC+LFE

‘5.0’

FL+FR+FC+BL+BR

‘5.0(side)’

FL+FR+FC+SL+SR

‘4.1’

FL+FR+FC+LFE+BC

‘5.1’

FL+FR+FC+LFE+BL+BR

‘5.1(side)’

FL+FR+FC+LFE+SL+SR

‘6.0’

FL+FR+FC+BC+SL+SR

‘6.0(front)’

FL+FR+FLC+FRC+SL+SR

‘hexagonal’

FL+FR+FC+BL+BR+BC

‘6.1’

FL+FR+FC+LFE+BC+SL+SR

‘6.1’

FL+FR+FC+LFE+BL+BR+BC

'6.1(front)'

FL+FR+LFE+FLC+FRC+SL+SR

'7.0'

FL+FR+FC+BL+BR+SL+SR

'7.0(front)'

FL+FR+FC+FLC+FRC+SL+SR

'7.1'

FL+FR+FC+LFE+BL+BR+SL+SR

'7.1(wide)'

FL+FR+FC+LFE+BL+BR+FLC+FRC

'7.1(wide-side)'

FL+FR+FC+LFE+FLC+FRC+SL+SR

'octagonal'

FL+FR+FC+BL+BR+BC+SL+SR

'downmix'

DL+DR

A custom channel layout can be specified as a sequence of terms, separated by '+' or '|'. Each term can be:

- the name of a standard channel layout (e.g. 'mono', 'stereo', '4.0', 'quad', '5.0', etc.)
- the name of a single channel (e.g. 'FL', 'FR', 'FC', 'LFE', etc.)
- a number of channels, in decimal, optionally followed by 'c', yielding the default channel layout for that number of channels (see the function `av_get_default_channel_layout`)
- a channel layout mask, in hexadecimal starting with "0x" (see the `AV_CH_*` macros in `libavutil/channel_layout.h`).

Starting from libavutil version 53 the trailing character "c" to specify a number of channels will be required, while a channel layout mask could also be specified as a decimal number (if and only if not followed by "c").

See also the function `av_get_channel_layout` defined in `libavutil/channel_layout.h`.

## 8 Expression Evaluation# TOC

When evaluating an arithmetic expression, FFmpeg uses an internal formula evaluator, implemented through the `libavutil/eval.h` interface.

An expression may contain unary, binary operators, constants, and functions.

Two expressions *expr1* and *expr2* can be combined to form another expression "*expr1;expr2*". *expr1* and *expr2* are evaluated in turn, and the new expression evaluates to the value of *expr2*.

The following binary operators are available: +, -, \*, /, ^.

The following unary operators are available: +, -.

The following functions are available:

`abs(x)`

Compute absolute value of *x*.

`acos(x)`

Compute arccosine of *x*.

`asin(x)`

Compute arcsine of *x*.

`atan(x)`

Compute arctangent of *x*.

`between(x, min, max)`

Return 1 if *x* is greater than or equal to *min* and lesser than or equal to *max*, 0 otherwise.

`bitand(x, y)`

`bitor(x, y)`

Compute bitwise and/or operation on *x* and *y*.

The results of the evaluation of *x* and *y* are converted to integers before executing the bitwise operation.

Note that both the conversion to integer and the conversion back to floating point can lose precision. Beware of unexpected results for large numbers (usually  $2^{53}$  and larger).

`ceil(expr)`

Round the value of expression *expr* upwards to the nearest integer. For example, "ceil(1.5)" is "2.0".

`clip(x, min, max)`

Return the value of *x* clipped between *min* and *max*.

`cos(x)`

Compute cosine of *x*.

`cosh(x)`

Compute hyperbolic cosine of *x*.

`eq(x, y)`

Return 1 if *x* and *y* are equivalent, 0 otherwise.

`exp(x)`

Compute exponential of *x* (with base *e*, the Euler's number).

`floor(expr)`

Round the value of expression *expr* downwards to the nearest integer. For example, "floor(-1.5)" is "-2.0".

`gauss(x)`

Compute Gauss function of *x*, corresponding to  $\exp(-x^2/2) / \sqrt{2\pi}$ .

`gcd(x, y)`

Return the greatest common divisor of *x* and *y*. If both *x* and *y* are 0 or either or both are less than zero then behavior is undefined.

`gt(x, y)`

Return 1 if *x* is greater than *y*, 0 otherwise.

`gte(x, y)`

Return 1 if *x* is greater than or equal to *y*, 0 otherwise.

`hypot(x, y)`

This function is similar to the C function with the same name; it returns " $\sqrt{x*x + y*y}$ ", the length of the hypotenuse of a right triangle with sides of length  $x$  and  $y$ , or the distance of the point  $(x, y)$  from the origin.

`if(x, y)`

Evaluate  $x$ , and if the result is non-zero return the result of the evaluation of  $y$ , return 0 otherwise.

`if(x, y, z)`

Evaluate  $x$ , and if the result is non-zero return the evaluation result of  $y$ , otherwise the evaluation result of  $z$ .

`ifnot(x, y)`

Evaluate  $x$ , and if the result is zero return the result of the evaluation of  $y$ , return 0 otherwise.

`ifnot(x, y, z)`

Evaluate  $x$ , and if the result is zero return the evaluation result of  $y$ , otherwise the evaluation result of  $z$ .

`isinf(x)`

Return 1.0 if  $x$  is +/-INFINITY, 0.0 otherwise.

`isnan(x)`

Return 1.0 if  $x$  is NAN, 0.0 otherwise.

`ld(var)`

Load the value of the internal variable with number  $var$ , which was previously stored with `st(var, expr)`. The function returns the loaded value.

`log(x)`

Compute natural logarithm of  $x$ .

`lt(x, y)`

Return 1 if  $x$  is lesser than  $y$ , 0 otherwise.

`lte(x, y)`

Return 1 if  $x$  is lesser than or equal to  $y$ , 0 otherwise.

`max(x, y)`

Return the maximum between  $x$  and  $y$ .

`min(x, y)`

Return the maximum between  $x$  and  $y$ .

`mod(x, y)`

Compute the remainder of division of  $x$  by  $y$ .

`not(expr)`

Return 1.0 if  $expr$  is zero, 0.0 otherwise.

`pow(x, y)`

Compute the power of  $x$  elevated  $y$ , it is equivalent to " $(x)^{(y)}$ ".

`print(t)`

`print(t, l)`

Print the value of expression  $t$  with loglevel  $l$ . If  $l$  is not specified then a default log level is used.  
Returns the value of the expression printed.

Prints  $t$  with loglevel  $l$

`random(x)`

Return a pseudo random value between 0.0 and 1.0.  $x$  is the index of the internal variable which will be used to save the seed/state.

`root(expr, max)`

Find an input value for which the function represented by  $expr$  with argument  $ld(0)$  is 0 in the interval  $0..max$ .

The expression in  $expr$  must denote a continuous function or the result is undefined.

$ld(0)$  is used to represent the function input value, which means that the given expression will be evaluated multiple times with various input values that the expression can access through `ld(0)`.  
When the expression evaluates to 0 then the corresponding input value will be returned.

`sin(x)`

Compute sine of  $x$ .

`sinh(x)`

Compute hyperbolic sine of  $x$ .

```
sqrt(expr)
```

Compute the square root of  $expr$ . This is equivalent to " $(expr)^{.5}$ ".

```
squish(x)
```

Compute expression  $1 / (1 + \exp(4 * x))$ .

```
st(var, expr)
```

Store the value of the expression  $expr$  in an internal variable.  $var$  specifies the number of the variable where to store the value, and it is a value ranging from 0 to 9. The function returns the value stored in the internal variable. Note, Variables are currently not shared between expressions.

```
tan(x)
```

Compute tangent of  $x$ .

```
tanh(x)
```

Compute hyperbolic tangent of  $x$ .

```
taylor(expr, x)
```

```
taylor(expr, x, id)
```

Evaluate a Taylor series at  $x$ , given an expression representing the  $ld(id)$ -th derivative of a function at 0.

When the series does not converge the result is undefined.

$ld(id)$  is used to represent the derivative order in  $expr$ , which means that the given expression will be evaluated multiple times with various input values that the expression can access through  $ld(id)$ . If  $id$  is not specified then 0 is assumed.

Note, when you have the derivatives at  $y$  instead of 0, `taylor(expr, x-y)` can be used.

```
time(0)
```

Return the current (wallclock) time in seconds.

```
trunc(expr)
```

Round the value of expression  $expr$  towards zero to the nearest integer. For example, "`trunc(-1.5)`" is "-1.0".

```
while(cond, expr)
```



Evaluate expression *expr* while the expression *cond* is non-zero, and returns the value of the last *expr* evaluation, or NAN if *cond* was always false.

The following constants are available:

PI

area of the unit disc, approximately 3.14

E

exp(1) (Euler's number), approximately 2.718

PHI

golden ratio  $(1+\sqrt{5})/2$ , approximately 1.618

Assuming that an expression is considered "true" if it has a non-zero value, note that:

\* works like AND

+ works like OR

For example the construct:

```
if (A AND B) then C
```

is equivalent to:

```
if (A*B, C)
```

In your C code, you can extend the list of unary and binary functions, and define recognized constants, so that they are available for your expressions.

The evaluator also recognizes the International System unit prefixes. If 'i' is appended after the prefix, binary prefixes are used, which are based on powers of 1024 instead of powers of 1000. The 'B' postfix multiplies the value by 8, and can be appended after a unit prefix or used alone. This allows using for example 'KB', 'MiB', 'G' and 'B' as number postfix.

The list of available International System prefixes follows, with indication of the corresponding powers of 10 and of 2.

Y

$10^{-24}$  /  $2^{-80}$

Z

$$10^{-21} / 2^{-70}$$

a

$$10^{-18} / 2^{-60}$$

f

$$10^{-15} / 2^{-50}$$

p

$$10^{-12} / 2^{-40}$$

n

$$10^{-9} / 2^{-30}$$

u

$$10^{-6} / 2^{-20}$$

m

$$10^{-3} / 2^{-10}$$

c

$$10^{-2}$$

d

$$10^{-1}$$

h

$$10^2$$

k

$$10^3 / 2^{10}$$

K

$$10^3 / 2^{10}$$

M

$10^6 / 2^{20}$

G

$10^9 / 2^{30}$

T

$10^{12} / 2^{40}$

P

$10^{15} / 2^{40}$

E

$10^{18} / 2^{50}$

Z

$10^{21} / 2^{60}$

Y

$10^{24} / 2^{70}$

## 9 OpenCL Options# TOC

When FFmpeg is configured with `--enable-opengl`, it is possible to set the options for the global OpenCL context.

The list of supported options follows:

`build_options`

Set build options used to compile the registered kernels.

See reference "OpenCL Specification Version: 1.2 chapter 5.6.4".

`platform_idx`

Select the index of the platform to run OpenCL code.

The specified index must be one of the indexes in the device list which can be obtained with `ffmpeg -opengl_bench` or `av_opengl_get_device_list()`.

`device_idx`

Select the index of the device used to run OpenCL code.

The specified index must be one of the indexes in the device list which can be obtained with `ffmpeg -opengl_bench` or `av_opengl_get_device_list()`.

## 10 Codec Options# TOC

libavcodec provides some generic global options, which can be set on all the encoders and decoders. In addition each codec may support so-called private options, which are specific for a given codec.

Sometimes, a global option may only affect a specific kind of codec, and may be nonsensical or ignored by another, so you need to be aware of the meaning of the specified options. Also some options are meant only for decoding or encoding.

Options may be set by specifying *-option value* in the FFmpeg tools, or by setting the value explicitly in the `AVCodecContext` options or using the `libavutil/opt.h` API for programmatic use.

The list of supported options follow:

*b integer (encoding, audio, video)*

Set bitrate in bits/s. Default value is 200K.

*ab integer (encoding, audio)*

Set audio bitrate (in bits/s). Default value is 128K.

*bt integer (encoding, video)*

Set video bitrate tolerance (in bits/s). In 1-pass mode, bitrate tolerance specifies how far ratecontrol is willing to deviate from the target average bitrate value. This is not related to min/max bitrate.

Lowering tolerance too much has an adverse effect on quality.

*flags flags (decoding/encoding, audio, video, subtitles)*

Set generic flags.

Possible values:

`'mv4'`

Use four motion vector by macroblock (mpeg4).

`'qpel'`

Use 1/4 pel motion compensation.

`'loop'`

Use loop filter.

`'qscale'`

Use fixed qscale.

`'gmc'`

Use gmc.

`'mv0'`

Always try a mb with mv=<0,0>.

`'input_preserved'`

`'pass1'`

Use internal 2pass ratecontrol in first pass mode.

`'pass2'`

Use internal 2pass ratecontrol in second pass mode.

`'gray'`

Only decode/encode grayscale.

`'emu_edge'`

Do not draw edges.

`'psnr'`

Set error[?] variables during encoding.

`'truncated'`

`'naq'`

Normalize adaptive quantization.

`'ildct'`

Use interlaced DCT.

`'low_delay'`

Force low delay.

`'global_header'`

Place global headers in extradata instead of every keyframe.

`'bitexact'`

Only write platform-, build- and time-independent data. (except (I)DCT). This ensures that file and data checksums are reproducible and match between platforms. Its primary use is for regression testing.

`'aic'`

Apply H263 advanced intra coding / mpeg4 ac prediction.

`'cbp'`

Deprecated, use mpegvideo private options instead.

`'qprd'`

Deprecated, use mpegvideo private options instead.

`'ilme'`

Apply interlaced motion estimation.

`'cgop'`

Use closed gop.

`me_method integer (encoding, video)`

Set motion estimation method.

Possible values:

`'zero'`

zero motion estimation (fastest)

`'full'`

full motion estimation (slowest)

`'epzs'`

EPZS motion estimation (default)

‘esa’

esa motion estimation (alias for full)

‘tesa’

tesa motion estimation

‘dia’

dia motion estimation (alias for epzs)

‘log’

log motion estimation

‘phods’

phods motion estimation

‘x1’

X1 motion estimation

‘hex’

hex motion estimation

‘umh’

umh motion estimation

‘iter’

iter motion estimation

`extradata_size` *integer*

Set extradata size.

`time_base` *rational number*

Set codec time base.

It is the fundamental unit of time (in seconds) in terms of which frame timestamps are represented. For fixed-fps content, timebase should be  $1 / \text{frame\_rate}$  and timestamp increments should be identically 1.

*g integer (encoding,video)*

Set the group of picture size. Default value is 12.

*ar integer (decoding/encoding,audio)*

Set audio sampling rate (in Hz).

*ac integer (decoding/encoding,audio)*

Set number of audio channels.

*cutoff integer (encoding,audio)*

Set cutoff bandwidth.

*frame\_size integer (encoding,audio)*

Set audio frame size.

Each submitted frame except the last must contain exactly *frame\_size* samples per channel. May be 0 when the codec has CODEC\_CAP\_VARIABLE\_FRAME\_SIZE set, in that case the frame size is not restricted. It is set by some decoders to indicate constant frame size.

*frame\_number integer*

Set the frame number.

*delay integer*

*qcomp float (encoding,video)*

Set video quantizer scale compression (VBR). It is used as a constant in the ratecontrol equation. Recommended range for default rc\_eq: 0.0-1.0.

*qblur float (encoding,video)*

Set video quantizer scale blur (VBR).

*qmin integer (encoding,video)*

Set min video quantizer scale (VBR). Must be included between -1 and 69, default value is 2.

*qmax integer (encoding,video)*

Set max video quantizer scale (VBR). Must be included between -1 and 1024, default value is 31.

*qdiff integer (encoding,video)*



Set max difference between the quantizer scale (VBR).

`bf integer (encoding,video)`

Set max number of B frames between non-B-frames.

Must be an integer between -1 and 16. 0 means that B-frames are disabled. If a value of -1 is used, it will choose an automatic value depending on the encoder.

Default value is 0.

`b_qfactor float (encoding,video)`

Set qp factor between P and B frames.

`rc_strategy integer (encoding,video)`

Set ratecontrol method.

`b_strategy integer (encoding,video)`

Set strategy to choose between I/P/B-frames.

`ps integer (encoding,video)`

Set RTP payload size in bytes.

`mv_bits integer`  
`header_bits integer`  
`i_tex_bits integer`  
`p_tex_bits integer`  
`i_count integer`  
`p_count integer`  
`skip_count integer`  
`misc_bits integer`  
`frame_bits integer`  
`codec_tag integer`  
`bug flags (decoding,video)`

Workaround not auto detected encoder bugs.

Possible values:

`'autodetect'`  
`'old_msmpeg4'`

some old lavc generated msmpeg4v3 files (no autodetection)

`'xvid_ilace'`

Xvid interlacing bug (autodetected if fourcc==XVIX)

`'ump4'`

(autodetected if fourcc==UMP4)

`'no_padding'`

padding bug (autodetected)

`'amv'`

`'ac_vlc'`

illegal vlc bug (autodetected per fourcc)

`'qpel_chroma'`

`'std_qpel'`

old standard qpel (autodetected per fourcc/version)

`'qpel_chroma2'`

`'direct_blocksize'`

direct-qpel-blocksize bug (autodetected per fourcc/version)

`'edge'`

edge padding bug (autodetected per fourcc/version)

`'hpel_chroma'`

`'dc_clip'`

`'ms'`

Workaround various bugs in microsoft broken decoders.

`'trunc'`

truncated frames

`lelim integer (encoding,video)`

Set single coefficient elimination threshold for luminance (negative values also consider DC coefficient).

`celim integer (encoding,video)`

Set single coefficient elimination threshold for chrominance (negative values also consider dc coefficient)

`strict integer (decoding/encoding,audio,video)`

Specify how strictly to follow the standards.

Possible values:

‘very’

strictly conform to a older more strict version of the spec or reference software

‘strict’

strictly conform to all the things in the spec no matter what consequences

‘normal’

‘unofficial’

allow unofficial extensions

‘experimental’

allow non standardized experimental things, experimental (unfinished/work in progress/not well tested) decoders and encoders. Note: experimental decoders can pose a security risk, do not use this for decoding untrusted input.

`b_qoffset float (encoding,video)`

Set QP offset between P and B frames.

`err_detect flags (decoding,audio,video)`

Set error detection flags.

Possible values:

‘crccheck’

verify embedded CRCs

‘bitstream’

detect bitstream specification deviations

‘buffer’

detect improper bitstream length

`'explode'`

abort decoding on minor error detection

`'ignore_err'`

ignore decoding errors, and continue decoding. This is useful if you want to analyze the content of a video and thus want everything to be decoded no matter what. This option will not result in a video that is pleasing to watch in case of errors.

`'careful'`

consider things that violate the spec and have not been seen in the wild as errors

`'compliant'`

consider all spec non compliances as errors

`'aggressive'`

consider things that a sane encoder should not do as an error

`has_b_frames integer`

`block_align integer`

`mpeg_quant integer (encoding,video)`

Use MPEG quantizers instead of H.263.

`qsquish float (encoding,video)`

How to keep quantizer between qmin and qmax (0 = clip, 1 = use differentiable function).

`rc_qmod_amp float (encoding,video)`

Set experimental quantizer modulation.

`rc_qmod_freq integer (encoding,video)`

Set experimental quantizer modulation.

`rc_override_count integer`

`rc_eq string (encoding,video)`

Set rate control equation. When computing the expression, besides the standard functions defined in the section 'Expression Evaluation', the following functions are available: bits2qp(bits), qp2bits(qp). Also the following constants are available: iTex pTex tex mv fCode iCount mcVar var isI isP isB avgQP qComp avgIITex avgPITex avgPPTex avgBPTex avgTex.

`maxrate integer (encoding, audio, video)`

Set max bitrate tolerance (in bits/s). Requires bufsize to be set.

`minrate integer (encoding, audio, video)`

Set min bitrate tolerance (in bits/s). Most useful in setting up a CBR encode. It is of little use otherwise.

`bufsize integer (encoding, audio, video)`

Set ratecontrol buffer size (in bits).

`rc_buf_aggressivity float (encoding, video)`

Currently useless.

`i_qfactor float (encoding, video)`

Set QP factor between P and I frames.

`i_qoffset float (encoding, video)`

Set QP offset between P and I frames.

`rc_init_cplx float (encoding, video)`

Set initial complexity for 1-pass encoding.

`dct integer (encoding, video)`

Set DCT algorithm.

Possible values:

‘auto’

    autoselect a good one (default)

‘fastint’

    fast integer

‘int’

    accurate integer

‘mmx’

`'altivec'`  
`'faan'`

floating point AAN DCT

`lumi_mask float (encoding,video)`

Compress bright areas stronger than medium ones.

`tcplx_mask float (encoding,video)`

Set temporal complexity masking.

`scplx_mask float (encoding,video)`

Set spatial complexity masking.

`p_mask float (encoding,video)`

Set inter masking.

`dark_mask float (encoding,video)`

Compress dark areas stronger than medium ones.

`idct integer (decoding/encoding,video)`

Select IDCT implementation.

Possible values:

`'auto'`  
`'int'`  
`'simple'`  
`'simplemmx'`  
`'simpleauto'`

Automatically pick a IDCT compatible with the simple one

`'arm'`  
`'altivec'`  
`'sh4'`  
`'simplearm'`  
`'simplearmv5te'`  
`'simplearmv6'`  
`'simpleneon'`  
`'simplealpha'`

`'ipp'`  
`'xvidmmx'`  
`'faani'`

floating point AAN IDCT

`slice_count integer`  
`ec flags (decoding,video)`

Set error concealment strategy.

Possible values:

`'guess_mvs'`

iterative motion vector (MV) search (slow)

`'deblock'`

use strong deblock filter for damaged MBs

`'favor_inter'`

favor predicting from the previous frame instead of the current

`bits_per_coded_sample integer`  
`pred integer (encoding,video)`

Set prediction method.

Possible values:

`'left'`

`'plane'`

`'median'`

`aspect rational number (encoding,video)`

Set sample aspect ratio.

`debug flags (decoding/encoding,audio,video,subtitles)`

Print specific debug info.

Possible values:

`'pict'`

picture info

`'rc'`

rate control

`'bitstream'`

`'mb_type'`

macroblock (MB) type

`'qp'`

per-block quantization parameter (QP)

`'mv'`

motion vector

`'dct_coeff'`

`'green_metadata'`

display complexity metadata for the upcoming frame, GoP or for a given duration.

`'skip'`

`'startcode'`

`'pts'`

`'er'`

error recognition

`'mmco'`

memory management control operations (H.264)

`'bugs'`

`'vis_qp'`

visualize quantization parameter (QP), lower QP are tinted greener

`'vis_mb_type'`

visualize block types

`'buffers'`

picture buffer allocations

`'thread_ops'`



threading operations

‘nomc’

skip motion compensation

`vismv integer (decoding,video)`

Visualize motion vectors (MVs).

This option is deprecated, see the codecview filter instead.

Possible values:

‘pf’

forward predicted MVs of P-frames

‘bf’

forward predicted MVs of B-frames

‘bb’

backward predicted MVs of B-frames

`cmp integer (encoding,video)`

Set full pel me compare function.

Possible values:

‘sad’

sum of absolute differences, fast (default)

‘sse’

sum of squared errors

‘satd’

sum of absolute Hadamard transformed differences

‘dct’

sum of absolute DCT transformed differences

‘psnr’

sum of squared quantization errors (avoid, low quality)

‘bit’

number of bits needed for the block

‘rd’

rate distortion optimal, slow

‘zero’

0

‘vsad’

sum of absolute vertical differences

‘vsse’

sum of squared vertical differences

‘nsse’

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`subcmp integer (encoding, video)`

Set sub pel me compare function.

Possible values:

‘sad’

sum of absolute differences, fast (default)

‘sse’

sum of squared errors

‘satd’

sum of absolute Hadamard transformed differences

‘dct’

sum of absolute DCT transformed differences

‘psnr’

sum of squared quantization errors (avoid, low quality)

‘bit’

number of bits needed for the block

‘rd’

rate distortion optimal, slow

‘zero’

0

‘vsad’

sum of absolute vertical differences

‘vsse’

sum of squared vertical differences

‘nsse’

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`mbcmp integer (encoding, video)`

Set macroblock compare function.

Possible values:

`'sad'`

sum of absolute differences, fast (default)

`'sse'`

sum of squared errors

`'satd'`

sum of absolute Hadamard transformed differences

`'dct'`

sum of absolute DCT transformed differences

`'psnr'`

sum of squared quantization errors (avoid, low quality)

`'bit'`

number of bits needed for the block

`'rd'`

rate distortion optimal, slow

`'zero'`

0

`'vsad'`

sum of absolute vertical differences

`'vsse'`

sum of squared vertical differences

`'nsse'`

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`ildctcmp integer (encoding, video)`

Set interlaced dct compare function.

Possible values:

‘sad’

sum of absolute differences, fast (default)

‘sse’

sum of squared errors

‘satd’

sum of absolute Hadamard transformed differences

‘dct’

sum of absolute DCT transformed differences

‘psnr’

sum of squared quantization errors (avoid, low quality)

‘bit’

number of bits needed for the block

‘rd’

rate distortion optimal, slow

‘zero’

0

‘vsad’

sum of absolute vertical differences

‘vsse’

sum of squared vertical differences

‘nsse’

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`dia_size integer (encoding,video)`

Set diamond type & size for motion estimation.

`last_pred integer (encoding,video)`

Set amount of motion predictors from the previous frame.

`preme integer (encoding,video)`

Set pre motion estimation.

`precmp integer (encoding,video)`

Set pre motion estimation compare function.

Possible values:

‘sad’

sum of absolute differences, fast (default)

‘sse’

sum of squared errors

‘satd’

sum of absolute Hadamard transformed differences

‘dct’

sum of absolute DCT transformed differences

‘psnr’

sum of squared quantization errors (avoid, low quality)

‘bit’

number of bits needed for the block

‘rd’

rate distortion optimal, slow

‘zero’

0

‘vsad’

sum of absolute vertical differences

‘vsse’

sum of squared vertical differences

‘nsse’

noise preserving sum of squared differences

‘w53’

5/3 wavelet, only used in snow

‘w97’

9/7 wavelet, only used in snow

‘dctmax’

‘chroma’

`pre_dia_size integer (encoding,video)`

Set diamond type & size for motion estimation pre-pass.

`subq integer (encoding,video)`

Set sub pel motion estimation quality.

`dtg_active_format integer`

`me_range integer (encoding,video)`

Set limit motion vectors range (1023 for DivX player).

`ibias integer (encoding,video)`

Set intra quant bias.

`pbias integer (encoding,video)`

Set inter quant bias.

`color_table_id integer`

`global_quality integer (encoding,audio,video)`

`coder integer (encoding,video)`

Possible values:

‘vlc’

variable length coder / huffman coder

‘ac’

arithmetic coder

‘raw’

raw (no encoding)

‘rle’

run-length coder

‘deflate’

deflate-based coder

`context integer (encoding,video)`



Set context model.

```
slice_flags integer  
xvmc_acceleration integer  
mbd integer (encoding,video)
```

Set macroblock decision algorithm (high quality mode).

Possible values:

‘simple’

use mbcmp (default)

‘bits’

use fewest bits

‘rd’

use best rate distortion

```
stream_codec_tag integer  
sc_threshold integer (encoding,video)
```

Set scene change threshold.

```
lmin integer (encoding,video)
```

Set min lagrange factor (VBR).

```
lmax integer (encoding,video)
```

Set max lagrange factor (VBR).

```
nr integer (encoding,video)
```

Set noise reduction.

```
rc_init_occupancy integer (encoding,video)
```

Set number of bits which should be loaded into the rc buffer before decoding starts.

```
flags2 flags (decoding/encoding,audio,video)
```

Possible values:

‘fast’

Allow non spec compliant speedup tricks.

‘sgop’

Deprecated, use mpegvideo private options instead.

‘noout’

Skip bitstream encoding.

‘ignorecrop’

Ignore cropping information from sps.

‘local\_header’

Place global headers at every keyframe instead of in extradata.

‘chunks’

Frame data might be split into multiple chunks.

‘showall’

Show all frames before the first keyframe.

‘skipprd’

Deprecated, use mpegvideo private options instead.

‘export\_mvs’

Export motion vectors into frame side-data (see AV\_FRAME\_DATA\_MOTION\_VECTORS) for codecs that support it. See also `doc/examples/export_mvs.c`.

*error integer (encoding,video)*

*qns integer (encoding,video)*

Deprecated, use mpegvideo private options instead.

*threads integer (decoding/encoding,video)*

Possible values:

‘auto’

detect a good number of threads

`me_threshold integer (encoding,video)`

Set motion estimation threshold.

`mb_threshold integer (encoding,video)`

Set macroblock threshold.

`dc integer (encoding,video)`

Set intra\_dc\_precision.

`nssew integer (encoding,video)`

Set nsse weight.

`skip_top integer (decoding,video)`

Set number of macroblock rows at the top which are skipped.

`skip_bottom integer (decoding,video)`

Set number of macroblock rows at the bottom which are skipped.

`profile integer (encoding,audio,video)`

Possible values:

‘unknown’  
‘aac\_main’  
‘aac\_low’  
‘aac\_ssr’  
‘aac\_ltp’  
‘aac\_he’  
‘aac\_he\_v2’  
‘aac\_ld’  
‘aac\_eld’  
‘mpeg2\_aac\_low’  
‘mpeg2\_aac\_he’  
‘mpeg4\_sp’  
‘mpeg4\_core’  
‘mpeg4\_main’  
‘mpeg4\_asp’  
‘dts’  
‘dts\_es’  
‘dts\_96\_24’  
‘dts\_hd\_hra’

`'dts_hd_ma'`  
`level integer (encoding, audio, video)`

Possible values:

`'unknown'`  
`lowres integer (decoding, audio, video)`

Decode at 1= 1/2, 2=1/4, 3=1/8 resolutions.

`skip_threshold integer (encoding, video)`

Set frame skip threshold.

`skip_factor integer (encoding, video)`

Set frame skip factor.

`skip_exp integer (encoding, video)`

Set frame skip exponent. Negative values behave identical to the corresponding positive ones, except that the score is normalized. Positive values exist primarily for compatibility reasons and are not so useful.

`skipcmp integer (encoding, video)`

Set frame skip compare function.

Possible values:

`'sad'`  
sum of absolute differences, fast (default)

`'sse'`  
sum of squared errors

`'satd'`  
sum of absolute Hadamard transformed differences

`'dct'`  
sum of absolute DCT transformed differences

`'psnr'`

sum of squared quantization errors (avoid, low quality)

`'bit'`

number of bits needed for the block

`'rd'`

rate distortion optimal, slow

`'zero'`

0

`'vsad'`

sum of absolute vertical differences

`'vsse'`

sum of squared vertical differences

`'nsse'`

noise preserving sum of squared differences

`'w53'`

5/3 wavelet, only used in snow

`'w97'`

9/7 wavelet, only used in snow

`'dctmax'`

`'chroma'`

`border_mask float (encoding, video)`

Increase the quantizer for macroblocks close to borders.

`mb_lmin integer (encoding, video)`

Set min macroblock lagrange factor (VBR).

`mb_lmax integer (encoding, video)`

Set max macroblock lagrange factor (VBR).

`mepc integer (encoding,video)`

Set motion estimation bitrate penalty compensation ( $1.0 = 256$ ).

`skip_loop_filter integer (decoding,video)`

`skip_idct integer (decoding,video)`

`skip_frame integer (decoding,video)`

Make decoder discard processing depending on the frame type selected by the option value.

`skip_loop_filter` skips frame loop filtering, `skip_idct` skips frame IDCT/dequantization, `skip_frame` skips decoding.

Possible values:

‘none’

Discard no frame.

‘default’

Discard useless frames like 0-sized frames.

‘noref’

Discard all non-reference frames.

‘bidir’

Discard all bidirectional frames.

‘nokey’

Discard all frames excepts keyframes.

‘all’

Discard all frames.

Default value is ‘default’.

`bidir_refine integer (encoding,video)`

Refine the two motion vectors used in bidirectional macroblocks.

`brd_scale integer (encoding,video)`

Downscale frames for dynamic B-frame decision.

`keyint_min integer (encoding,video)`

Set minimum interval between IDR-frames.

`refs integer (encoding,video)`

Set reference frames to consider for motion compensation.

`chromaoffset integer (encoding,video)`

Set chroma qp offset from luma.

`trellis integer (encoding,audio,video)`

Set rate-distortion optimal quantization.

`sc_factor integer (encoding,video)`

Set value multiplied by qscale for each frame and added to scene\_change\_score.

`mv0_threshold integer (encoding,video)`

`b_sensitivity integer (encoding,video)`

Adjust sensitivity of b\_frame\_strategy 1.

`compression_level integer (encoding,audio,video)`

`min_prediction_order integer (encoding,audio)`

`max_prediction_order integer (encoding,audio)`

`timecode_frame_start integer (encoding,video)`

Set GOP timecode frame start number, in non drop frame format.

`request_channels integer (decoding,audio)`

Set desired number of audio channels.

`bits_per_raw_sample integer`

`channel_layout integer (decoding/encoding,audio)`

Possible values:

`request_channel_layout integer (decoding,audio)`

Possible values:

`rc_max_vbv_use float (encoding,video)`

`rc_min_vbv_use float (encoding,video)`

`ticks_per_frame integer (decoding/encoding,audio,video)`

`color_primaries integer (decoding/encoding,video)`  
`color_trc integer (decoding/encoding,video)`  
`colospace integer (decoding/encoding,video)`  
`color_range integer (decoding/encoding,video)`

If used as input parameter, it serves as a hint to the decoder, which color\_range the input has.

`chroma_sample_location integer (decoding/encoding,video)`  
`log_level_offset integer`

Set the log level offset.

`slices integer (encoding,video)`

Number of slices, used in parallelized encoding.

`thread_type flags (decoding/encoding,video)`

Select which multithreading methods to use.

Use of 'frame' will increase decoding delay by one frame per thread, so clients which cannot provide future frames should not use it.

Possible values:

'slice'

Decode more than one part of a single frame at once.

Multithreading using slices works only when the video was encoded with slices.

'frame'

Decode more than one frame at once.

Default value is 'slice+frame'.

`audio_service_type integer (encoding,audio)`

Set audio service type.

Possible values:

'ma'

Main Audio Service

'ef'



Effects

‘vi’

Visually Impaired

‘hi’

Hearing Impaired

‘di’

Dialogue

‘co’

Commentary

‘em’

Emergency

‘vo’

Voice Over

‘ka’

Karaoke

`request_sample_fmt sample_fmt (decoding, audio)`

Set sample format audio decoders should prefer. Default value is none.

`pkt_timebase rational number`

`sub_charenc encoding (decoding, subtitles)`

Set the input subtitles character encoding.

`field_order field_order (video)`

Set/override the field order of the video. Possible values:

‘progressive’

Progressive video

‘tt’

Interlaced video, top field coded and displayed first

‘bb’

Interlaced video, bottom field coded and displayed first

‘tb’

Interlaced video, top coded first, bottom displayed first

‘bt’

Interlaced video, bottom coded first, top displayed first

`skip_alpha integer (decoding,video)`

Set to 1 to disable processing alpha (transparency). This works like the ‘gray’ flag in the `flags` option which skips chroma information instead of alpha. Default is 0.

`codec_whitelist list (input)`

"," separated List of allowed decoders. By default all are allowed.

`dump_separator string (input)`

Separator used to separate the fields printed on the command line about the Stream parameters. For example to separate the fields with newlines and indentation:

```
ffprobe -dump_separator "
                        " -i ~/videos/matrixbench_mpeg2.mpg
```

## 11 Decoders# TOC

Decoders are configured elements in FFmpeg which allow the decoding of multimedia streams.

When you configure your FFmpeg build, all the supported native decoders are enabled by default. Decoders requiring an external library must be enabled manually via the corresponding `--enable-lib` option. You can list all available decoders using the configure option `--list-decoders`.

You can disable all the decoders with the configure option `--disable-decoders` and selectively enable / disable single decoders with the options `--enable-decoder=DECODER` / `--disable-decoder=DECODER`.

The option `-decoders` of the ff\* tools will display the list of enabled decoders.

## 12 Video Decoders# TOC

A description of some of the currently available video decoders follows.

### 12.1 hevc# TOC

HEVC / H.265 decoder.

Note: the `skip_loop_filter` option has effect only at level `all`.

### 12.2 rawvideo# TOC

Raw video decoder.

This decoder decodes rawvideo streams.

#### 12.2.1 Options# TOC

`top top_field_first`

Specify the assumed field type of the input video.

-1

the video is assumed to be progressive (default)

0

bottom-field-first is assumed

1

top-field-first is assumed

## 13 Audio Decoders# TOC

A description of some of the currently available audio decoders follows.

### 13.1 ac3# TOC

AC-3 audio decoder.

This decoder implements part of ATSC A/52:2010 and ETSI TS 102 366, as well as the undocumented RealAudio 3 (a.k.a. dnet).

### 13.1.1 AC-3 Decoder Options# TOC

`-drc_scale value`

Dynamic Range Scale Factor. The factor to apply to dynamic range values from the AC-3 stream. This factor is applied exponentially. There are 3 notable scale factor ranges:

`drc_scale == 0`

DRC disabled. Produces full range audio.

`0 < drc_scale <= 1`

DRC enabled. Applies a fraction of the stream DRC value. Audio reproduction is between full range and full compression.

`drc_scale > 1`

DRC enabled. Applies `drc_scale` asymmetrically. Loud sounds are fully compressed. Soft sounds are enhanced.

## 13.2 flac# TOC

FLAC audio decoder.

This decoder aims to implement the complete FLAC specification from Xiph.

### 13.2.1 FLAC Decoder options# TOC

`-use_buggy_lpc`

The lavc FLAC encoder used to produce buggy streams with high lpc values (like the default value). This option makes it possible to decode such streams correctly by using lavc's old buggy lpc logic for decoding.

## 13.3 ffwavesynth# TOC

Internal wave synthesizer.

This decoder generates wave patterns according to predefined sequences. Its use is purely internal and the format of the data it accepts is not publicly documented.

## 13.4 libcelt# TOC

libcelt decoder wrapper.

libcelt allows libavcodec to decode the Xiph CELT ultra-low delay audio codec. Requires the presence of the libcelt headers and library during configuration. You need to explicitly configure the build with `--enable-libcelt`.

## 13.5 libgsm# TOC

libgsm decoder wrapper.

libgsm allows libavcodec to decode the GSM full rate audio codec. Requires the presence of the libgsm headers and library during configuration. You need to explicitly configure the build with `--enable-libgsm`.

This decoder supports both the ordinary GSM and the Microsoft variant.

## 13.6 libilbc# TOC

libilbc decoder wrapper.

libilbc allows libavcodec to decode the Internet Low Bitrate Codec (iLBC) audio codec. Requires the presence of the libilbc headers and library during configuration. You need to explicitly configure the build with `--enable-libilbc`.

### 13.6.1 Options# TOC

The following option is supported by the libilbc wrapper.

enhance

Enable the enhancement of the decoded audio when set to 1. The default value is 0 (disabled).

## 13.7 libopencore-amrnb# TOC

libopencore-amrnb decoder wrapper.

libopencore-amrnb allows libavcodec to decode the Adaptive Multi-Rate Narrowband audio codec. Using it requires the presence of the libopencore-amrnb headers and library during configuration. You need to explicitly configure the build with `--enable-libopencore-amrnb`.

An FFmpeg native decoder for AMR-NB exists, so users can decode AMR-NB without this library.

## 13.8 libopencore-amrwb# TOC

libopencore-amrwb decoder wrapper.

libopencore-amrwb allows libavcodec to decode the Adaptive Multi-Rate Wideband audio codec. Using it requires the presence of the libopencore-amrwb headers and library during configuration. You need to explicitly configure the build with `--enable-libopencore-amrwb`.

An FFmpeg native decoder for AMR-WB exists, so users can decode AMR-WB without this library.

## 13.9 libopus# TOC

libopus decoder wrapper.

libopus allows libavcodec to decode the Opus Interactive Audio Codec. Requires the presence of the libopus headers and library during configuration. You need to explicitly configure the build with `--enable-libopus`.

An FFmpeg native decoder for Opus exists, so users can decode Opus without this library.

## 14 Subtitles Decoders# TOC

### 14.1 dvbsub# TOC

#### 14.1.1 Options# TOC

`compute_clut`

-1

Compute clut if no matching CLUT is in the stream.

0

Never compute CLUT

1

Always compute CLUT and override the one provided in the stream.

`dvb_substream`

Selects the dvb substream, or all substreams if -1 which is default.

### 14.2 dvdsub# TOC

This codec decodes the bitmap subtitles used in DVDs; the same subtitles can also be found in VobSub file pairs and in some Matroska files.

#### 14.2.1 Options# TOC

`palette`

Specify the global palette used by the bitmaps. When stored in VobSub, the palette is normally specified in the index file; in Matroska, the palette is stored in the codec extra-data in the same format as in VobSub. In DVDs, the palette is stored in the IFO file, and therefore not available when reading from dumped VOB files.

The format for this option is a string containing 16 24-bits hexadecimal numbers (without 0x prefix) separated by comas, for example 0d00ee, ee450d, 101010, eaeaea, 0ce60b, ec14ed, ebff0b, 0d617a, 7b7b7b, d1d1d1, 7b2a0e, 0d950c, 0f007b, cf0dec, cfa80c, 7c127b.

ifo\_palette

Specify the IFO file from which the global palette is obtained. (experimental)

forced\_subs\_only

Only decode subtitle entries marked as forced. Some titles have forced and non-forced subtitles in the same track. Setting this flag to 1 will only keep the forced subtitles. Default value is 0.

## 14.3 libzvbi-teletext# TOC

Libzvbi allows libavcodec to decode DVB teletext pages and DVB teletext subtitles. Requires the presence of the libzvbi headers and library during configuration. You need to explicitly configure the build with `--enable-libzvbi`.

### 14.3.1 Options# TOC

txt\_page

List of teletext page numbers to decode. You may use the special \* string to match all pages. Pages that do not match the specified list are dropped. Default value is \*.

txt\_chop\_top

Discards the top teletext line. Default value is 1.

txt\_format

Specifies the format of the decoded subtitles. The teletext decoder is capable of decoding the teletext pages to bitmaps or to simple text, you should use "bitmap" for teletext pages, because certain graphics and colors cannot be expressed in simple text. You might use "text" for teletext based subtitles if your application can handle simple text based subtitles. Default value is bitmap.

txt\_left

X offset of generated bitmaps, default is 0.

txt\_top

Y offset of generated bitmaps, default is 0.

txt\_chop\_spaces

Chops leading and trailing spaces and removes empty lines from the generated text. This option is useful for teletext based subtitles where empty spaces may be present at the start or at the end of the lines or empty lines may be present between the subtitle lines because of double-sized teletext characters. Default value is 1.

`txt_duration`

Sets the display duration of the decoded teletext pages or subtitles in milliseconds. Default value is 30000 which is 30 seconds.

`txt_transparent`

Force transparent background of the generated teletext bitmaps. Default value is 0 which means an opaque (black) background.

## 15 Encoders# TOC

Encoders are configured elements in FFmpeg which allow the encoding of multimedia streams.

When you configure your FFmpeg build, all the supported native encoders are enabled by default. Encoders requiring an external library must be enabled manually via the corresponding `--enable-lib` option. You can list all available encoders using the configure option `--list-encoders`.

You can disable all the encoders with the configure option `--disable-encoders` and selectively enable / disable single encoders with the options `--enable-encoder=ENCODER` / `--disable-encoder=ENCODER`.

The option `-encoders` of the `ff*` tools will display the list of enabled encoders.

## 16 Audio Encoders# TOC

A description of some of the currently available audio encoders follows.

### 16.1 aac# TOC

Advanced Audio Coding (AAC) encoder.

This encoder is an experimental FFmpeg-native AAC encoder. Currently only the low complexity (AAC-LC) profile is supported. To use this encoder, you must set `strict` option to 'experimental' or lower.

As this encoder is experimental, unexpected behavior may exist from time to time. For a more stable AAC encoder, see `libvo-aacenc`. However, be warned that it has a worse quality reported by some users.

See also `libfdk_aac` and `libfaac`.



## 16.1.1 Options# TOC

b

Set bit rate in bits/s. Setting this automatically activates constant bit rate (CBR) mode.

q

Set quality for variable bit rate (VBR) mode. This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

stereo\_mode

Set stereo encoding mode. Possible values:

‘auto’

Automatically selected by the encoder.

‘ms\_off’

Disable middle/side encoding. This is the default.

‘ms\_force’

Force middle/side encoding.

aac\_coder

Set AAC encoder coding method. Possible values:

‘faac’

FAAC-inspired method.

This method is a simplified reimplementation of the method used in FAAC, which sets thresholds proportional to the band energies, and then decreases all the thresholds with quantizer steps to find the appropriate quantization with distortion below threshold band by band.

The quality of this method is comparable to the two loop searching method described below, but somewhat a little better and slower.

‘anmr’

Average noise to mask ratio (ANMR) trellis-based solution.

This has a theoretic best quality out of all the coding methods, but at the cost of the slowest speed.

`'twoloop'`

Two loop searching (TLS) method.

This method first sets quantizers depending on band thresholds and then tries to find an optimal combination by adding or subtracting a specific value from all quantizers and adjusting some individual quantizer a little.

This method produces similar quality with the FAAC method and is the default.

`'fast'`

Constant quantizer method.

This method sets a constant quantizer for all bands. This is the fastest of all the methods, yet produces the worst quality.

## 16.2 ac3 and ac3\_fixed# TOC

AC-3 audio encoders.

These encoders implement part of ATSC A/52:2010 and ETSI TS 102 366, as well as the undocumented RealAudio 3 (a.k.a. dnet).

The *ac3* encoder uses floating-point math, while the *ac3\_fixed* encoder only uses fixed-point integer math. This does not mean that one is always faster, just that one or the other may be better suited to a particular system. The floating-point encoder will generally produce better quality audio for a given bitrate. The *ac3\_fixed* encoder is not the default codec for any of the output formats, so it must be specified explicitly using the option `-acodec ac3_fixed` in order to use it.

### 16.2.1 AC-3 Metadata# TOC

The AC-3 metadata options are used to set parameters that describe the audio, but in most cases do not affect the audio encoding itself. Some of the options do directly affect or influence the decoding and playback of the resulting bitstream, while others are just for informational purposes. A few of the options will add bits to the output stream that could otherwise be used for audio data, and will thus affect the quality of the output. Those will be indicated accordingly with a note in the option list below.

These parameters are described in detail in several publicly-available documents.

- A/52:2010 - Digital Audio Compression (AC-3) (E-AC-3) Standard
- A/54 - Guide to the Use of the ATSC Digital Television Standard
- Dolby Metadata Guide
- Dolby Digital Professional Encoding Guidelines

### 16.2.1.1 Metadata Control Options# TOC

`-per_frame_metadata` *boolean*

Allow Per-Frame Metadata. Specifies if the encoder should check for changing metadata for each frame.

0

The metadata values set at initialization will be used for every frame in the stream. (default)

1

Metadata values can be changed before encoding each frame.

### 16.2.1.2 Downmix Levels# TOC

`-center_mixlev` *level*

Center Mix Level. The amount of gain the decoder should apply to the center channel when downmixing to stereo. This field will only be written to the bitstream if a center channel is present. The value is specified as a scale factor. There are 3 valid values:

0.707

Apply -3dB gain

0.595

Apply -4.5dB gain (default)

0.500

Apply -6dB gain

`-surround_mixlev` *level*

Surround Mix Level. The amount of gain the decoder should apply to the surround channel(s) when downmixing to stereo. This field will only be written to the bitstream if one or more surround channels are present. The value is specified as a scale factor. There are 3 valid values:

0.707

Apply -3dB gain

0.500

Apply -6dB gain (default)

0.000

Silence Surround Channel(s)

### 16.2.1.3 Audio Production Information# TOC

Audio Production Information is optional information describing the mixing environment. Either none or both of the fields are written to the bitstream.

`-mixing_level number`

Mixing Level. Specifies peak sound pressure level (SPL) in the production environment when the mix was mastered. Valid values are 80 to 111, or -1 for unknown or not indicated. The default value is -1, but that value cannot be used if the Audio Production Information is written to the bitstream. Therefore, if the `room_type` option is not the default value, the `mixing_level` option must not be -1.

`-room_type type`

Room Type. Describes the equalization used during the final mixing session at the studio or on the dubbing stage. A large room is a dubbing stage with the industry standard X-curve equalization; a small room has flat equalization. This field will not be written to the bitstream if both the `mixing_level` option and the `room_type` option have the default values.

0

notindicated

Not Indicated (default)

1

large

Large Room

2

small

Small Room

### 16.2.1.4 Other Metadata Options# TOC

`-copyright boolean`

Copyright Indicator. Specifies whether a copyright exists for this audio.

0  
off

No Copyright Exists (default)

1  
on

Copyright Exists

`-dialnorm value`

Dialogue Normalization. Indicates how far the average dialogue level of the program is below digital 100% full scale (0 dBFS). This parameter determines a level shift during audio reproduction that sets the average volume of the dialogue to a preset level. The goal is to match volume level between program sources. A value of -31dB will result in no volume level change, relative to the source volume, during audio reproduction. Valid values are whole numbers in the range -31 to -1, with -31 being the default.

`-dsur_mode mode`

Dolby Surround Mode. Specifies whether the stereo signal uses Dolby Surround (Pro Logic). This field will only be written to the bitstream if the audio stream is stereo. Using this option does **NOT** mean the encoder will actually apply Dolby Surround processing.

0  
notindicated

Not Indicated (default)

1  
off

Not Dolby Surround Encoded

2  
on

Dolby Surround Encoded

`-original boolean`

Original Bit Stream Indicator. Specifies whether this audio is from the original source and not a copy.

0  
off

Not Original Source

1  
on

Original Source (default)

## 16.2.2 Extended Bitstream Information# TOC

The extended bitstream options are part of the Alternate Bit Stream Syntax as specified in Annex D of the A/52:2010 standard. It is grouped into 2 parts. If any one parameter in a group is specified, all values in that group will be written to the bitstream. Default values are used for those that are written but have not been specified. If the mixing levels are written, the decoder will use these values instead of the ones specified in the `center_mixlev` and `surround_mixlev` options if it supports the Alternate Bit Stream Syntax.

### 16.2.2.1 Extended Bitstream Information - Part 1# TOC

`-dmix_mode mode`

Preferred Stereo Downmix Mode. Allows the user to select either Lt/Rt (Dolby Surround) or Lo/Ro (normal stereo) as the preferred stereo downmix mode.

0  
notindicated

Not Indicated (default)

1  
ltrt

Lt/Rt Downmix Preferred

2  
loro

Lo/Ro Downmix Preferred

`-ltrt_cmixlev level`

Lt/Rt Center Mix Level. The amount of gain the decoder should apply to the center channel when downmixing to stereo in Lt/Rt mode.

1.414

Apply +3dB gain

1.189

Apply +1.5dB gain

1.000

Apply 0dB gain

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain (default)

0.500

Apply -6.0dB gain

0.000

Silence Center Channel

`-lttrt_surmixlev level`

Lt/Rt Surround Mix Level. The amount of gain the decoder should apply to the surround channel(s) when downmixing to stereo in Lt/Rt mode.

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain

0.500

Apply -6.0dB gain (default)

0.000

Silence Surround Channel(s)

`-loro_cmixlev level`

Lo/Ro Center Mix Level. The amount of gain the decoder should apply to the center channel when downmixing to stereo in Lo/Ro mode.

1.414

Apply +3dB gain

1.189

Apply +1.5dB gain

1.000

Apply 0dB gain

0.841

Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain (default)

0.500

Apply -6.0dB gain

0.000

Silence Center Channel

`-loro_surmixlev level`

Lo/Ro Surround Mix Level. The amount of gain the decoder should apply to the surround channel(s) when downmixing to stereo in Lo/Ro mode.

0.841



Apply -1.5dB gain

0.707

Apply -3.0dB gain

0.595

Apply -4.5dB gain

0.500

Apply -6.0dB gain (default)

0.000

Silence Surround Channel(s)

### 16.2.2.2 Extended Bitstream Information - Part 2# TOC

`-dsurex_mode mode`

Dolby Surround EX Mode. Indicates whether the stream uses Dolby Surround EX (7.1 matrixed to 5.1). Using this option does **NOT** mean the encoder will actually apply Dolby Surround EX processing.

0

notindicated

Not Indicated (default)

1

on

Dolby Surround EX Off

2

off

Dolby Surround EX On

`-dheadphone_mode mode`

Dolby Headphone Mode. Indicates whether the stream uses Dolby Headphone encoding (multi-channel matrixed to 2.0 for use with headphones). Using this option does **NOT** mean the encoder will actually apply Dolby Headphone processing.

0

notindicated

Not Indicated (default)

1  
on

Dolby Headphone Off

2  
off

Dolby Headphone On

`-ad_conv_type type`

A/D Converter Type. Indicates whether the audio has passed through HDCD A/D conversion.

0  
standard

Standard A/D Converter (default)

1  
hdc

HDCD A/D Converter

### 16.2.3 Other AC-3 Encoding Options# TOC

`-stereo_rematrixing boolean`

Stereo Rematrixing. Enables/Disables use of rematrixing for stereo input. This is an optional AC-3 feature that increases quality by selectively encoding the left/right channels as mid/side. This option is enabled by default, and it is highly recommended that it be left as enabled except for testing purposes.

### 16.2.4 Floating-Point-Only AC-3 Encoding Options# TOC

These options are only valid for the floating-point encoder and do not exist for the fixed-point encoder due to the corresponding features not being implemented in fixed-point.

`-channel_coupling boolean`

Enables/Disables use of channel coupling, which is an optional AC-3 feature that increases quality by combining high frequency information from multiple channels into a single channel. The per-channel high frequency information is sent with less accuracy in both the frequency and time domains. This allows more bits to be used for lower frequencies while preserving enough information to reconstruct the high frequencies. This option is enabled by default for the floating-point encoder and should

generally be left as enabled except for testing purposes or to increase encoding speed.

-1  
auto

Selected by Encoder (default)

0  
off

Disable Channel Coupling

1  
on

Enable Channel Coupling

`-cpl_start_band number`

Coupling Start Band. Sets the channel coupling start band, from 1 to 15. If a value higher than the bandwidth is used, it will be reduced to 1 less than the coupling end band. If *auto* is used, the start band will be determined by the encoder based on the bit rate, sample rate, and channel layout. This option has no effect if channel coupling is disabled.

-1  
auto

Selected by Encoder (default)

## 16.3 flac# TOC

FLAC (Free Lossless Audio Codec) Encoder

### 16.3.1 Options# TOC

The following options are supported by FFmpeg's flac encoder.

`compression_level`

Sets the compression level, which chooses defaults for many other options if they are not set explicitly.

`frame_size`

Sets the size of the frames in samples per channel.

`lpc_coeff_precision`

Sets the LPC coefficient precision, valid values are from 1 to 15, 15 is the default.

`lpc_type`

Sets the first stage LPC algorithm

`'none'`

LPC is not used

`'fixed'`

fixed LPC coefficients

`'levinson'`

`'cholesky'`

`lpc_passes`

Number of passes to use for Cholesky factorization during LPC analysis

`min_partition_order`

The minimum partition order

`max_partition_order`

The maximum partition order

`prediction_order_method`

`'estimation'`

`'2level'`

`'4level'`

`'8level'`

`'search'`

Bruteforce search

`'log'`

`ch_mode`

Channel mode

`'auto'`

The mode is chosen automatically for each frame

`'indep'`

Channels are independently coded

```
'left_side'  
'right_side'  
'mid_side'  
exact_rice_parameters
```

Chooses if rice parameters are calculated exactly or approximately. if set to 1 then they are chosen exactly, which slows the code down slightly and improves compression slightly.

```
multi_dim_quant
```

Multi Dimensional Quantization. If set to 1 then a 2nd stage LPC algorithm is applied after the first stage to finetune the coefficients. This is quite slow and slightly improves compression.

## 16.4 libfaac# TOC

libfaac AAC (Advanced Audio Coding) encoder wrapper.

Requires the presence of the libfaac headers and library during configuration. You need to explicitly configure the build with `--enable-libfaac --enable-nonfree`.

This encoder is considered to be of higher quality with respect to the the native experimental FFmpeg AAC encoder.

For more information see the libfaac project at <http://www.audiocoding.com/faac.html/>.

### 16.4.1 Options# TOC

The following shared FFmpeg codec options are recognized.

The following options are supported by the libfaac wrapper. The `faac`-equivalent of the options are listed in parentheses.

`b (-b)`

Set bit rate in bits/s for ABR (Average Bit Rate) mode. If the bit rate is not explicitly specified, it is automatically set to a suitable value depending on the selected profile. `faac` bitrate is expressed in kilobits/s.

Note that libfaac does not support CBR (Constant Bit Rate) but only ABR (Average Bit Rate).

If VBR mode is enabled this option is ignored.

`ar (-R)`

Set audio sampling rate (in Hz).

`ac (-c)`

Set the number of audio channels.

`cutoff (-C)`

Set cutoff frequency. If not specified (or explicitly set to 0) it will use a value automatically computed by the library. Default value is 0.

`profile`

Set audio profile.

The following profiles are recognized:

`'aac_main'`

Main AAC (Main)

`'aac_low'`

Low Complexity AAC (LC)

`'aac_ssr'`

Scalable Sample Rate (SSR)

`'aac_ltp'`

Long Term Prediction (LTP)

If not specified it is set to `'aac_low'`.

`flags +qscale`

Set constant quality VBR (Variable Bit Rate) mode.

`global_quality`

Set quality in VBR mode as an integer number of lambda units.

Only relevant when VBR mode is enabled with `flags +qscale`. The value is converted to QP units by dividing it by `FF_QP2LAMBDA`, and used to set the quality value used by libfaac. A reasonable range for the option value in QP units is [10-500], the higher the value the higher the quality.

`q (-q)`

Enable VBR mode when set to a non-negative value, and set constant quality value as a double floating point value in QP units.

The value sets the quality value used by libfaac. A reasonable range for the option value is [10-500], the higher the value the higher the quality.

This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

## 16.4.2 Examples# TOC

- Use `ffmpeg` to convert an audio file to ABR 128 kbps AAC in an M4A (MP4) container:

```
ffmpeg -i input.wav -codec:a libfaac -b:a 128k -output.m4a
```

- Use `ffmpeg` to convert an audio file to VBR AAC, using the LTP AAC profile:

```
ffmpeg -i input.wav -c:a libfaac -profile:a aac_ltp -q:a 100 output.m4a
```

## 16.5 libfdk\_aac# TOC

libfdk-aac AAC (Advanced Audio Coding) encoder wrapper.

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

Requires the presence of the libfdk-aac headers and library during configuration. You need to explicitly configure the build with `--enable-libfdk-aac`. The library is also incompatible with GPL, so if you allow the use of GPL, you should configure with `--enable-gpl --enable-nonfree --enable-libfdk-aac`.

This encoder is considered to be of higher quality with respect to both the native experimental FFmpeg AAC encoder and libfaac.

VBR encoding, enabled through the `vbr` or `flags +qscale` options, is experimental and only works with some combinations of parameters.

Support for encoding 7.1 audio is only available with libfdk-aac 0.1.3 or higher.

For more information see the fdk-aac project at <http://sourceforge.net/p/opencore-amr/fdk-aac/>.

### 16.5.1 Options# TOC

The following options are mapped on the shared FFmpeg codec options.

b

Set bit rate in bits/s. If the bitrate is not explicitly specified, it is automatically set to a suitable value depending on the selected profile.

In case VBR mode is enabled the option is ignored.

ar

Set audio sampling rate (in Hz).

channels

Set the number of audio channels.

flags +qscale

Enable fixed quality, VBR (Variable Bit Rate) mode. Note that VBR is implicitly enabled when the vbr value is positive.

cutoff

Set cutoff frequency. If not specified (or explicitly set to 0) it will use a value automatically computed by the library. Default value is 0.

profile

Set audio profile.

The following profiles are recognized:

‘aac\_low’

Low Complexity AAC (LC)

‘aac\_he’

High Efficiency AAC (HE-AAC)

‘aac\_he\_v2’

High Efficiency AAC version 2 (HE-AACv2)

‘aac\_ld’

Low Delay AAC (LD)

‘aac\_eld’

Enhanced Low Delay AAC (ELD)

If not specified it is set to ‘aac\_low’.



The following are private options of the libfdk\_aac encoder.

#### afterburner

Enable afterburner feature if set to 1, disabled if set to 0. This improves the quality but also the required processing power.

Default value is 1.

#### eld\_sbr

Enable SBR (Spectral Band Replication) for ELD if set to 1, disabled if set to 0.

Default value is 0.

#### signaling

Set SBR/PS signaling style.

It can assume one of the following values:

‘default’

choose signaling implicitly (explicit hierarchical by default, implicit if global header is disabled)

‘implicit’

implicit backwards compatible signaling

‘explicit\_sbr’

explicit SBR, implicit PS signaling

‘explicit\_hierarchical’

explicit hierarchical signaling

Default value is ‘default’.

#### latm

Output LATM/LOAS encapsulated data if set to 1, disabled if set to 0.

Default value is 0.

#### header\_period

Set StreamMuxConfig and PCE repetition period (in frames) for sending in-band configuration buffers within LATM/LOAS transport layer.

Must be a 16-bits non-negative integer.

Default value is 0.

vbr

Set VBR mode, from 1 to 5. 1 is lowest quality (though still pretty good) and 5 is highest quality. A value of 0 will disable VBR, and CBR (Constant Bit Rate) is enabled.

Currently only the 'aac\_low' profile supports VBR encoding.

VBR modes 1-5 correspond to roughly the following average bit rates:

'1'

32 kbps/channel

'2'

40 kbps/channel

'3'

48-56 kbps/channel

'4'

64 kbps/channel

'5'

about 80-96 kbps/channel

Default value is 0.

## 16.5.2 Examples# TOC

- Use `ffmpeg` to convert an audio file to VBR AAC in an M4A (MP4) container:

```
ffmpeg -i input.wav -codec:a libfdk_aac -vbr 3 output.m4a
```

- Use `ffmpeg` to convert an audio file to CBR 64k kbps AAC, using the High-Efficiency AAC profile:

```
ffmpeg -i input.wav -c:a libfdk_aac -profile:a aac_he -b:a 64k output.m4a
```

## 16.6 libmp3lame# TOC

LAME (Lame Ain't an MP3 Encoder) MP3 encoder wrapper.

Requires the presence of the libmp3lame headers and library during configuration. You need to explicitly configure the build with `--enable-libmp3lame`.

See libshine for a fixed-point MP3 encoder, although with a lower quality.

### 16.6.1 Options# TOC

The following options are supported by the libmp3lame wrapper. The lame-equivalent of the options are listed in parentheses.

`b (-b)`

Set bitrate expressed in bits/s for CBR or ABR. LAME `bitrate` is expressed in kilobits/s.

`q (-V)`

Set constant quality setting for VBR. This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

`compression_level (-q)`

Set algorithm quality. Valid arguments are integers in the 0-9 range, with 0 meaning highest quality but slowest, and 9 meaning fastest while producing the worst quality.

`reservoir`

Enable use of bit reservoir when set to 1. Default value is 1. LAME has this enabled by default, but can be overridden by use `--nores` option.

`joint_stereo (-m j)`

Enable the encoder to use (on a frame by frame basis) either L/R stereo or mid/side stereo. Default value is 1.

`abr (--abr)`

Enable the encoder to use ABR when set to 1. The lame `--abr` sets the target bitrate, while this options only tells FFmpeg to use ABR still relies on `b` to set bitrate.

## 16.7 libopencore-amrnb# TOC

OpenCORE Adaptive Multi-Rate Narrowband encoder.

Requires the presence of the libopencore-amrnb headers and library during configuration. You need to explicitly configure the build with `--enable-libopencore-amrnb --enable-version3`.

This is a mono-only encoder. Officially it only supports 8000Hz sample rate, but you can override it by setting `strict` to 'unofficial' or lower.

### 16.7.1 Options# TOC

b

Set bitrate in bits per second. Only the following bitrates are supported, otherwise libavcodec will round to the nearest valid bitrate.

4750  
5150  
5900  
6700  
7400  
7950  
10200  
12200

dtx

Allow discontinuous transmission (generate comfort noise) when set to 1. The default value is 0 (disabled).

### 16.8 libshine# TOC

Shine Fixed-Point MP3 encoder wrapper.

Shine is a fixed-point MP3 encoder. It has a far better performance on platforms without an FPU, e.g. armel CPUs, and some phones and tablets. However, as it is more targeted on performance than quality, it is not on par with LAME and other production-grade encoders quality-wise. Also, according to the project's homepage, this encoder may not be free of bugs as the code was written a long time ago and the project was dead for at least 5 years.

This encoder only supports stereo and mono input. This is also CBR-only.

The original project (last updated in early 2007) is at <http://sourceforge.net/projects/libshine-fxp/>. We only support the updated fork by the Savonet/Liquidsoap project at <https://github.com/savonet/shine>.

Requires the presence of the libshine headers and library during configuration. You need to explicitly configure the build with `--enable-libshine`.

See also libmp3lame.

## 16.8.1 Options# TOC

The following options are supported by the libshine wrapper. The `shineenc`-equivalent of the options are listed in parentheses.

`b (-b)`

Set bitrate expressed in bits/s for CBR. `shineenc -b` option is expressed in kilobits/s.

## 16.9 libtwolame# TOC

TwoLAME MP2 encoder wrapper.

Requires the presence of the libtwolame headers and library during configuration. You need to explicitly configure the build with `--enable-libtwolame`.

### 16.9.1 Options# TOC

The following options are supported by the libtwolame wrapper. The `twolame`-equivalent options follow the FFmpeg ones and are in parentheses.

`b (-b)`

Set bitrate expressed in bits/s for CBR. `twolame b` option is expressed in kilobits/s. Default value is 128k.

`q (-V)`

Set quality for experimental VBR support. Maximum value range is from -50 to 50, useful range is from -10 to 10. The higher the value, the better the quality. This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

`mode (--mode)`

Set the mode of the resulting audio. Possible values:

`'auto'`

Choose mode automatically based on the input. This is the default.

`'stereo'`

Stereo

`'joint_stereo'`

Joint stereo

`'dual_channel'`

Dual channel

`'mono'`

Mono

`psymodel (--psyc-mode)`

Set psychoacoustic model to use in encoding. The argument must be an integer between -1 and 4, inclusive. The higher the value, the better the quality. The default value is 3.

`energy_levels (--energy)`

Enable energy levels extensions when set to 1. The default value is 0 (disabled).

`error_protection (--protect)`

Enable CRC error protection when set to 1. The default value is 0 (disabled).

`copyright (--copyright)`

Set MPEG audio copyright flag when set to 1. The default value is 0 (disabled).

`original (--original)`

Set MPEG audio original flag when set to 1. The default value is 0 (disabled).

## 16.10 libvo-aacenc# TOC

VisualOn AAC encoder.

Requires the presence of the libvo-aacenc headers and library during configuration. You need to explicitly configure the build with `--enable-libvo-aacenc --enable-version3`.

This encoder is considered to be worse than the native experimental FFmpeg AAC encoder, according to multiple sources.

### 16.10.1 Options# TOC

The VisualOn AAC encoder only support encoding AAC-LC and up to 2 channels. It is also CBR-only.

b

Set bit rate in bits/s.

## 16.11 libvo-amrwbenc# TOC

VisualOn Adaptive Multi-Rate Wideband encoder.

Requires the presence of the libvo-amrwbenc headers and library during configuration. You need to explicitly configure the build with `--enable-libvo-amrwbenc --enable-version3`.

This is a mono-only encoder. Officially it only supports 16000Hz sample rate, but you can override it by setting `strict` to 'unofficial' or lower.

### 16.11.1 Options# TOC

`b`

Set bitrate in bits/s. Only the following bitrates are supported, otherwise libavcodec will round to the nearest valid bitrate.

'6600'  
'8850'  
'12650'  
'14250'  
'15850'  
'18250'  
'19850'  
'23050'  
'23850'

`dtx`

Allow discontinuous transmission (generate comfort noise) when set to 1. The default value is 0 (disabled).

## 16.12 libopus# TOC

libopus Opus Interactive Audio Codec encoder wrapper.

Requires the presence of the libopus headers and library during configuration. You need to explicitly configure the build with `--enable-libopus`.

### 16.12.1 Option Mapping# TOC

Most libopus options are modelled after the `opusenc` utility from `opus-tools`. The following is an option mapping chart describing options supported by the libopus wrapper, and their `opusenc`-equivalent in parentheses.

`b` (*bitrate*)

Set the bit rate in bits/s. FFmpeg's `b` option is expressed in bits/s, while `opusenc`'s `bitrate` in kilobits/s.

`vbr` (*vbr*, *hard-cbr*, and *cvbr*)

Set VBR mode. The FFmpeg `vbr` option has the following valid arguments, with the their `opusenc` equivalent options in parentheses:

`'off (hard-cbr)'`

Use constant bit rate encoding.

`'on (vbr)'`

Use variable bit rate encoding (the default).

`'constrained (cvbr)'`

Use constrained variable bit rate encoding.

`compression_level` (*comp*)

Set encoding algorithm complexity. Valid options are integers in the 0-10 range. 0 gives the fastest encodes but lower quality, while 10 gives the highest quality but slowest encoding. The default is 10.

`frame_duration` (*framesize*)

Set maximum frame size, or duration of a frame in milliseconds. The argument must be exactly the following: 2.5, 5, 10, 20, 40, 60. Smaller frame sizes achieve lower latency but less quality at a given bitrate. Sizes greater than 20ms are only interesting at fairly low bitrates. The default is 20ms.

`packet_loss` (*expect-loss*)

Set expected packet loss percentage. The default is 0.

`application` (N.A.)

Set intended application type. Valid options are listed below:

`'voip'`

Favor improved speech intelligibility.

`'audio'`

Favor faithfulness to the input (the default).

`'lowdelay'`



Restrict to only the lowest delay modes.

`cutoff` (N.A.)

Set cutoff bandwidth in Hz. The argument must be exactly one of the following: 4000, 6000, 8000, 12000, or 20000, corresponding to narrowband, mediumband, wideband, super wideband, and fullband respectively. The default is 0 (cutoff disabled).

## 16.13 libvorbis# TOC

libvorbis encoder wrapper.

Requires the presence of the libvorbisenc headers and library during configuration. You need to explicitly configure the build with `--enable-libvorbis`.

### 16.13.1 Options# TOC

The following options are supported by the libvorbis wrapper. The `oggenc`-equivalent of the options are listed in parentheses.

To get a more accurate and extensive documentation of the libvorbis options, consult the libvorbisenc's and oggenc's documentations. See <http://xiph.org/vorbis/>, <http://wiki.xiph.org/Vorbis-tools>, and `oggenc(1)`.

`b` (`-b`)

Set bitrate expressed in bits/s for ABR. `oggenc -b` is expressed in kilobits/s.

`q` (`-q`)

Set constant quality setting for VBR. The value should be a float number in the range of -1.0 to 10.0. The higher the value, the better the quality. The default value is '3.0'.

This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

`cutoff` (`--advanced-encode-option lowpass_frequency=N`)

Set cutoff bandwidth in Hz, a value of 0 disables cutoff. `oggenc`'s related option is expressed in kHz. The default value is '0' (cutoff disabled).

`minrate` (`-m`)

Set minimum bitrate expressed in bits/s. `oggenc -m` is expressed in kilobits/s.

`maxrate` (`-M`)

Set maximum bitrate expressed in bits/s. `oggenc -M` is expressed in kilobits/s. This only has effect on ABR mode.

`iblock (--advanced-encode-option impulse_noisetune=N)`

Set noise floor bias for impulse blocks. The value is a float number from -15.0 to 0.0. A negative bias instructs the encoder to pay special attention to the crispness of transients in the encoded audio. The tradeoff for better transient response is a higher bitrate.

## 16.14 libwavpack# TOC

A wrapper providing WavPack encoding through libwavpack.

Only lossless mode using 32-bit integer samples is supported currently.

Requires the presence of the libwavpack headers and library during configuration. You need to explicitly configure the build with `--enable-libwavpack`.

Note that a libavcodec-native encoder for the WavPack codec exists so users can encode audios with this codec without using this encoder. See `wavpackenc`.

### 16.14.1 Options# TOC

wavpack command line utility's corresponding options are listed in parentheses, if any.

`frame_size (--blocksize)`

Default is 32768.

`compression_level`

Set speed vs. compression tradeoff. Acceptable arguments are listed below:

`'0 (-f)'`

Fast mode.

`'1'`

Normal (default) settings.

`'2 (-h)'`

High quality.

`'3 (-hh)'`

Very high quality.

`'4-8 (-hh -xEXTRAPROC)'`

Same as '3', but with extra processing enabled.

'4' is the same as -x2 and '8' is the same as -x6.

## 16.15 wavpack# TOC

WavPack lossless audio encoder.

This is a libavcodec-native WavPack encoder. There is also an encoder based on libwavpack, but there is virtually no reason to use that encoder.

See also libwavpack.

### 16.15.1 Options# TOC

The equivalent options for wavpack command line utility are listed in parentheses.

#### 16.15.1.1 Shared options# TOC

The following shared options are effective for this encoder. Only special notes about this particular encoder will be documented here. For the general meaning of the options, see the Codec Options chapter.

`frame_size (--blocksize)`

For this encoder, the range for this option is between 128 and 131072. Default is automatically decided based on sample rate and number of channel.

For the complete formula of calculating default, see `libavcodec/wavpackenc.c`.

`compression_level (-f, -h, -hh, and -x)`

This option's syntax is consistent with libwavpack's.

#### 16.15.1.2 Private options# TOC

`joint_stereo (-j)`

Set whether to enable joint stereo. Valid values are:

`'on (1)'`

Force mid/side audio encoding.

`'off (0)'`

Force left/right audio encoding.

`'auto'`

Let the encoder decide automatically.

`optimize_mono`

Set whether to enable optimization for mono. This option is only effective for non-mono streams.  
Available values:

`'on'`

enabled

`'off'`

disabled

## 17 Video Encoders# TOC

A description of some of the currently available video encoders follows.

### 17.1 jpeg2000# TOC

The native jpeg 2000 encoder is lossy by default, the `-q:v` option can be used to set the encoding quality. Lossless encoding can be selected with `-pred 1`.

#### 17.1.1 Options# TOC

`format`

Can be set to either `j2k` or `jp2` (the default) that makes it possible to store non-rgb `pix_fmts`.

### 17.2 snow# TOC

#### 17.2.1 Options# TOC

`iterative_dia_size`

dia size for the iterative motion estimation

## 17.3 libtheora# TOC

libtheora Theora encoder wrapper.

Requires the presence of the libtheora headers and library during configuration. You need to explicitly configure the build with `--enable-libtheora`.

For more information about the libtheora project see <http://www.theora.org/>.

### 17.3.1 Options# TOC

The following global options are mapped to internal libtheora options which affect the quality and the bitrate of the encoded stream.

`b`

Set the video bitrate in bit/s for CBR (Constant Bit Rate) mode. In case VBR (Variable Bit Rate) mode is enabled this option is ignored.

`flags`

Used to enable constant quality mode (VBR) encoding through the `qscale` flag, and to enable the `pass1` and `pass2` modes.

`g`

Set the GOP size.

`global_quality`

Set the global quality as an integer in lambda units.

Only relevant when VBR mode is enabled with `flags +qscale`. The value is converted to QP units by dividing it by `FF_QP2LAMBDA`, clipped in the [0 - 10] range, and then multiplied by 6.3 to get a value in the native libtheora range [0-63]. A higher value corresponds to a higher quality.

`q`

Enable VBR mode when set to a non-negative value, and set constant quality value as a double floating point value in QP units.

The value is clipped in the [0-10] range, and then multiplied by 6.3 to get a value in the native libtheora range [0-63].

This option is valid only using the `ffmpeg` command-line tool. For library interface users, use `global_quality`.

## 17.3.2 Examples# TOC

- Set maximum constant quality (VBR) encoding with `ffmpeg`:

```
ffmpeg -i INPUT -codec:v libtheora -q:v 10 OUTPUT.ogg
```

- Use `ffmpeg` to convert a CBR 1000 kbps Theora video stream:

```
ffmpeg -i INPUT -codec:v libtheora -b:v 1000k OUTPUT.ogg
```

## 17.4 libvpx# TOC

VP8/VP9 format supported through `libvpx`.

Requires the presence of the `libvpx` headers and library during configuration. You need to explicitly configure the build with `--enable-libvpx`.

### 17.4.1 Options# TOC

The following options are supported by the `libvpx` wrapper. The `vpxenc`-equivalent options or values are listed in parentheses for easy migration.

To reduce the duplication of documentation, only the private options and some others requiring special attention are documented here. For the documentation of the undocumented generic options, see the `Codec Options` chapter.

To get more documentation of the `libvpx` options, invoke the command `ffmpeg -h encoder=libvpx`, `ffmpeg -h encoder=libvpx-vp9` or `vpxenc --help`. Further information is available in the `libvpx` API documentation.

`b` (*target-bitrate*)

Set bitrate in bits/s. Note that `FFmpeg`'s `b` option is expressed in bits/s, while `vpxenc`'s `target-bitrate` is in kilobits/s.

`g` (*kf-max-dist*)

`keyint_min` (*kf-min-dist*)

`qmin` (*min-q*)

`qmax` (*max-q*)

`bufsize` (*buf-sz, buf-optimal-sz*)

Set `ratecontrol` buffer size (in bits). Note `vpxenc`'s options are specified in milliseconds, the `libvpx` wrapper converts this value as follows: `buf-sz = bufsize * 1000 / bitrate`,  
`buf-optimal-sz = bufsize * 1000 / bitrate * 5 / 6`.

`rc_init_occupancy` (*buf-initial-sz*)

Set number of bits which should be loaded into the rc buffer before decoding starts. Note vpxenc's option is specified in milliseconds, the libvpx wrapper converts this value as follows:

$rc\_init\_occupancy * 1000 / \text{bitrate}$ .

`undershoot-pct`

Set datarate undershoot (min) percentage of the target bitrate.

`overshoot-pct`

Set datarate overshoot (max) percentage of the target bitrate.

`skip_threshold (drop-frame)`

`qcomp (bias-pct)`

`maxrate (maxsection-pct)`

Set GOP max bitrate in bits/s. Note vpxenc's option is specified as a percentage of the target bitrate, the libvpx wrapper converts this value as follows:  $(\text{maxrate} * 100 / \text{bitrate})$ .

`minrate (minsection-pct)`

Set GOP min bitrate in bits/s. Note vpxenc's option is specified as a percentage of the target bitrate, the libvpx wrapper converts this value as follows:  $(\text{minrate} * 100 / \text{bitrate})$ .

`minrate, maxrate, b end-usage=cbr`

$(\text{minrate} == \text{maxrate} == \text{bitrate})$ .

`crf (end-usage=cq, cq-level)`

`quality, deadline (deadline)`

`'best'`

Use best quality deadline. Poorly named and quite slow, this option should be avoided as it may give worse quality output than good.

`'good'`

Use good quality deadline. This is a good trade-off between speed and quality when used with the `cpu-used` option.

`'realtime'`

Use realtime quality deadline.

`speed, cpu-used (cpu-used)`

Set quality/speed ratio modifier. Higher values speed up the encode at the cost of quality.

`nr` (*noise-sensitivity*)  
`static-thresh`

Set a change threshold on blocks below which they will be skipped by the encoder.

`slices` (*token-parts*)

Note that FFmpeg's `slices` option gives the total number of partitions, while vpxenc's `token-parts` is given as  $\log_2(\text{partitions})$ .

`max-intra-rate`

Set maximum I-frame bitrate as a percentage of the target bitrate. A value of 0 means unlimited.

`force_key_frames`

`VPX_EFLAG_FORCE_KF`

Alternate reference frame related  
`auto-alt-ref`

Enable use of alternate reference frames (2-pass only).

`arnr-max-frames`

Set altref noise reduction max frame count.

`arnr-type`

Set altref noise reduction filter type: backward, forward, centered.

`arnr-strength`

Set altref noise reduction filter strength.

`rc-lookahead, lag-in-frames` (*lag-in-frames*)

Set number of frames to look ahead for frametype and ratecontrol.

`error-resilient`

Enable error resiliency features.

VP9-specific options  
`lossless`

Enable lossless mode.



`tile-columns`

Set number of tile columns to use. Note this is given as `log2(tile_columns)`. For example, 8 tile columns would be requested by setting the `tile-columns` option to 3.

`tile-rows`

Set number of tile rows to use. Note this is given as `log2(tile_rows)`. For example, 4 tile rows would be requested by setting the `tile-rows` option to 2.

`frame-parallel`

Enable frame parallel decodability features.

`aq-mode`

Set adaptive quantization mode (0: off (default), 1: variance 2: complexity, 3: cyclic refresh).

`colorspace` *color-space*

Set input color space. The VP9 bitstream supports signaling the following colorspace:

```
'rgb' sRGB
'bt709' bt709
'unspecified' unknown
'bt470bg' bt601
'smpte170m' smpte170
'smpte240m' smpte240
'bt2020_ncl' bt2020
```

For more information about libvpx see: <http://www.webmproject.org/>

## 17.5 libwebp# TOC

libwebp WebP Image encoder wrapper

libwebp is Google's official encoder for WebP images. It can encode in either lossy or lossless mode. Lossy images are essentially a wrapper around a VP8 frame. Lossless images are a separate codec developed by Google.

### 17.5.1 Pixel Format# TOC

Currently, libwebp only supports YUV420 for lossy and RGB for lossless due to limitations of the format and libwebp. Alpha is supported for either mode. Because of API limitations, if RGB is passed in when encoding lossy or YUV is passed in for encoding lossless, the pixel format will automatically be converted using functions from libwebp. This is not ideal and is done only for convenience.

## 17.5.2 Options# TOC

`-lossless boolean`

Enables/Disables use of lossless mode. Default is 0.

`-compression_level integer`

For lossy, this is a quality/speed tradeoff. Higher values give better quality for a given size at the cost of increased encoding time. For lossless, this is a size/speed tradeoff. Higher values give smaller size at the cost of increased encoding time. More specifically, it controls the number of extra algorithms and compression tools used, and varies the combination of these tools. This maps to the *method* option in libwebp. The valid range is 0 to 6. Default is 4.

`-qscale float`

For lossy encoding, this controls image quality, 0 to 100. For lossless encoding, this controls the effort and time spent at compressing more. The default value is 75. Note that for usage via libavcodec, this option is called *global\_quality* and must be multiplied by *FF\_QP2LAMBDA*.

`-preset type`

Configuration preset. This does some automatic settings based on the general type of the image.

`none`

Do not use a preset.

`default`

Use the encoder default.

`picture`

Digital picture, like portrait, inner shot

`photo`

Outdoor photograph, with natural lighting

`drawing`

Hand or line drawing, with high-contrast details

`icon`

Small-sized colorful images

text

Text-like

## 17.6 libx264, libx264rgb# TOC

x264 H.264/MPEG-4 AVC encoder wrapper.

This encoder requires the presence of the libx264 headers and library during configuration. You need to explicitly configure the build with `--enable-libx264`.

libx264 supports an impressive number of features, including 8x8 and 4x4 adaptive spatial transform, adaptive B-frame placement, CAVLC/CABAC entropy coding, interlacing (MBAFF), lossless mode, psy optimizations for detail retention (adaptive quantization, psy-RD, psy-trellis).

Many libx264 encoder options are mapped to FFmpeg global codec options, while unique encoder options are provided through private options. Additionally the `x264opts` and `x264-params` private options allows one to pass a list of key=value tuples as accepted by the libx264 `x264_param_parse` function.

The x264 project website is at <http://www.videolan.org/developers/x264.html>.

The libx264rgb encoder is the same as libx264, except it accepts packed RGB pixel formats as input instead of YUV.

### 17.6.1 Supported Pixel Formats# TOC

x264 supports 8- to 10-bit color spaces. The exact bit depth is controlled at x264's configure time. FFmpeg only supports one bit depth in one particular build. In other words, it is not possible to build one FFmpeg with multiple versions of x264 with different bit depths.

### 17.6.2 Options# TOC

The following options are supported by the libx264 wrapper. The x264-equivalent options or values are listed in parentheses for easy migration.

To reduce the duplication of documentation, only the private options and some others requiring special attention are documented here. For the documentation of the undocumented generic options, see the Codec Options chapter.

To get a more accurate and extensive documentation of the libx264 options, invoke the command `x264 --full-help` or consult the libx264 documentation.

`b` (*bitrate*)

Set bitrate in bits/s. Note that FFmpeg's `b` option is expressed in bits/s, while x264's `bitrate` is in kilobits/s.

`bf (bframes)`  
`g (keyint)`  
`qmin (qpmin)`

Minimum quantizer scale.

`qmax (qpmax)`

Maximum quantizer scale.

`qdiff (qpstep)`

Maximum difference between quantizer scales.

`qblur (qblur)`

Quantizer curve blur

`qcomp (qcomp)`

Quantizer curve compression factor

`refs (ref)`

Number of reference frames each P-frame can use. The range is from 0-16.

`sc_threshold (scenecut)`

Sets the threshold for the scene change detection.

`trellis (trellis)`

Performs Trellis quantization to increase efficiency. Enabled by default.

`nr (nr)`

`me_range (merange)`

Maximum range of the motion search in pixels.

`me_method (me)`

Set motion estimation method. Possible values in the decreasing order of speed:

`'dia (dia)'`  
`'epzs (dia)'`

Diamond search with radius 1 (fastest). `'epzs'` is an alias for `'dia'`.

`'hex (hex)'`

Hexagonal search with radius 2.

`'umh (umh)'`

Uneven multi-hexagon search.

`'esa (esa)'`

Exhaustive search.

`'tesa (tesa)'`

Hadamard exhaustive search (slowest).

`subq (subme)`

Sub-pixel motion estimation method.

`b_strategy (b-adapt)`

Adaptive B-frame placement decision algorithm. Use only on first-pass.

`keyint_min (min-keyint)`

Minimum GOP size.

`coder`

Set entropy encoder. Possible values:

`'ac'`

Enable CABAC.

`'vlc'`

Enable CAVLC and disable CABAC. It generates the same effect as x264's `--no-cabac` option.

`cmp`

Set full pixel motion estimation comparison algorithm. Possible values:

`'chroma'`

Enable chroma in motion estimation.

`'sad'`

Ignore chroma in motion estimation. It generates the same effect as x264's `--no-chroma-me` option.

`threads` (*threads*)

Number of encoding threads.

`thread_type`

Set multithreading technique. Possible values:

`'slice'`

Slice-based multithreading. It generates the same effect as x264's `--sliced-threads` option.

`'frame'`

Frame-based multithreading.

`flags`

Set encoding flags. It can be used to disable closed GOP and enable open GOP by setting it to `-cgop`. The result is similar to the behavior of x264's `--open-gop` option.

`rc_init_occupancy` (*vbv-init*)

`preset` (*preset*)

Set the encoding preset.

`tune` (*tune*)

Set tuning of the encoding params.

`profile` (*profile*)

Set profile restrictions.

`fastfirstpass`

Enable fast settings when encoding first pass, when set to 1. When set to 0, it has the same effect of x264's `--slow-firstpass` option.

`crf` (*crf*)

Set the quality for constant quality mode.

`crf_max (crf-max)`

In CRF mode, prevents VBV from lowering quality beyond this point.

`qp (qp)`

Set constant quantization rate control method parameter.

`aq-mode (aq-mode)`

Set AQ method. Possible values:

`'none (0)'`

Disabled.

`'variance (1)'`

Variance AQ (complexity mask).

`'autovariance (2)'`

Auto-variance AQ (experimental).

`aq-strength (aq-strength)`

Set AQ strength, reduce blocking and blurring in flat and textured areas.

`psy`

Use psychovisual optimizations when set to 1. When set to 0, it has the same effect as x264's `--no-psy` option.

`psy-rd (psy-rd)`

Set strength of psychovisual optimization, in *psy-rd:psy-trellis* format.

`rc-lookahead (rc-lookahead)`

Set number of frames to look ahead for frametype and ratecontrol.

`weightb`

Enable weighted prediction for B-frames when set to 1. When set to 0, it has the same effect as x264's `--no-weightb` option.

`weightp (weightp)`

Set weighted prediction method for P-frames. Possible values:

`'none (0)'`

Disabled

`'simple (1)'`

Enable only weighted refs

`'smart (2)'`

Enable both weighted refs and duplicates

`ssim (ssim)`

Enable calculation and printing SSIM stats after the encoding.

`intra-refresh (intra-refresh)`

Enable the use of Periodic Intra Refresh instead of IDR frames when set to 1.

`avcintra-class (class)`

Configure the encoder to generate AVC-Intra. Valid values are 50,100 and 200

`bluray-compat (bluray-compat)`

Configure the encoder to be compatible with the bluray standard. It is a shorthand for setting "bluray-compat=1 force-cfr=1".

`b-bias (b-bias)`

Set the influence on how often B-frames are used.

`b-pyramid (b-pyramid)`

Set method for keeping of some B-frames as references. Possible values:

`'none (none)'`

Disabled.

`'strict (strict)'`

Strictly hierarchical pyramid.

`'normal (normal)'`



Non-strict (not Blu-ray compatible).

`mixed-refs`

Enable the use of one reference per partition, as opposed to one reference per macroblock when set to 1. When set to 0, it has the same effect as x264's `--no-mixed-refs` option.

`8x8dct`

Enable adaptive spatial transform (high profile 8x8 transform) when set to 1. When set to 0, it has the same effect as x264's `--no-8x8dct` option.

`fast-pskip`

Enable early SKIP detection on P-frames when set to 1. When set to 0, it has the same effect as x264's `--no-fast-pskip` option.

`aud` (*aud*)

Enable use of access unit delimiters when set to 1.

`mbtree`

Enable use macroblock tree ratecontrol when set to 1. When set to 0, it has the same effect as x264's `--no-mbtree` option.

`deblock` (*deblock*)

Set loop filter parameters, in *alpha:beta* form.

`cplxblur` (*cplxblur*)

Set fluctuations reduction in QP (before curve compression).

`partitions` (*partitions*)

Set partitions to consider as a comma-separated list of. Possible values in the list:

`'p8x8'`

8x8 P-frame partition.

`'p4x4'`

4x4 P-frame partition.

`'b8x8'`

4x4 B-frame partition.

`'i8x8'`

8x8 I-frame partition.

`'i4x4'`

4x4 I-frame partition. (Enabling `'p4x4'` requires `'p8x8'` to be enabled. Enabling `'i8x8'` requires adaptive spatial transform (`8x8dct` option) to be enabled.)

`'none (none)'`

Do not consider any partitions.

`'all (all)'`

Consider every partition.

`direct-pred (direct)`

Set direct MV prediction mode. Possible values:

`'none (none)'`

Disable MV prediction.

`'spatial (spatial)'`

Enable spatial predicting.

`'temporal (temporal)'`

Enable temporal predicting.

`'auto (auto)'`

Automatically decided.

`slice-max-size (slice-max-size)`

Set the limit of the size of each slice in bytes. If not specified but RTP payload size (`ps`) is specified, that is used.

`stats (stats)`

Set the file name for multi-pass stats.

`nal-hrd` (*nal-hrd*)

Set signal HRD information (requires `vbv-bufsize` to be set). Possible values:

`'none (none)'`

Disable HRD information signaling.

`'vbr (vbr)'`

Variable bit rate.

`'cbr (cbr)'`

Constant bit rate (not allowed in MP4 container).

`x264opts` (*N.A.*)

Set any x264 option, see `x264 --fullhelp` for a list.

Argument is a list of *key=value* couples separated by ":". In *filter* and *psy-rd* options that use ":" as a separator themselves, use "," instead. They accept it as well since long ago but this is kept undocumented for some reason.

For example to specify libx264 encoding options with `ffmpeg`:

```
ffmpeg -i foo.mpg -vcodec libx264 -x264opts keyint=123:min-keyint=20 -an out.mkv
```

`x264-params` (*N.A.*)

Override the x264 configuration using a `:`-separated list of *key=value* parameters.

This option is functionally the same as the `x264opts`, but is duplicated for compatibility with the Libav fork.

For example to specify libx264 encoding options with `ffmpeg`:

```
ffmpeg -i INPUT -c:v libx264 -x264-params level=30:bframes=0:weightp=0:\
cabac=0:ref=1:vbv-maxrate=768:vbv-bufsize=2000:analyse=all:me=umh:\
no-fast-pskip=1:subq=6:8x8dct=0:trellis=0 OUTPUT
```

Encoding `ffpresets` for common usages are provided so they can be used with the general presets system (e.g. passing the `pre` option).

## 17.7 libx265# TOC

x265 H.265/HEVC encoder wrapper.

This encoder requires the presence of the libx265 headers and library during configuration. You need to explicitly configure the build with `--enable-libx265`.

### 17.7.1 Options# TOC

`preset`

Set the x265 preset.

`tune`

Set the x265 tune parameter.

`x265-params`

Set x265 options using a list of *key=value* couples separated by ":". See `x265 --help` for a list of options.

For example to specify libx265 encoding options with `-x265-params`:

```
ffmpeg -i input -c:v libx265 -x265-params crf=26:psy-rd=1 output.mp4
```

## 17.8 libxvid# TOC

Xvid MPEG-4 Part 2 encoder wrapper.

This encoder requires the presence of the libxvidcore headers and library during configuration. You need to explicitly configure the build with `--enable-libxvid --enable-gpl`.

The native `mpeg4` encoder supports the MPEG-4 Part 2 format, so users can encode to this format without this library.

### 17.8.1 Options# TOC

The following options are supported by the libxvid wrapper. Some of the following options are listed but are not documented, and correspond to shared codec options. See the Codec Options chapter for their documentation. The other shared options which are not listed have no effect for the libxvid encoder.

`b`

`g`

`qmin`

`qmax`

`mpeg_quant`

`threads`

`bf`

`b_qfactor`

`b_qoffset`

flags

Set specific encoding flags. Possible values:

‘mv4’

Use four motion vector by macroblock.

‘aic’

Enable high quality AC prediction.

‘gray’

Only encode grayscale.

‘gmc’

Enable the use of global motion compensation (GMC).

‘qpel’

Enable quarter-pixel motion compensation.

‘cgop’

Enable closed GOP.

‘global\_header’

Place global headers in extradata instead of every keyframe.

trellis

me\_method

Set motion estimation method. Possible values in decreasing order of speed and increasing order of quality:

‘zero’

Use no motion estimation (default).

‘phods’

‘x1’

‘log’

Enable advanced diamond zonal search for 16x16 blocks and half-pixel refinement for 16x16 blocks. ‘x1’ and ‘log’ are aliases for ‘phods’.

`'epzs'`

Enable all of the things described above, plus advanced diamond zonal search for 8x8 blocks, half-pixel refinement for 8x8 blocks, and motion estimation on chroma planes.

`'full'`

Enable all of the things described above, plus extended 16x16 and 8x8 blocks search.

`mbd`

Set macroblock decision algorithm. Possible values in the increasing order of quality:

`'simple'`

Use macroblock comparing function algorithm (default).

`'bits'`

Enable rate distortion-based half pixel and quarter pixel refinement for 16x16 blocks.

`'rd'`

Enable all of the things described above, plus rate distortion-based half pixel and quarter pixel refinement for 8x8 blocks, and rate distortion-based search using square pattern.

`lumi_aq`

Enable lumi masking adaptive quantization when set to 1. Default is 0 (disabled).

`variance_aq`

Enable variance adaptive quantization when set to 1. Default is 0 (disabled).

When combined with `lumi_aq`, the resulting quality will not be better than any of the two specified individually. In other words, the resulting quality will be the worse one of the two effects.

`ssim`

Set structural similarity (SSIM) displaying method. Possible values:

`'off'`

Disable displaying of SSIM information.

`'avg'`

Output average SSIM at the end of encoding to stdout. The format of showing the average SSIM is:

Average SSIM: %f

For users who are not familiar with C, %f means a float number, or a decimal (e.g. 0.939232).

‘frame’

Output both per-frame SSIM data during encoding and average SSIM at the end of encoding to stdout. The format of per-frame information is:

SSIM: avg: %1.3f min: %1.3f max: %1.3f

For users who are not familiar with C, %1.3f means a float number rounded to 3 digits after the dot (e.g. 0.932).

ssim\_acc

Set SSIM accuracy. Valid options are integers within the range of 0-4, while 0 gives the most accurate result and 4 computes the fastest.

## 17.9 mpeg2# TOC

MPEG-2 video encoder.

### 17.9.1 Options# TOC

seq\_disp\_ext *integer*

Specifies if the encoder should write a sequence\_display\_extension to the output.

-1

auto

Decide automatically to write it or not (this is the default) by checking if the data to be written is different from the default or unspecified values.

0

never

Never write it.

1

always

Always write it.

## 17.10 png# TOC

PNG image encoder.

### 17.10.1 Private options# TOC

`dpi integer`

Set physical density of pixels, in dots per inch, unset by default

`dpm integer`

Set physical density of pixels, in dots per meter, unset by default

## 17.11 ProRes# TOC

Apple ProRes encoder.

FFmpeg contains 2 ProRes encoders, the `prores-aw` and `prores-ks` encoder. The used encoder can be chosen with the `-vcodec` option.

### 17.11.1 Private Options for prores-ks# TOC

`profile integer`

Select the ProRes profile to encode

`'proxy'`  
`'lt'`  
`'standard'`  
`'hq'`  
`'4444'`

`quant_mat integer`

Select quantization matrix.

`'auto'`  
`'default'`  
`'proxy'`  
`'lt'`  
`'standard'`  
`'hq'`

If set to *auto*, the matrix matching the profile will be picked. If not set, the matrix providing the highest quality, *default*, will be picked.



`bits_per_mb integer`

How many bits to allot for coding one macroblock. Different profiles use between 200 and 2400 bits per macroblock, the maximum is 8000.

`mbs_per_slice integer`

Number of macroblocks in each slice (1-8); the default value (8) should be good in almost all situations.

`vendor string`

Override the 4-byte vendor ID. A custom vendor ID like *apl0* would claim the stream was produced by the Apple encoder.

`alpha_bits integer`

Specify number of bits for alpha component. Possible values are 0, 8 and 16. Use 0 to disable alpha plane coding.

### 17.11.2 Speed considerations# TOC

In the default mode of operation the encoder has to honor frame constraints (i.e. not produce frames with size bigger than requested) while still making output picture as good as possible. A frame containing a lot of small details is harder to compress and the encoder would spend more time searching for appropriate quantizers for each slice.

Setting a higher `bits_per_mb` limit will improve the speed.

For the fastest encoding speed set the `qscale` parameter (4 is the recommended value) and do not set a size constraint.

### 17.12 libkvazaar# TOC

Kvazaar H.265/HEVC encoder.

Requires the presence of the libkvazaar headers and library during configuration. You need to explicitly configure the build with `--enable-libkvazaar`.

#### 17.12.1 Options# TOC

`b`

Set target video bitrate in bit/s and enable rate control.

`threads`

Set number of encoding threads.

`kvazaar-params`

Set kvazaar parameters as a list of *name=value* pairs separated by commas (.). See kvazaar documentation for a list of options.

## 18 Subtitles Encoders# TOC

### 18.1 dvdsub# TOC

This codec encodes the bitmap subtitle format that is used in DVDs. Typically they are stored in VOBSUB file pairs (\*.idx + \*.sub), and they can also be used in Matroska files.

#### 18.1.1 Options# TOC

`even_rows_fix`

When set to 1, enable a work-around that makes the number of pixel rows even in all subtitles. This fixes a problem with some players that cut off the bottom row if the number is odd. The work-around just adds a fully transparent row if needed. The overhead is low, typically one byte per subtitle on average.

By default, this work-around is disabled.

## 19 Bitstream Filters# TOC

When you configure your FFmpeg build, all the supported bitstream filters are enabled by default. You can list all available ones using the configure option `--list-bsfs`.

You can disable all the bitstream filters using the configure option `--disable-bsfs`, and selectively enable any bitstream filter using the option `--enable-bsf=BSF`, or you can disable a particular bitstream filter using the option `--disable-bsf=BSF`.

The option `-bsfs` of the ff\* tools will display the list of all the supported bitstream filters included in your build.

The ff\* tools have a `-bsf` option applied per stream, taking a comma-separated list of filters, whose parameters follow the filter name after a '='.

```
ffmpeg -i INPUT -c:v copy -bsf:v filter1[=opt1=str1/opt2=str2][,filter2] OUTPUT
```

Below is a description of the currently available bitstream filters, with their parameters, if any.

## 19.1 aac\_adtstoasc# TOC

Convert MPEG-2/4 AAC ADTS to MPEG-4 Audio Specific Configuration bitstream filter.

This filter creates an MPEG-4 AudioSpecificConfig from an MPEG-2/4 ADTS header and removes the ADTS header.

This is required for example when copying an AAC stream from a raw ADTS AAC container to a FLV or a MOV/MP4 file.

## 19.2 chomp# TOC

Remove zero padding at the end of a packet.

## 19.3 dump\_extra# TOC

Add extradata to the beginning of the filtered packets.

The additional argument specifies which packets should be filtered. It accepts the values:

‘a’

add extradata to all key packets, but only if *local\_header* is set in the *flags2* codec context field

‘k’

add extradata to all key packets

‘e’

add extradata to all packets

If not specified it is assumed ‘k’.

For example the following `ffmpeg` command forces a global header (thus disabling individual packet headers) in the H.264 packets generated by the `libx264` encoder, but corrects them by adding the header stored in extradata to the key packets:

```
ffmpeg -i INPUT -map 0 -flags:v +global_header -c:v libx264 -bsf:v dump_extra out.ts
```

## 19.4 h264\_mp4toannexb# TOC

Convert an H.264 bitstream from length prefixed mode to start code prefixed mode (as defined in the Annex B of the ITU-T H.264 specification).

This is required by some streaming formats, typically the MPEG-2 transport stream format ("mpegts").

For example to remux an MP4 file containing an H.264 stream to mpegts format with `ffmpeg`, you can use the command:

```
ffmpeg -i INPUT.mp4 -codec copy -bsf:v h264_mp4toannexb OUTPUT.ts
```

## 19.5 imxdump# TOC

Modifies the bitstream to fit in MOV and to be usable by the Final Cut Pro decoder. This filter only applies to the `mpeg2video` codec, and is likely not needed for Final Cut Pro 7 and newer with the appropriate `-tag:v`.

For example, to remux 30 MB/sec NTSC IMX to MOV:

```
ffmpeg -i input.mxf -c copy -bsf:v imxdump -tag:v mx3n output.mov
```

## 19.6 mjpeg2jpeg# TOC

Convert MJPEG/AVI1 packets to full JPEG/JFIF packets.

MJPEG is a video codec wherein each video frame is essentially a JPEG image. The individual frames can be extracted without loss, e.g. by

```
ffmpeg -i ../some_mjpeg.avi -c:v copy frames_%d.jpg
```

Unfortunately, these chunks are incomplete JPEG images, because they lack the DHT segment required for decoding. Quoting from <http://www.digitalpreservation.gov/formats/fdd/fdd000063.shtml>:

Avery Lee, writing in the `rec.video.desktop` newsgroup in 2001, commented that "MJPEG, or at least the MJPEG in AVIs having the MJPG fourcc, is restricted JPEG with a fixed – and \*omitted\* – Huffman table. The JPEG must be YCbCr colorspace, it must be 4:2:2, and it must use basic Huffman encoding, not arithmetic or progressive. . . . You can indeed extract the MJPEG frames and decode them with a regular JPEG decoder, but you have to prepend the DHT segment to them, or else the decoder won't have any idea how to decompress the data. The exact table necessary is given in the OpenDML spec."

This bitstream filter patches the header of frames extracted from an MJPEG stream (carrying the AVI1 header ID and lacking a DHT segment) to produce fully qualified JPEG images.

```
ffmpeg -i mjpeg-movie.avi -c:v copy -bsf:v mjpeg2jpeg frame_%d.jpg
exiftran -i -9 frame*.jpg
ffmpeg -i frame_%d.jpg -c:v copy rotated.avi
```

## 19.7 mjpega\_dump\_header# TOC

## 19.8 movsub# TOC

## 19.9 mp3\_header\_decompress# TOC

## 19.10 mpeg4\_unpack\_bframes# TOC

Unpack DivX-style packed B-frames.

DivX-style packed B-frames are not valid MPEG-4 and were only a workaround for the broken Video for Windows subsystem. They use more space, can cause minor AV sync issues, require more CPU power to decode (unless the player has some decoded picture queue to compensate the 2,0,2,0 frame per packet style) and cause trouble if copied into a standard container like mp4 or mpeg-ps/ts, because MPEG-4 decoders may not be able to decode them, since they are not valid MPEG-4.

For example to fix an AVI file containing an MPEG-4 stream with DivX-style packed B-frames using `ffmpeg`, you can use the command:

```
ffmpeg -i INPUT.avi -codec copy -bsf:v mpeg4_unpack_bframes OUTPUT.avi
```

## 19.11 noise# TOC

Damages the contents of packets without damaging the container. Can be used for fuzzing or testing error resilience/concealment.

Parameters: A numeral string, whose value is related to how often output bytes will be modified. Therefore, values below or equal to 0 are forbidden, and the lower the more frequent bytes will be modified, with 1 meaning every byte is modified.

```
ffmpeg -i INPUT -c copy -bsf noise[=1] output.mkv
```

applies the modification to every byte.

## 19.12 remove\_extra# TOC

## 20 Format Options# TOC

The `libavformat` library provides some generic global options, which can be set on all the muxers and demuxers. In addition each muxer or demuxer may support so-called private options, which are specific for that component.

Options may be set by specifying *-option value* in the `FFmpeg` tools, or by setting the value explicitly in the `AVFormatContext` options or using the `libavutil/opt.h` API for programmatic use.

The list of supported options follows:

```
avioflags flags (input/output)
```

Possible values:

‘direct’

Reduce buffering.

`probesize integer (input)`

Set probing size in bytes, i.e. the size of the data to analyze to get stream information. A higher value will enable detecting more information in case it is dispersed into the stream, but will increase latency. Must be an integer not lesser than 32. It is 5000000 by default.

`packetsize integer (output)`

Set packet size.

`fflags flags (input/output)`

Set format flags.

Possible values:

‘ignidx’

Ignore index.

‘fastseek’

Enable fast, but inaccurate seeks for some formats.

‘genpts’

Generate PTS.

‘nofillin’

Do not fill in missing values that can be exactly calculated.

‘noparse’

Disable AVParsers, this needs +nofillin too.

‘igndts’

Ignore DTS.

‘discardcorrupt’

Discard corrupted frames.

`'sortdts'`

Try to interleave output packets by DTS.

`'keepside'`

Do not merge side data.

`'latm'`

Enable RTP MP4A-LATM payload.

`'nobuffer'`

Reduce the latency introduced by optional buffering

`'bitexact'`

Only write platform-, build- and time-independent data. This ensures that file and data checksums are reproducible and match between platforms. Its primary use is for regression testing.

`seek2any integer (input)`

Allow seeking to non-keyframes on demuxer level when supported if set to 1. Default is 0.

`analyzeduration integer (input)`

Specify how many microseconds are analyzed to probe the input. A higher value will enable detecting more accurate information, but will increase latency. It defaults to 5,000,000 microseconds = 5 seconds.

`cryptokey hexadecimal string (input)`

Set decryption key.

`indexmem integer (input)`

Set max memory used for timestamp index (per stream).

`rtbufsize integer (input)`

Set max memory used for buffering real-time frames.

`fdebug flags (input/output)`

Print specific debug info.

Possible values:

‘ts’

`max_delay integer (input/output)`

Set maximum muxing or demuxing delay in microseconds.

`fpsprobesize integer (input)`

Set number of frames used to probe fps.

`audio_preload integer (output)`

Set microseconds by which audio packets should be interleaved earlier.

`chunk_duration integer (output)`

Set microseconds for each chunk.

`chunk_size integer (output)`

Set size in bytes for each chunk.

`err_detect, f_err_detect flags (input)`

Set error detection flags. `f_err_detect` is deprecated and should be used only via the `ffmpeg` tool.

Possible values:

‘crccheck’

Verify embedded CRCs.

‘bitstream’

Detect bitstream specification deviations.

‘buffer’

Detect improper bitstream length.

‘explode’

Abort decoding on minor error detection.



‘careful’

Consider things that violate the spec and have not been seen in the wild as errors.

‘compliant’

Consider all spec non compliances as errors.

‘aggressive’

Consider things that a sane encoder should not do as an error.

`max_interleave_delta integer (output)`

Set maximum buffering duration for interleaving. The duration is expressed in microseconds, and defaults to 1000000 (1 second).

To ensure all the streams are interleaved correctly, libavformat will wait until it has at least one packet for each stream before actually writing any packets to the output file. When some streams are "sparse" (i.e. there are large gaps between successive packets), this can result in excessive buffering.

This field specifies the maximum difference between the timestamps of the first and the last packet in the muxing queue, above which libavformat will output a packet regardless of whether it has queued a packet for all the streams.

If set to 0, libavformat will continue buffering packets until it has a packet for each stream, regardless of the maximum timestamp difference between the buffered packets.

`use_wallclock_as_timestamps integer (input)`

Use wallclock as timestamps.

`avoid_negative_ts integer (output)`

Possible values:

‘make\_non\_negative’

Shift timestamps to make them non-negative. Also note that this affects only leading negative timestamps, and not non-monotonic negative timestamps.

‘make\_zero’

Shift timestamps so that the first timestamp is 0.

‘auto (default)’

Enables shifting when required by the target format.

`'disabled'`

Disables shifting of timestamp.

When shifting is enabled, all output timestamps are shifted by the same amount. Audio, video, and subtitles desynching and relative timestamp differences are preserved compared to how they would have been without shifting.

`skip_initial_bytes integer (input)`

Set number of bytes to skip before reading header and frames if set to 1. Default is 0.

`correct_ts_overflow integer (input)`

Correct single timestamp overflows if set to 1. Default is 1.

`flush_packets integer (output)`

Flush the underlying I/O stream after each packet. Default 1 enables it, and has the effect of reducing the latency; 0 disables it and may slightly increase performance in some cases.

`output_ts_offset offset (output)`

Set the output time offset.

*offset* must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

The offset is added by the muxer to the output timestamps.

Specifying a positive offset means that the corresponding streams are delayed by the time duration specified in *offset*. Default value is 0 (meaning that no offset is applied).

`format_whitelist list (input)`

"," separated List of allowed demuxers. By default all are allowed.

`dump_separator string (input)`

Separator used to separate the fields printed on the command line about the Stream parameters. For example to separate the fields with newlines and indentation:

```
ffprobe -dump_separator "
    " -i ~/videos/matrixbench_mpeg2.mpg
```

## 20.1 Format stream specifiers# TOC

Format stream specifiers allow selection of one or more streams that match specific properties.

Possible forms of stream specifiers are:

*stream\_index*

Matches the stream with this index.

*stream\_type*[ :*stream\_index*]

*stream\_type* is one of following: 'v' for video, 'a' for audio, 's' for subtitle, 'd' for data, and 't' for attachments. If *stream\_index* is given, then it matches the stream number *stream\_index* of this type. Otherwise, it matches all streams of this type.

*p*:*program\_id*[ :*stream\_index*]

If *stream\_index* is given, then it matches the stream with number *stream\_index* in the program with the id *program\_id*. Otherwise, it matches all streams in the program.

#*stream\_id*

Matches the stream by a format-specific ID.

The exact semantics of stream specifiers is defined by the `avformat_match_stream_specifier()` function declared in the `libavformat/avformat.h` header.

## 21 Demuxers# TOC

Demuxers are configured elements in FFmpeg that can read the multimedia streams from a particular type of file.

When you configure your FFmpeg build, all the supported demuxers are enabled by default. You can list all available ones using the configure option `--list-demuxers`.

You can disable all the demuxers using the configure option `--disable-demuxers`, and selectively enable a single demuxer with the option `--enable-demuxer=DEMUXER`, or disable it with the option `--disable-demuxer=DEMUXER`.

The option `-formats` of the `ff*` tools will display the list of enabled demuxers.

The description of some of the currently available demuxers follows.

## 21.1 aa# TOC

Audible Format 2, 3, and 4 demuxer.

This demuxer is used to demux Audible Format 2, 3, and 4 (.aa) files.

## 21.2 applehttp# TOC

Apple HTTP Live Streaming demuxer.

This demuxer presents all AVStreams from all variant streams. The id field is set to the bitrate variant index number. By setting the discard flags on AVStreams (by pressing 'a' or 'v' in ffplay), the caller can decide which variant streams to actually receive. The total bitrate of the variant that the stream belongs to is available in a metadata key named "variant\_bitrate".

## 21.3 apng# TOC

Animated Portable Network Graphics demuxer.

This demuxer is used to demux APNG files. All headers, but the PNG signature, up to (but not including) the first fcTL chunk are transmitted as extradata. Frames are then split as being all the chunks between two fcTL ones, or between the last fcTL and IEND chunks.

`-ignore_loop bool`

Ignore the loop variable in the file if set.

`-max_fps int`

Maximum framerate in frames per second (0 for no limit).

`-default_fps int`

Default framerate in frames per second when none is specified in the file (0 meaning as fast as possible).

## 21.4 asf# TOC

Advanced Systems Format demuxer.

This demuxer is used to demux ASF files and MMS network streams.

`-no_resync_search bool`

Do not try to resynchronize by looking for a certain optional start code.

## 21.5 concat# TOC

Virtual concatenation script demuxer.

This demuxer reads a list of files and other directives from a text file and demuxes them one after the other, as if all their packet had been muxed together.

The timestamps in the files are adjusted so that the first file starts at 0 and each next file starts where the previous one finishes. Note that it is done globally and may cause gaps if all streams do not have exactly the same length.

All files must have the same streams (same codecs, same time base, etc.).

The duration of each file is used to adjust the timestamps of the next file: if the duration is incorrect (because it was computed using the bit-rate or because the file is truncated, for example), it can cause artifacts. The `duration` directive can be used to override the duration stored in each file.

### 21.5.1 Syntax# TOC

The script is a text file in extended-ASCII, with one directive per line. Empty lines, leading spaces and lines starting with '#' are ignored. The following directive is recognized:

`file path`

Path to a file to read; special characters and spaces must be escaped with backslash or single quotes.

All subsequent file-related directives apply to that file.

`ffconcat version 1.0`

Identify the script type and version. It also sets the `safe` option to 1 if it was to its default -1.

To make FFmpeg recognize the format automatically, this directive must appear exactly as is (no extra space or byte-order-mark) on the very first line of the script.

`duration dur`

Duration of the file. This information can be specified from the file; specifying it here may be more efficient or help if the information from the file is not available or accurate.

If the duration is set for all files, then it is possible to seek in the whole concatenated video.

`inpoint timestamp`

In point of the file. When the demuxer opens the file it instantly seeks to the specified timestamp. Seeking is done so that all streams can be presented successfully at In point.

This directive works best with intra frame codecs, because for non-intra frame ones you will usually get extra packets before the actual In point and the decoded content will most likely contain frames before In point too.

For each file, packets before the file In point will have timestamps less than the calculated start timestamp of the file (negative in case of the first file), and the duration of the files (if not specified by the `duration` directive) will be reduced based on their specified In point.

Because of potential packets before the specified In point, packet timestamps may overlap between two concatenated files.

`outpoint timestamp`

Out point of the file. When the demuxer reaches the specified decoding timestamp in any of the streams, it handles it as an end of file condition and skips the current and all the remaining packets from all streams.

Out point is exclusive, which means that the demuxer will not output packets with a decoding timestamp greater or equal to Out point.

This directive works best with intra frame codecs and formats where all streams are tightly interleaved. For non-intra frame codecs you will usually get additional packets with presentation timestamp after Out point therefore the decoded content will most likely contain frames after Out point too. If your streams are not tightly interleaved you may not get all the packets from all streams before Out point and you may only will be able to decode the earliest stream until Out point.

The duration of the files (if not specified by the `duration` directive) will be reduced based on their specified Out point.

`file_packet_metadata key=value`

Metadata of the packets of the file. The specified metadata will be set for each file packet. You can specify this directive multiple times to add multiple metadata entries.

`stream`

Introduce a stream in the virtual file. All subsequent stream-related directives apply to the last introduced stream. Some streams properties must be set in order to allow identifying the matching streams in the subfiles. If no streams are defined in the script, the streams from the first file are copied.

`exact_stream_id id`

Set the id of the stream. If this directive is given, the string with the corresponding id in the subfiles will be used. This is especially useful for MPEG-PS (VOB) files, where the order of the streams is not reliable.

## 21.5.2 Options# TOC

This demuxer accepts the following option:

`safe`

If set to 1, reject unsafe file paths. A file path is considered safe if it does not contain a protocol specification and is relative and all components only contain characters from the portable character set (letters, digits, period, underscore and hyphen) and have no period at the beginning of a component.

If set to 0, any file name is accepted.

The default is -1, it is equivalent to 1 if the format was automatically probed and 0 otherwise.

`auto_convert`

If set to 1, try to perform automatic conversions on packet data to make the streams concatenable. The default is 1.

Currently, the only conversion is adding the `h264_mp4toannexb` bitstream filter to H.264 streams in MP4 format. This is necessary in particular if there are resolution changes.

## 21.6 flv# TOC

Adobe Flash Video Format demuxer.

This demuxer is used to demux FLV files and RTMP network streams.

`-flv_metadata bool`

Allocate the streams according to the `onMetaData` array content.

## 21.7 libgme# TOC

The Game Music Emu library is a collection of video game music file emulators.

See <http://code.google.com/p/game-music-emu/> for more information.

Some files have multiple tracks. The demuxer will pick the first track by default. The `track_index` option can be used to select a different track. Track indexes start at 0. The demuxer exports the number of tracks as *tracks* meta data entry.

For very large files, the `max_size` option may have to be adjusted.

## 21.8 libquvi# TOC

Play media from Internet services using the quvi project.

The demuxer accepts a `format` option to request a specific quality. It is by default set to *best*.

See <http://quvi.sourceforge.net/> for more information.

FFmpeg needs to be built with `--enable-libquvi` for this demuxer to be enabled.

## 21.9 gif# TOC

Animated GIF demuxer.

It accepts the following options:

`min_delay`

Set the minimum valid delay between frames in hundredths of seconds. Range is 0 to 6000. Default value is 2.

`max_gif_delay`

Set the maximum valid delay between frames in hundredth of seconds. Range is 0 to 65535. Default value is 65535 (nearly eleven minutes), the maximum value allowed by the specification.

`default_delay`

Set the default delay between frames in hundredths of seconds. Range is 0 to 6000. Default value is 10.

`ignore_loop`

GIF files can contain information to loop a certain number of times (or infinitely). If `ignore_loop` is set to 1, then the loop setting from the input will be ignored and looping will not occur. If set to 0, then looping will occur and will cycle the number of times according to the GIF. Default value is 1.

For example, with the overlay filter, place an infinitely looping GIF over another video:

```
ffmpeg -i input.mp4 -ignore_loop 0 -i input.gif -filter_complex overlay=shortest=1 out.mkv
```

Note that in the above example the `shortest` option for overlay filter is used to end the output video at the length of the shortest input file, which in this case is `input.mp4` as the GIF in this example loops infinitely.



## 21.10 image2# TOC

Image file demuxer.

This demuxer reads from a list of image files specified by a pattern. The syntax and meaning of the pattern is specified by the option *pattern\_type*.

The pattern may contain a suffix which is used to automatically determine the format of the images contained in the files.

The size, the pixel format, and the format of each image must be the same for all the files in the sequence.

This demuxer accepts the following options:

`framerate`

Set the frame rate for the video stream. It defaults to 25.

`loop`

If set to 1, loop over the input. Default value is 0.

`pattern_type`

Select the pattern type used to interpret the provided filename.

*pattern\_type* accepts one of the following values.

`none`

Disable pattern matching, therefore the video will only contain the specified image. You should use this option if you do not want to create sequences from multiple images and your filenames may contain special pattern characters.

`sequence`

Select a sequence pattern type, used to specify a sequence of files indexed by sequential numbers.

A sequence pattern may contain the string "%d" or "%0Nd", which specifies the position of the characters representing a sequential number in each filename matched by the pattern. If the form "%d0Nd" is used, the string representing the number in each filename is 0-padded and *N* is the total number of 0-padded digits representing the number. The literal character '%' can be specified in the pattern with the string "%%".

If the sequence pattern contains "%d" or "%0Nd", the first filename of the file list specified by the pattern must contain a number inclusively contained between *start\_number* and *start\_number+start\_number\_range-1*, and all the following numbers must be sequential.

For example the pattern "img-%03d.bmp" will match a sequence of filenames of the form `img-001.bmp`, `img-002.bmp`, ..., `img-010.bmp`, etc.; the pattern "i%%m%%g-%d.jpg" will match a sequence of filenames of the form `i%m%g-1.jpg`, `i%m%g-2.jpg`, ..., `i%m%g-10.jpg`, etc.

Note that the pattern must not necessarily contain "%d" or "%Nd", for example to convert a single image file `img.jpeg` you can employ the command:

```
ffmpeg -i img.jpeg img.png
```

`glob`

Select a glob wildcard pattern type.

The pattern is interpreted like a `glob()` pattern. This is only selectable if libavformat was compiled with globbing support.

`glob_sequence` (*deprecated, will be removed*)

Select a mixed glob wildcard/sequence pattern.

If your version of libavformat was compiled with globbing support, and the provided pattern contains at least one glob meta character among `%*?[ ]{ }` that is preceded by an unescaped "%", the pattern is interpreted like a `glob()` pattern, otherwise it is interpreted like a sequence pattern.

All glob special characters `%*?[ ]{ }` must be prefixed with "%". To escape a literal "%" you shall use "%%".

For example the pattern `foo-%*.jpeg` will match all the filenames prefixed by "foo-" and terminating with ".jpeg", and `foo-%????.jpeg` will match all the filenames prefixed with "foo-", followed by a sequence of three characters, and terminating with ".jpeg".

This pattern type is deprecated in favor of *glob* and *sequence*.

Default value is *glob\_sequence*.

`pixel_format`

Set the pixel format of the images to read. If not specified the pixel format is guessed from the first image file in the sequence.

`start_number`

Set the index of the file matched by the image file pattern to start to read from. Default value is 0.

`start_number_range`

Set the index interval range to check when looking for the first image file in the sequence, starting from *start\_number*. Default value is 5.

`ts_from_file`

If set to 1, will set frame timestamp to modification time of image file. Note that monotonicity of timestamps is not provided: images go in the same order as without this option. Default value is 0. If set to 2, will set frame timestamp to the modification time of the image file in nanosecond precision.

`video_size`

Set the video size of the images to read. If not specified the video size is guessed from the first image file in the sequence.

### 21.10.1 Examples# TOC

- Use `ffmpeg` for creating a video from the images in the file sequence `img-001.jpeg`, `img-002.jpeg`, ..., assuming an input frame rate of 10 frames per second:

```
ffmpeg -framerate 10 -i 'img-%03d.jpeg' out.mkv
```

- As above, but start by reading from a file with index 100 in the sequence:

```
ffmpeg -framerate 10 -start_number 100 -i 'img-%03d.jpeg' out.mkv
```

- Read images matching the `"*.png"` glob pattern, that is all the files terminating with the `".png"` suffix:

```
ffmpeg -framerate 10 -pattern_type glob -i "*.png" out.mkv
```

### 21.11 mpegts# TOC

MPEG-2 transport stream demuxer.

This demuxer accepts the following options:

`resync_size`

Set size limit for looking up a new synchronization. Default value is 65536.

`fix_teletext_pts`

Override teletext packet PTS and DTS values with the timestamps calculated from the PCR of the first program which the teletext stream is part of and is not discarded. Default value is 1, set this option to 0 if you want your teletext packet PTS and DTS values untouched.

`ts_packet_size`

Output option carrying the raw packet size in bytes. Show the detected raw packet size, cannot be set by the user.

`scan_all_pmts`

Scan and combine all PMTs. The value is an integer with value from -1 to 1 (-1 means automatic setting, 1 means enabled, 0 means disabled). Default value is -1.

## 21.12 rawvideo# TOC

Raw video demuxer.

This demuxer allows one to read raw video data. Since there is no header specifying the assumed video parameters, the user must specify them in order to be able to decode the data correctly.

This demuxer accepts the following options:

`framerate`

Set input video frame rate. Default value is 25.

`pixel_format`

Set the input video pixel format. Default value is yuv420p.

`video_size`

Set the input video size. This value must be specified explicitly.

For example to read a rawvideo file `input.raw` with `ffplay`, assuming a pixel format of `rgb24`, a video size of `320x240`, and a frame rate of 10 images per second, use the command:

```
ffplay -f rawvideo -pixel_format rgb24 -video_size 320x240 -framerate 10 input.raw
```

## 21.13 sbg# TOC

SBaGen script demuxer.

This demuxer reads the script language used by SBaGen <http://uazu.net/sbagen/> to generate binaural beats sessions. A SBG script looks like that:

```
-SE
a: 300-2.5/3 440+4.5/0
b: 300-2.5/0 440+4.5/3
off: -
NOW      == a
+0:07:00 == b
+0:14:00 == a
+0:21:00 == b
+0:30:00  off
```

A SBG script can mix absolute and relative timestamps. If the script uses either only absolute timestamps (including the script start time) or only relative ones, then its layout is fixed, and the conversion is straightforward. On the other hand, if the script mixes both kind of timestamps, then the *NOW* reference for relative timestamps will be taken from the current time of day at the time the script is read, and the script layout will be frozen according to that reference. That means that if the script is directly played, the actual times will match the absolute timestamps up to the sound controller's clock accuracy, but if the user somehow pauses the playback or seeks, all times will be shifted accordingly.

## 21.14 tedcaptions# TOC

JSON captions used for TED Talks.

TED does not provide links to the captions, but they can be guessed from the page. The file `tools/bookmarklets.html` from the FFmpeg source tree contains a bookmarklet to expose them.

This demuxer accepts the following option:

`start_time`

Set the start time of the TED talk, in milliseconds. The default is 15000 (15s). It is used to sync the captions with the downloadable videos, because they include a 15s intro.

Example: convert the captions to a format most players understand:

```
ffmpeg -i http://www.ted.com/talks/subtitles/id/1/lang/en talk1-en.srt
```

## 22 Muxers# TOC

Muxers are configured elements in FFmpeg which allow writing multimedia streams to a particular type of file.

When you configure your FFmpeg build, all the supported muxers are enabled by default. You can list all available muxers using the configure option `--list-muxers`.

You can disable all the muxers with the configure option `--disable-muxers` and selectively enable / disable single muxers with the options `--enable-muxer=MUXER` / `--disable-muxer=MUXER`.

The option `-formats` of the ff\* tools will display the list of enabled muxers.

A description of some of the currently available muxers follows.

### 22.1 aiff# TOC

Audio Interchange File Format muxer.

### 22.1.1 Options# TOC

It accepts the following options:

`write_id3v2`

Enable ID3v2 tags writing when set to 1. Default is 0 (disabled).

`id3v2_version`

Select ID3v2 version to write. Currently only version 3 and 4 (aka. ID3v2.3 and ID3v2.4) are supported. The default is version 4.

### 22.2 crc# TOC

CRC (Cyclic Redundancy Check) testing format.

This muxer computes and prints the Adler-32 CRC of all the input audio and video frames. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the CRC.

The output of the muxer consists of a single line of the form: `CRC=0xCRC`, where *CRC* is a hexadecimal number 0-padded to 8 digits containing the CRC for all the decoded input frames.

See also the `framecrc` muxer.

#### 22.2.1 Examples# TOC

For example to compute the CRC of the input, and store it in the file `out.crc`:

```
ffmpeg -i INPUT -f crc out.crc
```

You can print the CRC to stdout with the command:

```
ffmpeg -i INPUT -f crc -
```

You can select the output format of each frame with `ffmpeg` by specifying the audio and video codec and format. For example to compute the CRC of the input audio converted to PCM unsigned 8-bit and the input video converted to MPEG-2 video, use the command:

```
ffmpeg -i INPUT -c:a pcm_u8 -c:v mpeg2video -f crc -
```

### 22.3 framecrc# TOC

Per-packet CRC (Cyclic Redundancy Check) testing format.

This muxer computes and prints the Adler-32 CRC for each audio and video packet. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the CRC.

The output of the muxer consists of a line for each audio and video packet of the form:

*stream\_index, packet\_dts, packet\_pts, packet\_duration, packet\_size, 0xCRC*

*CRC* is a hexadecimal number 0-padded to 8 digits containing the CRC of the packet.

### 22.3.1 Examples# TOC

For example to compute the CRC of the audio and video frames in `INPUT`, converted to raw audio and video packets, and store it in the file `out.crc`:

```
ffmpeg -i INPUT -f framecrc out.crc
```

To print the information to stdout, use the command:

```
ffmpeg -i INPUT -f framecrc -
```

With `ffmpeg`, you can select the output format to which the audio and video frames are encoded before computing the CRC for each packet by specifying the audio and video codec. For example, to compute the CRC of each decoded input audio frame converted to PCM unsigned 8-bit and of each decoded input video frame converted to MPEG-2 video, use the command:

```
ffmpeg -i INPUT -c:a pcm_u8 -c:v mpeg2video -f framecrc -
```

See also the `crc muxer`.

## 22.4 framemd5# TOC

Per-packet MD5 testing format.

This muxer computes and prints the MD5 hash for each audio and video packet. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the hash.

The output of the muxer consists of a line for each audio and video packet of the form:

*stream\_index, packet\_dts, packet\_pts, packet\_duration, packet\_size, MD5*

*MD5* is a hexadecimal number representing the computed MD5 hash for the packet.

### 22.4.1 Examples# TOC

For example to compute the MD5 of the audio and video frames in `INPUT`, converted to raw audio and video packets, and store it in the file `out.md5`:

```
ffmpeg -i INPUT -f framemd5 out.md5
```

To print the information to stdout, use the command:

```
ffmpeg -i INPUT -f framemd5 -
```

See also the md5 muxer.

## 22.5 gif# TOC

Animated GIF muxer.

It accepts the following options:

`loop`

Set the number of times to loop the output. Use `-1` for no loop, 0 for looping indefinitely (default).

`final_delay`

Force the delay (expressed in centiseconds) after the last frame. Each frame ends with a delay until the next frame. The default is `-1`, which is a special value to tell the muxer to re-use the previous delay. In case of a loop, you might want to customize this value to mark a pause for instance.

For example, to encode a gif looping 10 times, with a 5 seconds delay between the loops:

```
ffmpeg -i INPUT -loop 10 -final_delay 500 out.gif
```

Note 1: if you wish to extract the frames in separate GIF files, you need to force the image2 muxer:

```
ffmpeg -i INPUT -c:v gif -f image2 "out%d.gif"
```

Note 2: the GIF format has a very small time base: the delay between two frames can not be smaller than one centi second.

## 22.6 hls# TOC

Apple HTTP Live Streaming muxer that segments MPEG-TS according to the HTTP Live Streaming (HLS) specification.

It creates a playlist file, and one or more segment files. The output filename specifies the playlist filename.

By default, the muxer creates a file for each segment produced. These files have the same name as the playlist, followed by a sequential number and a `.ts` extension.

For example, to convert an input file with `ffmpeg`:

```
ffmpeg -i in.nut out.m3u8
```

This example will produce the playlist, `out.m3u8`, and segment files: `out0.ts`, `out1.ts`, `out2.ts`, etc.



See also the segment muxer, which provides a more generic and flexible implementation of a segmenter, and can be used to perform HLS segmentation.

## 22.6.1 Options# TOC

This muxer supports the following options:

`hls_time` *seconds*

Set the segment length in seconds. Default value is 2.

`hls_list_size` *size*

Set the maximum number of playlist entries. If set to 0 the list file will contain all the segments. Default value is 5.

`hls_ts_options` *options\_list*

Set output format options using a `:`-separated list of key=value parameters. Values containing special characters must be escaped.

`hls_wrap` *wrap*

Set the number after which the segment filename number (the number specified in each segment file) wraps. If set to 0 the number will be never wrapped. Default value is 0.

This option is useful to avoid to fill the disk with many segment files, and limits the maximum number of segment files written to disk to *wrap*.

`start_number` *number*

Start the playlist sequence number from *number*. Default value is 0.

`hls_allow_cache` *allowcache*

Explicitly set whether the client MAY (1) or MUST NOT (0) cache media segments.

`hls_base_url` *baseurl*

Append *baseurl* to every entry in the playlist. Useful to generate playlists with absolute paths.

Note that the playlist sequence number must be unique for each segment and it is not to be confused with the segment filename sequence number which can be cyclic, for example if the `wrap` option is specified.

`hls_segment_filename` *filename*

Set the segment filename. Unless `hls_flags` `single_file` is set *filename* is used as a string format with the segment number:

```
ffmpeg in.nut -hls_segment_filename 'file%03d.ts' out.m3u8
```

This example will produce the playlist, `out.m3u8`, and segment files: `file000.ts`, `file001.ts`, `file002.ts`, etc.

```
hls_key_info_file key_info_file
```

Use the information in *key\_info\_file* for segment encryption. The first line of *key\_info\_file* specifies the key URI written to the playlist. The key URL is used to access the encryption key during playback. The second line specifies the path to the key file used to obtain the key during the encryption process. The key file is read as a single packed array of 16 octets in binary format. The optional third line specifies the initialization vector (IV) as a hexadecimal string to be used instead of the segment sequence number (default) for encryption. Changes to *key\_info\_file* will result in segment encryption with the new key/IV and an entry in the playlist for the new key URI/IV.

Key info file format:

```
key URI  
key file path  
IV (optional)
```

Example key URIs:

```
http://server/file.key  
/path/to/file.key  
file.key
```

Example key file paths:

```
file.key  
/path/to/file.key
```

Example IV:

```
0123456789ABCDEF0123456789ABCDEF
```

Key info file example:

```
http://server/file.key  
/path/to/file.key  
0123456789ABCDEF0123456789ABCDEF
```

Example shell script:

```
#!/bin/sh
BASE_URL=${1:-'.'}
openssl rand 16 > file.key
echo $BASE_URL/file.key > file.keyinfo
echo file.key >> file.keyinfo
echo $(openssl rand -hex 16) >> file.keyinfo
ffmpeg -f lavfi -re -i testsrc -c:v h264 -hls_flags delete_segments \
    -hls_key_info_file file.keyinfo out.m3u8
```

`hls_flags single_file`

If this flag is set, the muxer will store all segments in a single MPEG-TS file, and will use byte ranges in the playlist. HLS playlists generated with this way will have the version number 4. For example:

```
ffmpeg -i in.nut -hls_flags single_file out.m3u8
```

Will produce the playlist, `out.m3u8`, and a single segment file, `out.ts`.

`hls_flags delete_segments`

Segment files removed from the playlist are deleted after a period of time equal to the duration of the segment plus the duration of the playlist.

## 22.7 ico# TOC

ICO file muxer.

Microsoft's icon file format (ICO) has some strict limitations that should be noted:

- Size cannot exceed 256 pixels in any dimension
- Only BMP and PNG images can be stored
- If a BMP image is used, it must be one of the following pixel formats:

BMP Bit Depth	FFmpeg Pixel Format
1bit	pal8
4bit	pal8
8bit	pal8
16bit	rgb555le
24bit	bgr24
32bit	bgra

- If a BMP image is used, it must use the BITMAPINFOHEADER DIB header
- If a PNG image is used, it must use the rgba pixel format

## 22.8 image2# TOC

Image file muxer.

The image file muxer writes video frames to image files.

The output filenames are specified by a pattern, which can be used to produce sequentially numbered series of files. The pattern may contain the string "%d" or "%0Nd", this string specifies the position of the characters representing a numbering in the filenames. If the form "%0Nd" is used, the string representing the number in each filename is 0-padded to *N* digits. The literal character '%' can be specified in the pattern with the string "%%".

If the pattern contains "%d" or "%0Nd", the first filename of the file list specified will contain the number 1, all the following numbers will be sequential.

The pattern may contain a suffix which is used to automatically determine the format of the image files to write.

For example the pattern "img-%03d.bmp" will specify a sequence of filenames of the form img-001.bmp, img-002.bmp, ..., img-010.bmp, etc. The pattern "img%%-%d.jpg" will specify a sequence of filenames of the form img%-1.jpg, img%-2.jpg, ..., img%-10.jpg, etc.

### 22.8.1 Examples# TOC

The following example shows how to use `ffmpeg` for creating a sequence of files `img-001.jpeg`, `img-002.jpeg`, ..., taking one image every second from the input video:

```
ffmpeg -i in.avi -vsync 1 -r 1 -f image2 'img-%03d.jpeg'
```

Note that with `ffmpeg`, if the format is not specified with the `-f` option and the output filename specifies an image file format, the `image2` muxer is automatically selected, so the previous command can be written as:

```
ffmpeg -i in.avi -vsync 1 -r 1 'img-%03d.jpeg'
```

Note also that the pattern must not necessarily contain "%d" or "%0Nd", for example to create a single image file `img.jpeg` from the input video you can employ the command:

```
ffmpeg -i in.avi -f image2 -frames:v 1 img.jpeg
```

The `strftime` option allows you to expand the filename with date and time information. Check the documentation of the `strftime()` function for the syntax.

For example to generate image files from the `strftime()` "%Y-%m-%d\_%H-%M-%S" pattern, the following `ffmpeg` command can be used:

```
ffmpeg -f v4l2 -r 1 -i /dev/video0 -f image2 -strftime 1 "%Y-%m-%d_%H-%M-%S.jpg"
```

### 22.8.2 Options# TOC

`start_number`

Start the sequence from the specified number. Default value is 0.

`update`

If set to 1, the filename will always be interpreted as just a filename, not a pattern, and the corresponding file will be continuously overwritten with new images. Default value is 0.

`strftime`

If set to 1, expand the filename with date and time information from `strftime()`. Default value is 0.

The image muxer supports the .Y.U.V image file format. This format is special in that each image frame consists of three files, for each of the YUV420P components. To read or write this image file format, specify the name of the '.Y' file. The muxer will automatically open the '.U' and '.V' files as required.

## 22.9 matroska# TOC

Matroska container muxer.

This muxer implements the matroska and webm container specs.

### 22.9.1 Metadata# TOC

The recognized metadata settings in this muxer are:

`title`

Set title name provided to a single track.

`language`

Specify the language of the track in the Matroska languages form.

The language can be either the 3 letters bibliographic ISO-639-2 (ISO 639-2/B) form (like "fre" for French), or a language code mixed with a country code for specialities in languages (like "fre-ca" for Canadian French).

`stereo_mode`

Set stereo 3D video layout of two views in a single video track.

The following values are recognized:

'mono'

video is not stereo

`'left_right'`

Both views are arranged side by side, Left-eye view is on the left

`'bottom_top'`

Both views are arranged in top-bottom orientation, Left-eye view is at bottom

`'top_bottom'`

Both views are arranged in top-bottom orientation, Left-eye view is on top

`'checkerboard_rl'`

Each view is arranged in a checkerboard interleaved pattern, Left-eye view being first

`'checkerboard_lr'`

Each view is arranged in a checkerboard interleaved pattern, Right-eye view being first

`'row_interleaved_rl'`

Each view is constituted by a row based interleaving, Right-eye view is first row

`'row_interleaved_lr'`

Each view is constituted by a row based interleaving, Left-eye view is first row

`'col_interleaved_rl'`

Both views are arranged in a column based interleaving manner, Right-eye view is first column

`'col_interleaved_lr'`

Both views are arranged in a column based interleaving manner, Left-eye view is first column

`'anaglyph_cyan_red'`

All frames are in anaglyph format viewable through red-cyan filters

`'right_left'`

Both views are arranged side by side, Right-eye view is on the left

`'anaglyph_green_magenta'`

All frames are in anaglyph format viewable through green-magenta filters

`'block_lr'`

Both eyes laced in one Block, Left-eye view is first

`'block_rl'`

Both eyes laced in one Block, Right-eye view is first

For example a 3D WebM clip can be created using the following command line:

```
ffmpeg -i sample_left_right_clip.mpg -an -c:v libvpx -metadata stereo_mode=left_right -y stereo_clip.webm
```

## 22.9.2 Options# TOC

This muxer supports the following options:

`reserve_index_space`

By default, this muxer writes the index for seeking (called cues in Matroska terms) at the end of the file, because it cannot know in advance how much space to leave for the index at the beginning of the file. However for some use cases – e.g. streaming where seeking is possible but slow – it is useful to put the index at the beginning of the file.

If this option is set to a non-zero value, the muxer will reserve a given amount of space in the file header and then try to write the cues there when the muxing finishes. If the available space does not suffice, muxing will fail. A safe size for most use cases should be about 50kB per hour of video.

Note that cues are only written if the output is seekable and this option will have no effect if it is not.

## 22.10 md5# TOC

MD5 testing format.

This muxer computes and prints the MD5 hash of all the input audio and video frames. By default audio frames are converted to signed 16-bit raw audio and video frames to raw video before computing the hash.

The output of the muxer consists of a single line of the form: `MD5=MD5`, where *MD5* is a hexadecimal number representing the computed MD5 hash.

For example to compute the MD5 hash of the input converted to raw audio and video, and store it in the file `out.md5`:

```
ffmpeg -i INPUT -f md5 out.md5
```

You can print the MD5 to stdout with the command:

```
ffmpeg -i INPUT -f md5 -
```

See also the framemd5 muxer.

## 22.11 mov, mp4, ismv# TOC

MOV/MP4/ISMV (Smooth Streaming) muxer.

The mov/mp4/ismv muxer supports fragmentation. Normally, a MOV/MP4 file has all the metadata about all packets stored in one location (written at the end of the file, it can be moved to the start for better playback by adding *faststart* to the *movflags*, or using the *qt-faststart* tool). A fragmented file consists of a number of fragments, where packets and metadata about these packets are stored together. Writing a fragmented file has the advantage that the file is decodable even if the writing is interrupted (while a normal MOV/MP4 is undecodable if it is not properly finished), and it requires less memory when writing very long files (since writing normal MOV/MP4 files stores info about every single packet in memory until the file is closed). The downside is that it is less compatible with other applications.

### 22.11.1 Options# TOC

Fragmentation is enabled by setting one of the AVOptions that define how to cut the file into fragments:

`-moov_size bytes`

Reserves space for the moov atom at the beginning of the file instead of placing the moov atom at the end. If the space reserved is insufficient, muxing will fail.

`-movflags frag_keyframe`

Start a new fragment at each video keyframe.

`-frag_duration duration`

Create fragments that are *duration* microseconds long.

`-frag_size size`

Create fragments that contain up to *size* bytes of payload data.

`-movflags frag_custom`

Allow the caller to manually choose when to cut fragments, by calling `av_write_frame(ctx, NULL)` to write a fragment with the packets written so far. (This is only useful with other applications integrating libavformat, not from `ffmpeg`.)

`-min_frag_duration duration`

Don't create fragments that are shorter than *duration* microseconds long.



If more than one condition is specified, fragments are cut when one of the specified conditions is fulfilled. The exception to this is `-min_frag_duration`, which has to be fulfilled for any of the other conditions to apply.

Additionally, the way the output file is written can be adjusted through a few other options:

`-movflags empty_moov`

Write an initial moov atom directly at the start of the file, without describing any samples in it. Generally, an mdat/moov pair is written at the start of the file, as a normal MOV/MP4 file, containing only a short portion of the file. With this option set, there is no initial mdat atom, and the moov atom only describes the tracks but has a zero duration.

This option is implicitly set when writing ismv (Smooth Streaming) files.

`-movflags separate_moof`

Write a separate moof (movie fragment) atom for each track. Normally, packets for all tracks are written in a moof atom (which is slightly more efficient), but with this option set, the muxer writes one moof/mdat pair for each track, making it easier to separate tracks.

This option is implicitly set when writing ismv (Smooth Streaming) files.

`-movflags faststart`

Run a second pass moving the index (moov atom) to the beginning of the file. This operation can take a while, and will not work in various situations such as fragmented output, thus it is not enabled by default.

`-movflags rtphint`

Add RTP hinting tracks to the output file.

`-movflags disable_chpl`

Disable Nero chapter markers (chpl atom). Normally, both Nero chapters and a QuickTime chapter track are written to the file. With this option set, only the QuickTime chapter track will be written. Nero chapters can cause failures when the file is reprocessed with certain tagging programs, like mp3Tag 2.61a and iTunes 11.3, most likely other versions are affected as well.

`-movflags omit_tfhd_offset`

Do not write any absolute base\_data\_offset in tfhd atoms. This avoids tying fragments to absolute byte positions in the file/streams.

`-movflags default_base_moof`

Similarly to the `omit_tfhd_offset`, this flag avoids writing the absolute `base_data_offset` field in `tfhd` atoms, but does so by using the new `default-base-is-moof` flag instead. This flag is new from 14496-12:2012. This may make the fragments easier to parse in certain circumstances (avoiding basing track fragment location calculations on the implicit end of the previous track fragment).

### 22.11.2 Example# TOC

Smooth Streaming content can be pushed in real time to a publishing point on IIS with this muxer.

Example:

```
ffmpeg -re <normal input/transcoding options> -movflags isml+frag_keyframe -f ismv http://server/publishingpoint.isml/Streams(Encoder1)
```

### 22.11.3 Audible AAX# TOC

Audible AAX files are encrypted M4B files, and they can be decrypted by specifying a 4 byte activation secret.

```
ffmpeg -activation_bytes 1CEB00DA -i test.aax -vn -c:a copy output.mp4
```

## 22.12 mp3# TOC

The MP3 muxer writes a raw MP3 stream with the following optional features:

- An ID3v2 metadata header at the beginning (enabled by default). Versions 2.3 and 2.4 are supported, the `id3v2_version` private option controls which one is used (3 or 4). Setting `id3v2_version` to 0 disables the ID3v2 header completely.

The muxer supports writing attached pictures (APIC frames) to the ID3v2 header. The pictures are supplied to the muxer in form of a video stream with a single packet. There can be any number of those streams, each will correspond to a single APIC frame. The stream metadata tags *title* and *comment* map to APIC *description* and *picture type* respectively. See <http://id3.org/id3v2.4.0-frames> for allowed picture types.

Note that the APIC frames must be written at the beginning, so the muxer will buffer the audio frames until it gets all the pictures. It is therefore advised to provide the pictures as soon as possible to avoid excessive buffering.

- A Xing/LAME frame right after the ID3v2 header (if present). It is enabled by default, but will be written only if the output is seekable. The `write_xing` private option can be used to disable it. The frame contains various information that may be useful to the decoder, like the audio duration or encoder delay.
- A legacy ID3v1 tag at the end of the file (disabled by default). It may be enabled with the `write_id3v1` private option, but as its capabilities are very limited, its usage is not recommended.

Examples:

Write an mp3 with an ID3v2.3 header and an ID3v1 footer:

```
ffmpeg -i INPUT -id3v2_version 3 -write_id3v1 1 out.mp3
```

To attach a picture to an mp3 file select both the audio and the picture stream with map:

```
ffmpeg -i input.mp3 -i cover.png -c copy -map 0 -map 1  
-metadata:s:v title="Album cover" -metadata:s:v comment="Cover (Front)" out.mp3
```

Write a "clean" MP3 without any extra features:

```
ffmpeg -i input.wav -write_xing 0 -id3v2_version 0 out.mp3
```

## 22.13 mpegts# TOC

MPEG transport stream muxer.

This muxer implements ISO 13818-1 and part of ETSI EN 300 468.

The recognized metadata settings in mpegts muxer are `service_provider` and `service_name`. If they are not set the default for `service_provider` is "FFmpeg" and the default for `service_name` is "Service01".

### 22.13.1 Options# TOC

The muxer options are:

`-mpegts_original_network_id number`

Set the `original_network_id` (default 0x0001). This is unique identifier of a network in DVB. Its main use is in the unique identification of a service through the path `Original_Network_ID`, `Transport_Stream_ID`.

`-mpegts_transport_stream_id number`

Set the `transport_stream_id` (default 0x0001). This identifies a transponder in DVB.

`-mpegts_service_id number`

Set the `service_id` (default 0x0001) also known as program in DVB.

`-mpegts_service_type number`

Set the program `service_type` (default *digital\_tv*), see below a list of pre defined values.

`-mpegts_pmt_start_pid number`

Set the first PID for PMT (default 0x1000, max 0x1f00).

`-mpegts_start_pid number`

Set the first PID for data packets (default 0x0100, max 0x0f00).

`-mpegts_m2ts_mode number`

Enable m2ts mode if set to 1. Default value is -1 which disables m2ts mode.

`-muxrate number`

Set a constant muxrate (default VBR).

`-pcr_period number`

Override the default PCR retransmission time (default 20ms), ignored if variable muxrate is selected.

`pat_period number`

Maximal time in seconds between PAT/PMT tables.

`sdt_period number`

Maximal time in seconds between SDT tables.

`-pes_payload_size number`

Set minimum PES packet payload in bytes.

`-mpegts_flags flags`

Set flags (see below).

`-mpegts_copyts number`

Preserve original timestamps, if value is set to 1. Default value is -1, which results in shifting timestamps so that they start from 0.

`-tables_version number`

Set PAT, PMT and SDT version (default 0, valid values are from 0 to 31, inclusively). This option allows updating stream structure so that standard consumer may detect the change. To do so, reopen output AVFormatContext (in case of API usage) or restart ffmpeg instance, cyclically changing `tables_version` value:

```
ffmpeg -i source1.ts -codec copy -f mpegts -tables_version 0 udp://1.1.1.1:1111
ffmpeg -i source2.ts -codec copy -f mpegts -tables_version 1 udp://1.1.1.1:1111
...
ffmpeg -i source3.ts -codec copy -f mpegts -tables_version 31 udp://1.1.1.1:1111
ffmpeg -i source1.ts -codec copy -f mpegts -tables_version 0 udp://1.1.1.1:1111
ffmpeg -i source2.ts -codec copy -f mpegts -tables_version 1 udp://1.1.1.1:1111
...
```

Option `mpegts_service_type` accepts the following values:

`hex_value`

Any hexadecimal value between 0x01 to 0xff as defined in ETSI 300 468.

`digital_tv`

Digital TV service.

`digital_radio`

Digital Radio service.

`teletext`

Teletext service.

`advanced_codec_digital_radio`

Advanced Codec Digital Radio service.

`mpeg2_digital_hdtv`

MPEG2 Digital HDTV service.

`advanced_codec_digital_sdtv`

Advanced Codec Digital SDTV service.

`advanced_codec_digital_hdtv`

Advanced Codec Digital HDTV service.

Option `mpegts_flags` may take a set of such flags:

`resend_headers`

Reemit PAT/PMT before writing the next packet.

`latm`

Use LATM packetization for AAC.

`pat_pmt_at_frames`

Reemit PAT and PMT at each video frame.

### 22.13.2 Example# TOC

```
ffmpeg -i file.mpg -c copy \  
-mpegts_original_network_id 0x1122 \  
-mpegts_transport_stream_id 0x3344 \  
-mpegts_service_id 0x5566 \  
-mpegts_pmt_start_pid 0x1500 \  
-mpegts_start_pid 0x150 \  
-metadata service_provider="Some provider" \  
-metadata service_name="Some Channel" \  
-y out.ts
```

### 22.14 null# TOC

Null muxer.

This muxer does not generate any output file, it is mainly useful for testing or benchmarking purposes.

For example to benchmark decoding with `ffmpeg` you can use the command:

```
ffmpeg -benchmark -i INPUT -f null out.null
```

Note that the above command does not read or write the `out.null` file, but specifying the output file is required by the `ffmpeg` syntax.

Alternatively you can write the command as:

```
ffmpeg -benchmark -i INPUT -f null -
```

### 22.15 nut# TOC

`-syncpoints flags`

Change the syncpoint usage in nut:

*default* use the normal low-overhead seeking aids.

*none* do not use the syncpoints at all, reducing the overhead but making the stream non-seekable;

Use of this option is not recommended, as the resulting files are very damage sensitive and seeking is not possible. Also in general the overhead from syncpoints is negligible. Note, `-write_index 0` can be used to disable all growing data tables, allowing to mux endless streams with limited memory and without these disadvantages.

*timestamped* extend the syncpoint with a wallclock field.

The *none* and *timestamped* flags are experimental.

`-write_index bool`

Write index at the end, the default is to write an index.

```
ffmpeg -i INPUT -f_strict experimental -syncpoints none - | processor
```

## 22.16 ogg# TOC

Ogg container muxer.

`-page_duration duration`

Preferred page duration, in microseconds. The muxer will attempt to create pages that are approximately *duration* microseconds long. This allows the user to compromise between seek granularity and container overhead. The default is 1 second. A value of 0 will fill all segments, making pages as large as possible. A value of 1 will effectively use 1 packet-per-page in most situations, giving a small seek granularity at the cost of additional container overhead.

`-serial_offset value`

Serial value from which to set the streams serial number. Setting it to different and sufficiently large values ensures that the produced ogg files can be safely chained.

## 22.17 segment, stream\_segment, ssegment# TOC

Basic stream segmenter.

This muxer outputs streams to a number of separate files of nearly fixed duration. Output filename pattern can be set in a fashion similar to image2, or by using a `strftime` template if the `strftime` option is enabled.

`stream_segment` is a variant of the muxer used to write to streaming output formats, i.e. which do not require global headers, and is recommended for outputting e.g. to MPEG transport stream segments. `ssegment` is a shorter alias for `stream_segment`.

Every segment starts with a keyframe of the selected reference stream, which is set through the `reference_stream` option.

Note that if you want accurate splitting for a video file, you need to make the input key frames correspond to the exact splitting times expected by the segmenter, or the segment muxer will start the new segment with the key frame found next after the specified start time.

The segment muxer works best with a single constant frame rate video.

Optionally it can generate a list of the created segments, by setting the option `segment_list`. The list type is specified by the `segment_list_type` option. The entry filenames in the segment list are set by default to the basename of the corresponding segment files.

See also the hls muxer, which provides a more specific implementation for HLS segmentation.

### 22.17.1 Options# TOC

The segment muxer supports the following options:

`reference_stream specifier`

Set the reference stream, as specified by the string *specifier*. If *specifier* is set to `auto`, the reference is chosen automatically. Otherwise it must be a stream specifier (see the “Stream specifiers” chapter in the ffmpeg manual) which specifies the reference stream. The default value is `auto`.

`segment_format format`

Override the inner container format, by default it is guessed by the filename extension.

`segment_format_options options_list`

Set output format options using a `:`-separated list of `key=value` parameters. Values containing the `:` special character must be escaped.

`segment_list name`

Generate also a listfile named *name*. If not specified no listfile is generated.

`segment_list_flags flags`

Set flags affecting the segment list generation.

It currently supports the following flags:

‘cache’

Allow caching (only affects M3U8 list files).

‘live’

Allow live-friendly file generation.

`segment_list_size size`

Update the list file so that it contains at most *size* segments. If 0 the list file will contain all the segments. Default value is 0.

`segment_list_entry_prefix prefix`

Prepend *prefix* to each entry. Useful to generate absolute paths. By default no prefix is applied.



`segment_list_type type`

Select the listing format.

The following values are recognized:

‘flat’

Generate a flat list for the created segments, one segment per line.

‘csv, ext’

Generate a list for the created segments, one segment per line, each line matching the format (comma-separated values):

*segment\_filename, segment\_start\_time, segment\_end\_time*

*segment\_filename* is the name of the output file generated by the muxer according to the provided pattern. CSV escaping (according to RFC4180) is applied if required.

*segment\_start\_time* and *segment\_end\_time* specify the segment start and end time expressed in seconds.

A list file with the suffix ".csv" or ".ext" will auto-select this format.

‘ext’ is deprecated in favor of ‘csv’.

‘ffconcat’

Generate an ffconcat file for the created segments. The resulting file can be read using the FFmpeg concat demuxer.

A list file with the suffix ".ffcat" or ".ffconcat" will auto-select this format.

‘m3u8’

Generate an extended M3U8 file, version 3, compliant with <http://tools.ietf.org/id/draft-pantos-http-live-streaming>.

A list file with the suffix ".m3u8" will auto-select this format.

If not specified the type is guessed from the list file name suffix.

`segment_time time`

Set segment duration to *time*, the value must be a duration specification. Default value is "2". See also the `segment_times` option.

Note that splitting may not be accurate, unless you force the reference stream key-frames at the given time. See the introductory notice and the examples below.

`segment_atclocktime 1/0`

If set to "1" split at regular clock time intervals starting from 00:00 o'clock. The *time* value specified in `segment_time` is used for setting the length of the splitting interval.

For example with `segment_time` set to "900" this makes it possible to create files at 12:00 o'clock, 12:15, 12:30, etc.

Default value is "0".

`segment_time_delta delta`

Specify the accuracy time when selecting the start time for a segment, expressed as a duration specification. Default value is "0".

When delta is specified a key-frame will start a new segment if its PTS satisfies the relation:

$$\text{PTS} \geq \text{start\_time} - \text{time\_delta}$$

This option is useful when splitting video content, which is always split at GOP boundaries, in case a key frame is found just before the specified split time.

In particular may be used in combination with the `ffmpeg` option *force\_key\_frames*. The key frame times specified by *force\_key\_frames* may not be set accurately because of rounding issues, with the consequence that a key frame time may result set just before the specified time. For constant frame rate videos a value of  $1/(2*\text{frame\_rate})$  should address the worst case mismatch between the specified time and the time set by *force\_key\_frames*.

`segment_times times`

Specify a list of split points. *times* contains a list of comma separated duration specifications, in increasing order. See also the `segment_time` option.

`segment_frames frames`

Specify a list of split video frame numbers. *frames* contains a list of comma separated integer numbers, in increasing order.

This option specifies to start a new segment whenever a reference stream key frame is found and the sequential number (starting from 0) of the frame is greater or equal to the next value in the list.

`segment_wrap limit`

Wrap around segment index once it reaches *limit*.

`segment_start_number` *number*

Set the sequence number of the first segment. Defaults to 0.

`strftime` *1/0*

Use the `strftime` function to define the name of the new segments to write. If this is selected, the output segment name must contain a `strftime` function template. Default value is 0.

`break_non_keyframes` *1/0*

If enabled, allow segments to start on frames other than keyframes. This improves behavior on some players when the time between keyframes is inconsistent, but may make things worse on others, and can cause some oddities during seeking. Defaults to 0.

`reset_timestamps` *1/0*

Reset timestamps at the begin of each segment, so that each segment will start with near-zero timestamps. It is meant to ease the playback of the generated segments. May not work with some combinations of muxers/codecs. It is set to 0 by default.

`initial_offset` *offset*

Specify timestamp offset to apply to the output packet timestamps. The argument must be a time duration specification, and defaults to 0.

## 22.17.2 Examples# TOC

- Remux the content of file `in.mkv` to a list of segments `out-000.nut`, `out-001.nut`, etc., and write the list of generated segments to `out.list`:

```
ffmpeg -i in.mkv -codec copy -map 0 -f segment -segment_list out.list out%03d.nut
```

- Segment input and set output format options for the output segments:

```
ffmpeg -i in.mkv -f segment -segment_time 10 -segment_format_options movflags=+faststart out%03d.mp4
```

- Segment the input file according to the split points specified by the `segment_times` option:

```
ffmpeg -i in.mkv -codec copy -map 0 -f segment -segment_list out.csv -segment_times 1,2,3,5,8,13,21 out%03d.nut
```

- Use the `ffmpeg force_key_frames` option to force key frames in the input at the specified location, together with the segment option `segment_time_delta` to account for possible roundings operated when setting key frame times.

```
ffmpeg -i in.mkv -force_key_frames 1,2,3,5,8,13,21 -codec:v mpeg4 -codec:a pcm_s16le -map 0 \
-f segment -segment_list out.csv -segment_times 1,2,3,5,8,13,21 -segment_time_delta 0.05 out%03d.nut
```

In order to force key frames on the input file, transcoding is required.

- Segment the input file by splitting the input file according to the frame numbers sequence specified with the `segment_frames` option:

```
ffmpeg -i in.mkv -codec copy -map 0 -f segment -segment_list out.csv -segment_frames 100,200,300,500,800 out%03d.nut
```

- Convert the `in.mkv` to TS segments using the `libx264` and `libfaac` encoders:

```
ffmpeg -i in.mkv -map 0 -codec:v libx264 -codec:a libfaac -f ssegment -segment_list out.list out%03d.ts
```

- Segment the input file, and create an M3U8 live playlist (can be used as live HLS source):

```
ffmpeg -re -i in.mkv -codec copy -map 0 -f segment -segment_list playlist.m3u8 \
-segment_list_flags +live -segment_time 10 out%03d.mkv
```

## 22.18 smoothstreaming# TOC

Smooth Streaming muxer generates a set of files (Manifest, chunks) suitable for serving with conventional web server.

`window_size`

Specify the number of fragments kept in the manifest. Default 0 (keep all).

`extra_window_size`

Specify the number of fragments kept outside of the manifest before removing from disk. Default 5.

`lookahead_count`

Specify the number of lookahead fragments. Default 2.

`min_frag_duration`

Specify the minimum fragment duration (in microseconds). Default 5000000.

`remove_at_exit`

Specify whether to remove all fragments when finished. Default 0 (do not remove).

## 22.19 tee# TOC

The tee muxer can be used to write the same data to several files or any other kind of muxer. It can be used, for example, to both stream a video to the network and save it to disk at the same time.

It is different from specifying several outputs to the `ffmpeg` command-line tool because the audio and video data will be encoded only once with the tee muxer; encoding can be a very expensive process. It is not useful when using the `libavformat` API directly because it is then possible to feed the same packets to several muxers directly.

The slave outputs are specified in the file name given to the muxer, separated by '|'. If any of the slave name contains the '|' separator, leading or trailing spaces or any special character, it must be escaped (see (ffmpeg-utils)the "Quoting and escaping" section in the ffmpeg-utils(1) manual).

Muxer options can be specified for each slave by prepending them as a list of *key=value* pairs separated by ':', between square brackets. If the options values contain a special character or the ':' separator, they must be escaped; note that this is a second level escaping.

The following special options are also recognized:

`f`

Specify the format name. Useful if it cannot be guessed from the output name suffix.

`bsfs[ /spec ]`

Specify a list of bitstream filters to apply to the specified output.

It is possible to specify to which streams a given bitstream filter applies, by appending a stream specifier to the option separated by /. *spec* must be a stream specifier (see Format stream specifiers). If the stream specifier is not specified, the bitstream filters will be applied to all streams in the output.

Several bitstream filters can be specified, separated by ",".

`select`

Select the streams that should be mapped to the slave output, specified by a stream specifier. If not specified, this defaults to all the input streams.

## 22.19.1 Examples# TOC

- Encode something and both archive it in a WebM file and stream it as MPEG-TS over UDP (the streams need to be explicitly mapped):

```
ffmpeg -i ... -c:v libx264 -c:a mp2 -f tee -map 0:v -map 0:a  
"archive-20121107.mkv|[f=mpegts]udp://10.0.1.255:1234/"
```

- Use ffmpeg to encode the input, and send the output to three different destinations. The `dump_extra` bitstream filter is used to add extradata information to all the output video keyframes packets, as requested by the MPEG-TS format. The `select` option is applied to `out.aac` in order to make it contain only audio packets.

```
ffmpeg -i ... -map 0 -flags +global_header -c:v libx264 -c:a aac -strict experimental  
-f tee "[bsfs/v=dump_extra]out.ts|[movflags=+faststart]out.mp4|[select=a]out.aac"
```

- As below, but select only stream `a:1` for the audio output. Note that a second level escaping must be performed, as `":"` is a special character used to separate options.

```
ffmpeg -i ... -map 0 -flags +global_header -c:v libx264 -c:a aac -strict experimental  
-f tee "[bsfs/v=dump_extra]out.ts|[movflags=+faststart]out.mp4|[select='a:1\']out.aac"
```

Note: some codecs may need different options depending on the output format; the auto-detection of this can not work with the tee muxer. The main example is the `global_header` flag.

## 22.20 webm\_dash\_manifest# TOC

WebM DASH Manifest muxer.

This muxer implements the WebM DASH Manifest specification to generate the DASH manifest XML. It also supports manifest generation for DASH live streams.

For more information see:

- WebM DASH Specification:  
<https://sites.google.com/a/webmproject.org/wiki/adaptive-streaming/webm-dash-specification>
- ISO DASH Specification:  
[http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274\\_ISO\\_IEC\\_23009-1\\_2014.zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274_ISO_IEC_23009-1_2014.zip)

### 22.20.1 Options# TOC

This muxer supports the following options:

`adaptation_sets`

This option has the following syntax: "id=x,streams=a,b,c id=y,streams=d,e" where x and y are the unique identifiers of the adaptation sets and a,b,c,d and e are the indices of the corresponding audio and video streams. Any number of adaptation sets can be added using this option.

`live`

Set this to 1 to create a live stream DASH Manifest. Default: 0.

`chunk_start_index`

Start index of the first chunk. This will go in the 'startNumber' attribute of the 'SegmentTemplate' element in the manifest. Default: 0.

`chunk_duration_ms`

Duration of each chunk in milliseconds. This will go in the 'duration' attribute of the 'SegmentTemplate' element in the manifest. Default: 1000.

`utc_timing_url`

URL of the page that will return the UTC timestamp in ISO format. This will go in the 'value' attribute of the 'UTCTiming' element in the manifest. Default: None.

time\_shift\_buffer\_depth

Smallest time (in seconds) shifting buffer for which any Representation is guaranteed to be available. This will go in the 'timeShiftBufferDepth' attribute of the 'MPD' element. Default: 60.

minimum\_update\_period

Minimum update period (in seconds) of the manifest. This will go in the 'minimumUpdatePeriod' attribute of the 'MPD' element. Default: 0.

## 22.20.2 Example# TOC

```
ffmpeg -f webm_dash_manifest -i video1.webm \  
-f webm_dash_manifest -i video2.webm \  
-f webm_dash_manifest -i audio1.webm \  
-f webm_dash_manifest -i audio2.webm \  
-map 0 -map 1 -map 2 -map 3 \  
-c copy \  
-f webm_dash_manifest \  
-adaptation_sets "id=0,streams=0,1 id=1,streams=2,3" \  
manifest.xml
```

## 22.21 webm\_chunk# TOC

WebM Live Chunk Muxer.

This muxer writes out WebM headers and chunks as separate files which can be consumed by clients that support WebM Live streams via DASH.

### 22.21.1 Options# TOC

This muxer supports the following options:

chunk\_start\_index

Index of the first chunk (defaults to 0).

header

Filename of the header where the initialization data will be written.

audio\_chunk\_duration

Duration of each audio chunk in milliseconds (defaults to 5000).

### 22.21.2 Example# TOC

```
ffmpeg -f v4l2 -i /dev/video0 \  
-f alsa -i hw:0 \  
-map 0:0 \  
-c:v libvpx-vp9 \  
-c:a libopus -b:a 128k -ac 2 -ar 48000 -f webm_dash_manifest -i /dev/null -o manifest.xml
```

```

-s 640x360 -keyint_min 30 -g 30 \
-f webm_chunk \
-header webm_live_video_360.hdr \
-chunk_start_index 1 \
webm_live_video_360_%d.chk \
-map 1:0 \
-c:a libvorbis \
-b:a 128k \
-f webm_chunk \
-header webm_live_audio_128.hdr \
-chunk_start_index 1 \
-audio_chunk_duration 1000 \
webm_live_audio_128_%d.chk

```

## 23 Metadata# TOC

FFmpeg is able to dump metadata from media files into a simple UTF-8-encoded INI-like text file and then load it back using the metadata muxer/demuxer.

The file format is as follows:

1. A file consists of a header and a number of metadata tags divided into sections, each on its own line.
2. The header is a ‘;FFMETADATA’ string, followed by a version number (now 1).
3. Metadata tags are of the form ‘key=value’
4. Immediately after header follows global metadata
5. After global metadata there may be sections with per-stream/per-chapter metadata.
6. A section starts with the section name in uppercase (i.e. STREAM or CHAPTER) in brackets (‘[’, ‘]’) and ends with next section or end of file.
7. At the beginning of a chapter section there may be an optional timebase to be used for start/end values. It must be in form ‘TIMEBASE=num/den’, where *num* and *den* are integers. If the timebase is missing then start/end times are assumed to be in milliseconds.

Next a chapter section must contain chapter start and end times in form ‘START=num’, ‘END=num’, where *num* is a positive integer.

8. Empty lines and lines starting with ‘;’ or ‘#’ are ignored.
9. Metadata keys or values containing special characters (‘=’, ‘;’, ‘#’, ‘\’ and a newline) must be escaped with a backslash ‘\’.
10. Note that whitespace in metadata (e.g. ‘foo = bar’) is considered to be a part of the tag (in the example above key is ‘foo ’, value is ‘ bar’).

A ffmadata file might look like this:

```

;FFMETADATA1
title=bike\shed
;this is a comment
artist=FFmpeg troll team

```

```

[CHAPTER]
TIMEBASE=1/1000

```



```
START=0
#chapter ends at 0:01:00
END=60000
title=chapter \#1
[STREAM]
title=multi\
line
```

By using the `ffmetadata` muxer and demuxer it is possible to extract metadata from an input file to an `ffmetadata` file, and then transcode the file into an output file with the edited `ffmetadata` file.

Extracting an `ffmetadata` file with `ffmpeg` goes as follows:

```
ffmpeg -i INPUT -f ffmetadata FFMETADATAFILE
```

Reinserting edited metadata information from the `FFMETADATAFILE` file can be done as:

```
ffmpeg -i INPUT -i FFMETADATAFILE -map_metadata 1 -codec copy OUTPUT
```

## 24 Protocols# TOC

Protocols are configured elements in `FFmpeg` that enable access to resources that require specific protocols.

When you configure your `FFmpeg` build, all the supported protocols are enabled by default. You can list all available ones using the configure option "`--list-protocols`".

You can disable all the protocols using the configure option "`--disable-protocols`", and selectively enable a protocol using the option "`--enable-protocol=PROTOCOL`", or you can disable a particular protocol using the option "`--disable-protocol=PROTOCOL`".

The option "`-protocols`" of the `ff*` tools will display the list of supported protocols.

A description of the currently available protocols follows.

### 24.1 async# TOC

Asynchronous data filling wrapper for input stream.

Fill data in a background thread, to decouple I/O operation from demux thread.

```
async:URL
async:http://host/resource
async:cache:http://host/resource
```

## 24.2 bluray# TOC

Read BluRay playlist.

The accepted options are:

angle

BluRay angle

chapter

Start chapter (1...N)

playlist

Playlist to read (BDMV/PLAYLIST/?????.mpls)

Examples:

Read longest playlist from BluRay mounted to /mnt/bluray:

```
bluray:/mnt/bluray
```

Read angle 2 of playlist 4 from BluRay mounted to /mnt/bluray, start from chapter 2:

```
-playlist 4 -angle 2 -chapter 2 bluray:/mnt/bluray
```

## 24.3 cache# TOC

Caching wrapper for input stream.

Cache the input stream to temporary file. It brings seeking capability to live streams.

```
cache:URL
```

## 24.4 concat# TOC

Physical concatenation protocol.

Read and seek from many resources in sequence as if they were a unique resource.

A URL accepted by this protocol has the syntax:

```
concat:URL1|URL2|...|URLN
```

where *URL1*, *URL2*, ..., *URLN* are the urls of the resource to be concatenated, each one possibly specifying a distinct protocol.

For example to read a sequence of files `split1.mpeg`, `split2.mpeg`, `split3.mpeg` with `ffplay` use the command:

```
ffplay concat:split1.mpeg\|split2.mpeg\|split3.mpeg
```

Note that you may need to escape the character "|" which is special for many shells.

## 24.5 crypto# TOC

AES-encrypted stream reading protocol.

The accepted options are:

`key`

Set the AES decryption key binary block from given hexadecimal representation.

`iv`

Set the AES decryption initialization vector binary block from given hexadecimal representation.

Accepted URL formats:

```
crypto:URL  
crypto+URL
```

## 24.6 data# TOC

Data in-line in the URI. See [http://en.wikipedia.org/wiki/Data\\_URI\\_scheme](http://en.wikipedia.org/wiki/Data_URI_scheme).

For example, to convert a GIF file given inline with `ffmpeg`:

```
ffmpeg -i "data:image/gif;base64,R0lGODdhCAAIAMIEAAAAAAAAA//8AAP//AP//////////////////ywAAAAACAAIAAADF0gEDLojDgdGiJdJqUX02iB4E8Q9jUMkADs=" smiley.png
```

## 24.7 file# TOC

File access protocol.

Read from or write to a file.

A file URL can have the form:

```
file:filename
```

where *filename* is the path of the file to read.

An URL that does not have a protocol prefix will be assumed to be a file URL. Depending on the build, an URL that looks like a Windows path with the drive letter at the beginning will also be assumed to be a file URL (usually not the case in builds for unix-like systems).

For example to read from a file `input.mpeg` with `ffmpeg` use the command:

```
ffmpeg -i file:input.mpeg output.mpeg
```

This protocol accepts the following options:

`truncate`

Truncate existing files on write, if set to 1. A value of 0 prevents truncating. Default value is 1.

`blocksize`

Set I/O operation maximum block size, in bytes. Default value is `INT_MAX`, which results in not limiting the requested block size. Setting this value reasonably low improves user termination request reaction time, which is valuable for files on slow medium.

## 24.8 ftp# TOC

FTP (File Transfer Protocol).

Read from or write to remote resources using FTP protocol.

Following syntax is required.

```
ftp://[user[:password]@]server[:port]/path/to/remote/resource.mpeg
```

This protocol accepts the following options.

`timeout`

Set timeout in microseconds of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

`ftp-anonymous-password`

Password used when login as anonymous user. Typically an e-mail address should be used.

`ftp-write-seekable`

Control seekability of connection during encoding. If set to 1 the resource is supposed to be seekable, if set to 0 it is assumed not to be seekable. Default value is 0.

NOTE: Protocol can be used as output, but it is recommended to not do it, unless special care is taken (tests, customized server configuration etc.). Different FTP servers behave in different way during seek operation. `ff*` tools may produce incomplete content due to server limitations.

## 24.9 gopher# TOC

Gopher protocol.

## 24.10 hls# TOC

Read Apple HTTP Live Streaming compliant segmented stream as a uniform one. The M3U8 playlists describing the segments can be remote HTTP resources or local files, accessed using the standard file protocol. The nested protocol is declared by specifying "+*proto*" after the hls URI scheme name, where *proto* is either "file" or "http".

```
hls+http://host/path/to/remote/resource.m3u8
hls+file://path/to/local/resource.m3u8
```

Using this protocol is discouraged - the hls demuxer should work just as well (if not, please report the issues) and is more complete. To use the hls demuxer instead, simply use the direct URLs to the m3u8 files.

## 24.11 http# TOC

HTTP (Hyper Text Transfer Protocol).

This protocol accepts the following options:

`seekable`

Control seekability of connection. If set to 1 the resource is supposed to be seekable, if set to 0 it is assumed not to be seekable, if set to -1 it will try to autodetect if it is seekable. Default value is -1.

`chunked_post`

If set to 1 use chunked Transfer-Encoding for posts, default is 1.

`content_type`

Set a specific content type for the POST messages.

`headers`

Set custom HTTP headers, can override built in default headers. The value must be a string encoding the headers.

`multiple_requests`

Use persistent connections if set to 1, default is 0.

`post_data`

Set custom HTTP post data.

`user-agent`

`user_agent`

Override the User-Agent header. If not specified the protocol will use a string describing the libavformat build. ("Lavf/<version>")

`timeout`

Set timeout in microseconds of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

`mime_type`

Export the MIME type.

`icy`

If set to 1 request ICY (SHOUTcast) metadata from the server. If the server supports this, the metadata has to be retrieved by the application by reading the `icy_metadata_headers` and `icy_metadata_packet` options. The default is 1.

`icy_metadata_headers`

If the server supports ICY metadata, this contains the ICY-specific HTTP reply headers, separated by newline characters.

`icy_metadata_packet`

If the server supports ICY metadata, and `icy` was set to 1, this contains the last non-empty metadata packet sent by the server. It should be polled in regular intervals by applications interested in mid-stream metadata updates.

`cookies`

Set the cookies to be sent in future requests. The format of each cookie is the same as the value of a Set-Cookie HTTP response field. Multiple cookies can be delimited by a newline character.

`offset`

Set initial byte offset.

`end_offset`

Try to limit the request to bytes preceding this offset.

`method`

When used as a client option it sets the HTTP method for the request.

When used as a server option it sets the HTTP method that is going to be expected from the client(s). If the expected and the received HTTP method do not match the client will be given a Bad Request response. When unset the HTTP method is not checked for now. This will be replaced by autodetection in the future.

## listen

If set to 1 enables experimental HTTP server. This can be used to send data when used as an output option, or read data from a client with HTTP POST when used as an input option. If set to 2 enables experimental mutli-client HTTP server. This is not yet implemented in ffmpeg.c or ffserver.c and thus must not be used as a command line option.

```
# Server side (sending):
ffmpeg -i somefile.ogg -c copy -listen 1 -f ogg http://server:port

# Client side (receiving):
ffmpeg -i http://server:port -c copy somefile.ogg

# Client can also be done with wget:
wget http://server:port -O somefile.ogg

# Server side (receiving):
ffmpeg -listen 1 -i http://server:port -c copy somefile.ogg

# Client side (sending):
ffmpeg -i somefile.ogg -chunked_post 0 -c copy -f ogg http://server:port

# Client can also be done with wget:
wget --post-file=somefile.ogg http://server:port
```

### 24.11.1 HTTP Cookies# TOC

Some HTTP requests will be denied unless cookie values are passed in with the request. The `cookies` option allows these cookies to be specified. At the very least, each cookie must specify a value along with a path and domain. HTTP requests that match both the domain and path will automatically include the cookie value in the HTTP Cookie header field. Multiple cookies can be delimited by a newline.

The required syntax to play a stream specifying a cookie is:

```
ffplay -cookies "nlqtid=nltd=tsn; path=/; domain=somedomain.com;" http://somedomain.com/somestream.m3u8
```

### 24.12 Icecast# TOC

Icecast protocol (stream to Icecast servers)

This protocol accepts the following options:

`ice_genre`

Set the stream genre.

`ice_name`

Set the stream name.

`ice_description`

Set the stream description.

`ice_url`

Set the stream website URL.

`ice_public`

Set if the stream should be public. The default is 0 (not public).

`user_agent`

Override the User-Agent header. If not specified a string of the form "Lavf/<version>" will be used.

`password`

Set the Icecast mountpoint password.

`content_type`

Set the stream content type. This must be set if it is different from audio/mpeg.

`legacy_icecast`

This enables support for Icecast versions < 2.4.0, that do not support the HTTP PUT method but the SOURCE method.

`icecast://[username[:password]@]server:port/mountpoint`

## **24.13 mmst# TOC**

MMS (Microsoft Media Server) protocol over TCP.

## **24.14 mmsh# TOC**

MMS (Microsoft Media Server) protocol over HTTP.



The required syntax is:

```
mmsch://server[:port][[/app][/playpath]]
```

## 24.15 md5# TOC

MD5 output protocol.

Computes the MD5 hash of the data to be written, and on close writes this to the designated output or stdout if none is specified. It can be used to test muxers without writing an actual file.

Some examples follow.

```
# Write the MD5 hash of the encoded AVI file to the file output.avi.md5.
ffmpeg -i input.flv -f avi -y md5:output.avi.md5
```

```
# Write the MD5 hash of the encoded AVI file to stdout.
ffmpeg -i input.flv -f avi -y md5:
```

Note that some formats (typically MOV) require the output protocol to be seekable, so they will fail with the MD5 output protocol.

## 24.16 pipe# TOC

UNIX pipe access protocol.

Read and write from UNIX pipes.

The accepted syntax is:

```
pipe:[number]
```

*number* is the number corresponding to the file descriptor of the pipe (e.g. 0 for stdin, 1 for stdout, 2 for stderr). If *number* is not specified, by default the stdout file descriptor will be used for writing, stdin for reading.

For example to read from stdin with `ffmpeg`:

```
cat test.wav | ffmpeg -i pipe:0
# ...this is the same as...
cat test.wav | ffmpeg -i pipe:
```

For writing to stdout with `ffmpeg`:

```
ffmpeg -i test.wav -f avi pipe:1 | cat > test.avi
# ...this is the same as...
ffmpeg -i test.wav -f avi pipe: | cat > test.avi
```

This protocol accepts the following options:

blocksize

Set I/O operation maximum block size, in bytes. Default value is `INT_MAX`, which results in not limiting the requested block size. Setting this value reasonably low improves user termination request reaction time, which is valuable if data transmission is slow.

Note that some formats (typically MOV), require the output protocol to be seekable, so they will fail with the pipe output protocol.

## 24.17 rtmp# TOC

Real-Time Messaging Protocol.

The Real-Time Messaging Protocol (RTMP) is used for streaming multimedia content across a TCP/IP network.

The required syntax is:

```
rtmp://[username:password@]server[:port][/app][/instance][/playpath]
```

The accepted parameters are:

username

An optional username (mostly for publishing).

password

An optional password (mostly for publishing).

server

The address of the RTMP server.

port

The number of the TCP port to use (by default is 1935).

app

It is the name of the application to access. It usually corresponds to the path where the application is installed on the RTMP server (e.g. `/ondemand/`, `/flash/live/`, etc.). You can override the value parsed from the URI through the `rtmp_app` option, too.

playpath

It is the path or name of the resource to play with reference to the application specified in *app*, may be prefixed by "mp4:". You can override the value parsed from the URI through the `rtmp_playpath` option, too.

`listen`

Act as a server, listening for an incoming connection.

`timeout`

Maximum time to wait for the incoming connection. Implies `listen`.

Additionally, the following parameters can be set via command line options (or in code via `AVOptions`):

`rtmp_app`

Name of application to connect on the RTMP server. This option overrides the parameter specified in the URI.

`rtmp_buffer`

Set the client buffer time in milliseconds. The default is 3000.

`rtmp_conn`

Extra arbitrary AMF connection parameters, parsed from a string, e.g. like `B:1 S:authMe O:1 NN:code:1.23 NS:flag:ok O:0`. Each value is prefixed by a single character denoting the type, B for Boolean, N for number, S for string, O for object, or Z for null, followed by a colon. For Booleans the data must be either 0 or 1 for FALSE or TRUE, respectively. Likewise for Objects the data must be 0 or 1 to end or begin an object, respectively. Data items in subobjects may be named, by prefixing the type with 'N' and specifying the name before the value (i.e. `NB:myFlag:1`). This option may be used multiple times to construct arbitrary AMF sequences.

`rtmp_flashver`

Version of the Flash plugin used to run the SWF player. The default is LNX 9,0,124,2. (When publishing, the default is FMLE/3.0 (compatible; <libavformat version>).)

`rtmp_flush_interval`

Number of packets flushed in the same request (RTMPT only). The default is 10.

`rtmp_live`

Specify that the media is a live stream. No resuming or seeking in live streams is possible. The default value is `any`, which means the subscriber first tries to play the live stream specified in the playpath. If a live stream of that name is not found, it plays the recorded stream. The other possible values are `live` and `recorded`.

`rtmp_pageurl`

URL of the web page in which the media was embedded. By default no value will be sent.

`rtmp_playpath`

Stream identifier to play or to publish. This option overrides the parameter specified in the URL.

`rtmp_subscribe`

Name of live stream to subscribe to. By default no value will be sent. It is only sent if the option is specified or if `rtmp_live` is set to live.

`rtmp_swfhash`

SHA256 hash of the decompressed SWF file (32 bytes).

`rtmp_swfsize`

Size of the decompressed SWF file, required for SWFVerification.

`rtmp_swfurl`

URL of the SWF player for the media. By default no value will be sent.

`rtmp_swfverify`

URL to player swf file, compute hash/size automatically.

`rtmp_tcurl`

URL of the target stream. Defaults to `proto://host[:port]/app`.

For example to read with `ffplay` a multimedia resource named "sample" from the application "vod" from an RTMP server "myserver":

```
ffplay rtmp://myserver/vod/sample
```

To publish to a password protected server, passing the playpath and app names separately:

```
ffmpeg -re -i <input> -f flv -rtmp_playpath some/long/path -rtmp_app long/app/name rtmp://username:password@myserver/
```

## 24.18 rtmpe# TOC

Encrypted Real-Time Messaging Protocol.

The Encrypted Real-Time Messaging Protocol (RTMPE) is used for streaming multimedia content within standard cryptographic primitives, consisting of Diffie-Hellman key exchange and HMACSHA256, generating a pair of RC4 keys.

## 24.19 rtmps# TOC

Real-Time Messaging Protocol over a secure SSL connection.

The Real-Time Messaging Protocol (RTMPS) is used for streaming multimedia content across an encrypted connection.

## 24.20 rtmpt# TOC

Real-Time Messaging Protocol tunneled through HTTP.

The Real-Time Messaging Protocol tunneled through HTTP (RTMPT) is used for streaming multimedia content within HTTP requests to traverse firewalls.

## 24.21 rtmpte# TOC

Encrypted Real-Time Messaging Protocol tunneled through HTTP.

The Encrypted Real-Time Messaging Protocol tunneled through HTTP (RTMPTE) is used for streaming multimedia content within HTTP requests to traverse firewalls.

## 24.22 rtmpts# TOC

Real-Time Messaging Protocol tunneled through HTTPS.

The Real-Time Messaging Protocol tunneled through HTTPS (RTMPTS) is used for streaming multimedia content within HTTPS requests to traverse firewalls.

## 24.23 libsmbclient# TOC

libsmbclient permits one to manipulate CIFS/SMB network resources.

Following syntax is required.

```
smb://[[domain:]user[:password@]]server[/share[/path[/file]]]
```

This protocol accepts the following options.

`timeout`

Set timeout in milliseconds of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

`truncate`

Truncate existing files on write, if set to 1. A value of 0 prevents truncating. Default value is 1.

workgroup

Set the workgroup used for making connections. By default workgroup is not specified.

For more information see: <http://www.samba.org/>.

## 24.24 libssh# TOC

Secure File Transfer Protocol via libssh

Read from or write to remote resources using SFTP protocol.

Following syntax is required.

```
sftp://[user[:password]@]server[:port]/path/to/remote/resource.mpeg
```

This protocol accepts the following options.

timeout

Set timeout of socket I/O operations used by the underlying low level operation. By default it is set to -1, which means that the timeout is not specified.

truncate

Truncate existing files on write, if set to 1. A value of 0 prevents truncating. Default value is 1.

private\_key

Specify the path of the file containing private key to use during authorization. By default libssh searches for keys in the `~/ .ssh/` directory.

Example: Play a file stored on remote server.

```
ffplay sftp://user:password@server_address:22/home/user/resource.mpeg
```

## 24.25 librtmp rtmp, rtmpe, rtmps, rtmpt, rtmppte# TOC

Real-Time Messaging Protocol and its variants supported through librtmp.

Requires the presence of the librtmp headers and library during configuration. You need to explicitly configure the build with "`--enable-librtmp`". If enabled this will replace the native RTMP protocol.

This protocol provides most client functions and a few server functions needed to support RTMP, RTMP tunneled in HTTP (RTMPT), encrypted RTMP (RTMPE), RTMP over SSL/TLS (RTMPS) and tunneled variants of these encrypted types (RTMPTE, RTMPTS).

The required syntax is:

```
rtmp_proto://server[:port][/app][/playpath] options
```

where *rtmp\_proto* is one of the strings "rtmp", "rtmpt", "rtmpe", "rtmps", "rtmpte", "rtmpts" corresponding to each RTMP variant, and *server*, *port*, *app* and *playpath* have the same meaning as specified for the RTMP native protocol. *options* contains a list of space-separated options of the form *key=val*.

See the librtmp manual page (man 3 librtmp) for more information.

For example, to stream a file in real-time to an RTMP server using `ffmpeg`:

```
ffmpeg -re -i myfile -f flv rtmp://myserver/live/mystream
```

To play the same stream using `ffplay`:

```
ffplay "rtmp://myserver/live/mystream live=1"
```

## 24.26 rtp# TOC

Real-time Transport Protocol.

The required syntax for an RTP URL is: `rtp://hostname[:port][?option=val...]`

*port* specifies the RTP port to use.

The following URL options are supported:

`ttl=n`

Set the TTL (Time-To-Live) value (for multicast only).

`rtcpport=n`

Set the remote RTCP port to *n*.

`localrtpport=n`

Set the local RTP port to *n*.

`localrtcpport=n'`

Set the local RTCP port to *n*.

`pkt_size=n`

Set max packet size (in bytes) to *n*.

`connect=0 | 1`

Do a `connect ( )` on the UDP socket (if set to 1) or not (if set to 0).

`sources=ip[ , ip]`

List allowed source IP addresses.

`block=ip[ , ip]`

List disallowed (blocked) source IP addresses.

`write_to_source=0 | 1`

Send packets to the source address of the latest received packet (if set to 1) or to a default remote address (if set to 0).

`localport=n`

Set the local RTP port to *n*.

This is a deprecated option. Instead, `localrtpport` should be used.

Important notes:

1. If `rtcpport` is not set the RTCP port will be set to the RTP port value plus 1.
2. If `localrtpport` (the local RTP port) is not set any available port will be used for the local RTP and RTCP ports.
3. If `localrtcpport` (the local RTCP port) is not set it will be set to the local RTP port value plus 1.

## 24.27 rtsp# TOC

Real-Time Streaming Protocol.

RTSP is not technically a protocol handler in libavformat, it is a demuxer and muxer. The demuxer supports both normal RTSP (with data transferred over RTP; this is used by e.g. Apple and Microsoft) and Real-RTSP (with data transferred over RDT).

The muxer can be used to send a stream using RTSP ANNOUNCE to a server supporting it (currently Darwin Streaming Server and Mischa Spiegelmock's RTSP server).

The required syntax for a RTSP url is:

`rtsp://hostname[:port]/path`

Options can be set on the `ffmpeg/ffplay` command line, or set in code via `AVOptions` or in `avformat_open_input`.

The following options are supported.



`initial_pause`

Do not start playing the stream immediately if set to 1. Default value is 0.

`rtsp_transport`

Set RTSP transport protocols.

It accepts the following values:

`'udp'`

Use UDP as lower transport protocol.

`'tcp'`

Use TCP (interleaving within the RTSP control channel) as lower transport protocol.

`'udp_multicast'`

Use UDP multicast as lower transport protocol.

`'http'`

Use HTTP tunneling as lower transport protocol, which is useful for passing proxies.

Multiple lower transport protocols may be specified, in that case they are tried one at a time (if the setup of one fails, the next one is tried). For the muxer, only the `'tcp'` and `'udp'` options are supported.

`rtsp_flags`

Set RTSP flags.

The following values are accepted:

`'filter_src'`

Accept packets only from negotiated peer address and port.

`'listen'`

Act as a server, listening for an incoming connection.

`'prefer_tcp'`

Try TCP for RTP transport first, if TCP is available as RTSP RTP transport.

Default value is 'none'.

`allowed_media_types`

Set media types to accept from the server.

The following flags are accepted:

'video'  
'audio'  
'data'

By default it accepts all media types.

`min_port`

Set minimum local UDP port. Default value is 5000.

`max_port`

Set maximum local UDP port. Default value is 65000.

`timeout`

Set maximum timeout (in seconds) to wait for incoming connections.

A value of -1 means infinite (default). This option implies the `rtsp_flags` set to 'listen'.

`reorder_queue_size`

Set number of packets to buffer for handling of reordered packets.

`sttimeout`

Set socket TCP I/O timeout in microseconds.

`user-agent`

Override User-Agent header. If not specified, it defaults to the libavformat identifier string.

When receiving data over UDP, the demuxer tries to reorder received packets (since they may arrive out of order, or packets may get lost totally). This can be disabled by setting the maximum demuxing delay to zero (via the `max_delay` field of `AVFormatContext`).

When watching multi-bitrate Real-RTSP streams with `ffplay`, the streams to display can be chosen with `-vst n` and `-ast n` for video and audio respectively, and can be switched on the fly by pressing `v` and `a`.

## 24.27.1 Examples# TOC

The following examples all make use of the `ffplay` and `ffmpeg` tools.

- Watch a stream over UDP, with a max reordering delay of 0.5 seconds:

```
ffplay -max_delay 500000 -rtsp_transport udp rtsp://server/video.mp4
```

- Watch a stream tunneled over HTTP:

```
ffplay -rtsp_transport http rtsp://server/video.mp4
```

- Send a stream in realtime to a RTSP server, for others to watch:

```
ffmpeg -re -i input -f rtsp -muxdelay 0.1 rtsp://server/live.sdp
```

- Receive a stream in realtime:

```
ffmpeg -rtsp_flags listen -i rtsp://ownaddress/live.sdp output
```

## 24.28 sap# TOC

Session Announcement Protocol (RFC 2974). This is not technically a protocol handler in libavformat, it is a muxer and demuxer. It is used for signalling of RTP streams, by announcing the SDP for the streams regularly on a separate port.

### 24.28.1 Muxer# TOC

The syntax for a SAP url given to the muxer is:

```
sap://destination[:port][?options]
```

The RTP packets are sent to *destination* on port *port*, or to port 5004 if no port is specified. *options* is a &-separated list. The following options are supported:

`announce_addr=address`

Specify the destination IP address for sending the announcements to. If omitted, the announcements are sent to the commonly used SAP announcement multicast address 224.2.127.254 (sap.mcast.net), or ff0e::2:7ffe if *destination* is an IPv6 address.

`announce_port=port`

Specify the port to send the announcements on, defaults to 9875 if not specified.

`ttl=t1`

Specify the time to live value for the announcements and RTP packets, defaults to 255.

`same_port=0/1`

If set to 1, send all RTP streams on the same port pair. If zero (the default), all streams are sent on unique ports, with each stream on a port 2 numbers higher than the previous. VLC/Live555 requires this to be set to 1, to be able to receive the stream. The RTP stack in libavformat for receiving requires all streams to be sent on unique ports.

Example command lines follow.

To broadcast a stream on the local subnet, for watching in VLC:

```
ffmpeg -re -i input -f sap sap://224.0.0.255?same_port=1
```

Similarly, for watching in `ffplay`:

```
ffmpeg -re -i input -f sap sap://224.0.0.255
```

And for watching in `ffplay`, over IPv6:

```
ffmpeg -re -i input -f sap sap://[ff0e::1:2:3:4]
```

## 24.28.2 Demuxer# TOC

The syntax for a SAP url given to the demuxer is:

```
sap://[address][:port]
```

*address* is the multicast address to listen for announcements on, if omitted, the default 224.2.127.254 (sap.mcast.net) is used. *port* is the port that is listened on, 9875 if omitted.

The demuxers listens for announcements on the given address and port. Once an announcement is received, it tries to receive that particular stream.

Example command lines follow.

To play back the first stream announced on the normal SAP multicast address:

```
ffplay sap://
```

To play back the first stream announced on one the default IPv6 SAP multicast address:

```
ffplay sap://[ff0e::2:7ffe]
```

## 24.29 sctp# TOC

Stream Control Transmission Protocol.

The accepted URL syntax is:

`sctp://host:port[?options]`

The protocol accepts the following options:

`listen`

If set to any value, listen for an incoming connection. Outgoing connection is done by default.

`max_streams`

Set the maximum number of streams. By default no limit is set.

## 24.30 srtp# TOC

Secure Real-time Transport Protocol.

The accepted options are:

`srtp_in_suite`

`srtp_out_suite`

Select input and output encoding suites.

Supported values:

`'AES_CM_128_HMAC_SHA1_80'`

`'SRTP_AES128_CM_HMAC_SHA1_80'`

`'AES_CM_128_HMAC_SHA1_32'`

`'SRTP_AES128_CM_HMAC_SHA1_32'`

`srtp_in_params`

`srtp_out_params`

Set input and output encoding parameters, which are expressed by a base64-encoded representation of a binary block. The first 16 bytes of this binary block are used as master key, the following 14 bytes are used as master salt.

## 24.31 subfile# TOC

Virtually extract a segment of a file or another stream. The underlying stream must be seekable.

Accepted options:

`start`

Start offset of the extracted segment, in bytes.

`end`

End offset of the extracted segment, in bytes.

Examples:

Extract a chapter from a DVD VOB file (start and end sectors obtained externally and multiplied by 2048):

```
subfile,,start,153391104,end,268142592,,:/media/dvd/VIDEO_TS/VTS_08_1.VOB
```

Play an AVI file directly from a TAR archive:

```
subfile,,start,183241728,end,366490624,, :archive.tar
```

## 24.32 tcp# TOC

Transmission Control Protocol.

The required syntax for a TCP url is:

```
tcp://hostname:port[?options]
```

*options* contains a list of &-separated options of the form *key=val*.

The list of supported options follows.

```
listen=1/0
```

Listen for an incoming connection. Default value is 0.

```
timeout=microseconds
```

Set raise error timeout, expressed in microseconds.

This option is only relevant in read mode: if no data arrived in more than this time interval, raise error.

```
listen_timeout=milliseconds
```

Set listen timeout, expressed in milliseconds.

The following example shows how to setup a listening TCP connection with `ffmpeg`, which is then accessed with `ffplay`:

```
ffmpeg -i input -f format tcp://hostname:port?listen  
ffplay tcp://hostname:port
```

## 24.33 `tls`# TOC

Transport Layer Security (TLS) / Secure Sockets Layer (SSL)

The required syntax for a TLS/SSL url is:

```
tls://hostname:port[?options]
```

The following parameters can be set via command line options (or in code via `AVOptions`):

```
ca_file, cafile=filename
```

A file containing certificate authority (CA) root certificates to treat as trusted. If the linked TLS library contains a default this might not need to be specified for verification to work, but not all libraries and setups have defaults built in. The file must be in OpenSSL PEM format.

```
tls_verify=1/0
```

If enabled, try to verify the peer that we are communicating with. Note, if using OpenSSL, this currently only makes sure that the peer certificate is signed by one of the root certificates in the CA database, but it does not validate that the certificate actually matches the host name we are trying to connect to. (With GnuTLS, the host name is validated as well.)

This is disabled by default since it requires a CA database to be provided by the caller in many cases.

```
cert_file, cert=filename
```

A file containing a certificate to use in the handshake with the peer. (When operating as server, in listen mode, this is more often required by the peer, while client certificates only are mandated in certain setups.)

```
key_file, key=filename
```

A file containing the private key for the certificate.

```
listen=1/0
```

If enabled, listen for connections on the provided port, and assume the server role in the handshake instead of the client role.

Example command lines:

To create a TLS/SSL server that serves an input stream.

```
ffmpeg -i input -f format tls://hostname:port?listen&cert=server.crt&key=server.key
```

To play back a stream from the TLS/SSL server using `ffplay`:

```
ffplay tls://hostname:port
```

## 24.34 udp# TOC

User Datagram Protocol.

The required syntax for an UDP URL is:

```
udp://hostname:port[?options]
```

*options* contains a list of &-separated options of the form *key=val*.

In case threading is enabled on the system, a circular buffer is used to store the incoming data, which allows one to reduce loss of data due to UDP socket buffer overruns. The *fifo\_size* and *overrun\_nonfatal* options are related to this buffer.

The list of supported options follows.

*buffer\_size=size*

Set the UDP maximum socket buffer size in bytes. This is used to set either the receive or send buffer size, depending on what the socket is used for. Default is 64KB. See also *fifo\_size*.

*localport=port*

Override the local UDP port to bind with.

*localaddr=addr*

Choose the local IP address. This is useful e.g. if sending multicast and the host has multiple interfaces, where the user can choose which interface to send on by specifying the IP address of that interface.

*pkt\_size=size*

Set the size in bytes of UDP packets.

*reuse=1/0*

Explicitly allow or disallow reusing UDP sockets.

*ttl=t1*

Set the time to live value (for multicast only).

*connect=1/0*

Initialize the UDP socket with `connect ( )`. In this case, the destination address can't be changed with `ff_udp_set_remote_url` later. If the destination address isn't known at the start, this option can be specified in `ff_udp_set_remote_url`, too. This allows finding out the source address for the packets



with `getsockname`, and makes writes return with `AVERROR(ECONNREFUSED)` if "destination unreachable" is received. For receiving, this gives the benefit of only receiving packets from the specified peer address/port.

```
sources=address[,address]
```

Only receive packets sent to the multicast group from one of the specified sender IP addresses.

```
block=address[,address]
```

Ignore packets sent to the multicast group from the specified sender IP addresses.

```
fifo_size=units
```

Set the UDP receiving circular buffer size, expressed as a number of packets with size of 188 bytes. If not specified defaults to  $7 \times 4096$ .

```
overrun_nonfatal=1/0
```

Survive in case of UDP receiving circular buffer overrun. Default value is 0.

```
timeout=microseconds
```

Set raise error timeout, expressed in microseconds.

This option is only relevant in read mode: if no data arrived in more than this time interval, raise error.

```
broadcast=1/0
```

Explicitly allow or disallow UDP broadcasting.

Note that broadcasting may not work properly on networks having a broadcast storm protection.

## 24.34.1 Examples# TOC

- Use `ffmpeg` to stream over UDP to a remote endpoint:

```
ffmpeg -i input -f format udp://hostname:port
```

- Use `ffmpeg` to stream in mpegts format over UDP using 188 sized UDP packets, using a large input buffer:

```
ffmpeg -i input -f mpegts udp://hostname:port?pkt_size=188&buffer_size=65535
```

- Use `ffmpeg` to receive over UDP from a remote endpoint:

```
ffmpeg -i udp://[multicast-address]:port ...
```

## 24.35 unix# TOC

Unix local socket

The required syntax for a Unix socket URL is:

```
unix://filepath
```

The following parameters can be set via command line options (or in code via `AVOptions`):

`timeout`

Timeout in ms.

`listen`

Create the Unix socket in listening mode.

## 25 Device Options# TOC

The `libavdevice` library provides the same interface as `libavformat`. Namely, an input device is considered like a demuxer, and an output device like a muxer, and the interface and generic device options are the same provided by `libavformat` (see the `ffmpeg-formats` manual).

In addition each input or output device may support so-called private options, which are specific for that component.

Options may be set by specifying *-option value* in the FFmpeg tools, or by setting the value explicitly in the device `AVFormatContext` options or using the `libavutil/opt.h` API for programmatic use.

## 26 Input Devices# TOC

Input devices are configured elements in FFmpeg which enable accessing the data coming from a multimedia device attached to your system.

When you configure your FFmpeg build, all the supported input devices are enabled by default. You can list all available ones using the configure option `"-list-indevs"`.

You can disable all the input devices using the configure option `"-disable-indevs"`, and selectively enable an input device using the option `"-enable-indev=INDEV"`, or you can disable a particular input device using the option `"-disable-indev=INDEV"`.

The option `"-devices"` of the `ff*` tools will display the list of supported input devices.

A description of the currently available input devices follows.

## 26.1 alsa# TOC

ALSA (Advanced Linux Sound Architecture) input device.

To enable this input device during configuration you need libasound installed on your system.

This device allows capturing from an ALSA device. The name of the device to capture has to be an ALSA card identifier.

An ALSA identifier has the syntax:

```
hw: CARD[ , DEV[ , SUBDEV ] ]
```

where the *DEV* and *SUBDEV* components are optional.

The three arguments (in order: *CARD*, *DEV*, *SUBDEV*) specify card number or identifier, device number and subdevice number (-1 means any).

To see the list of cards currently recognized by your system check the files `/proc/asound/cards` and `/proc/asound/devices`.

For example to capture with `ffmpeg` from an ALSA device with card id 0, you may run the command:

```
ffmpeg -f alsa -i hw:0 alsaout.wav
```

For more information see: <http://www.alsa-project.org/alsa-doc/alsa-lib/pcm.html>

### 26.1.1 Options# TOC

`sample_rate`

Set the sample rate in Hz. Default is 48000.

`channels`

Set the number of channels. Default is 2.

## 26.2 avfoundation# TOC

AVFoundation input device.

AVFoundation is the currently recommended framework by Apple for streamgrabbing on OSX >= 10.7 as well as on iOS. The older QTKit framework has been marked deprecated since OSX version 10.7.

The input filename has to be given in the following syntax:

```
-i "[[VIDEO]:[AUDIO]]"
```

The first entry selects the video input while the latter selects the audio input. The stream has to be specified by the device name or the device index as shown by the device list. Alternatively, the video and/or audio input device can be chosen by index using the `-video_device_index <INDEX>` and/or `-audio_device_index <INDEX>`, overriding any device name or index given in the input filename.

All available devices can be enumerated by using `-list_devices true`, listing all device names and corresponding indices.

There are two device name aliases:

`default`

Select the AVFoundation default device of the corresponding type.

`none`

Do not record the corresponding media type. This is equivalent to specifying an empty device name or index.

### 26.2.1 Options# TOC

AVFoundation supports the following options:

`-list_devices <TRUE|FALSE>`

If set to true, a list of all available input devices is given showing all device names and indices.

`-video_device_index <INDEX>`

Specify the video device by its index. Overrides anything given in the input filename.

`-audio_device_index <INDEX>`

Specify the audio device by its index. Overrides anything given in the input filename.

`-pixel_format <FORMAT>`

Request the video device to use a specific pixel format. If the specified format is not supported, a list of available formats is given and the first one in this list is used instead. Available pixel formats are: `monob`, `rgb555be`, `rgb555le`, `rgb565be`, `rgb565le`, `rgb24`, `bgr24`, `0rgb`, `bgr0`, `0bgr`, `rgb0`, `bgr48be`, `uyvy422`, `yuva444p`, `yuva444p16le`, `yuv444p`, `yuv422p16`, `yuv422p10`, `yuv444p10`, `yuv420p`, `nv12`, `yuyv422`, `gray`

`-framerate`

Set the grabbing frame rate. Default is `ntsc`, corresponding to a frame rate of `30000/1001`.

`-video_size`

Set the video frame size.

`-capture_cursor`

Capture the mouse pointer. Default is 0.

`-capture_mouse_clicks`

Capture the screen mouse clicks. Default is 0.

## 26.2.2 Examples# TOC

- Print the list of AVFoundation supported devices and exit:

```
$ ffmpeg -f avfoundation -list_devices true -i ""
```

- Record video from video device 0 and audio from audio device 0 into out.avi:

```
$ ffmpeg -f avfoundation -i "0:0" out.avi
```

- Record video from video device 2 and audio from audio device 1 into out.avi:

```
$ ffmpeg -f avfoundation -video_device_index 2 -i ":1" out.avi
```

- Record video from the system default video device using the pixel format bgr0 and do not record any audio into out.avi:

```
$ ffmpeg -f avfoundation -pixel_format bgr0 -i "default:none" out.avi
```

## 26.3 bktr# TOC

BSD video input device.

### 26.3.1 Options# TOC

`framerate`

Set the frame rate.

`video_size`

Set the video frame size. Default is vga.

`standard`

Available values are:

```
'pal'
'ntsc'
'secam'
'paln'
'palm'
'ntscj'
```

## 26.4 decklink# TOC

The decklink input device provides capture capabilities for Blackmagic DeckLink devices.

To enable this input device, you need the Blackmagic DeckLink SDK and you need to configure with the appropriate `--extra-cflags` and `--extra-ldflags`. On Windows, you need to run the IDL files through `widl`.

DeckLink is very picky about the formats it supports. Pixel format is `uyvy422` or `v210`, framerate and video size must be determined for your device with `-list_formats 1`. Audio sample rate is always 48 kHz and the number of channels can be 2, 8 or 16.

### 26.4.1 Options# TOC

`list_devices`

If set to `true`, print a list of devices and exit. Defaults to `false`.

`list_formats`

If set to `true`, print a list of supported formats and exit. Defaults to `false`.

`bm_v210`

If set to `'1'`, video is captured in 10 bit `v210` instead of `uyvy422`. Not all Blackmagic devices support this option.

### 26.4.2 Examples# TOC

- List input devices:

```
ffmpeg -f decklink -list_devices 1 -i dummy
```

- List supported formats:

```
ffmpeg -f decklink -list_formats 1 -i 'Intensity Pro'
```

- Capture video clip at 1080i50 (format 11):

```
ffmpeg -f decklink -i 'Intensity Pro@11' -acodec copy -vcodec copy output.avi
```

- Capture video clip at 1080i50 10 bit:

```
ffmpeg -bm_v210 1 -f decklink -i 'UltraStudio Mini Recorder@11' -acodec copy -vcodec copy output.avi
```

## 26.5 dshow# TOC

Windows DirectShow input device.

DirectShow support is enabled when FFmpeg is built with the mingw-w64 project. Currently only audio and video devices are supported.

Multiple devices may be opened as separate inputs, but they may also be opened on the same input, which should improve synchronism between them.

The input name should be in the format:

```
TYPE=NAME[ :TYPE=NAME]
```

where *TYPE* can be either *audio* or *video*, and *NAME* is the device's name or alternative name..

### 26.5.1 Options# TOC

If no options are specified, the device's defaults are used. If the device does not support the requested options, it will fail to open.

`video_size`

Set the video size in the captured video.

`framerate`

Set the frame rate in the captured video.

`sample_rate`

Set the sample rate (in Hz) of the captured audio.

`sample_size`

Set the sample size (in bits) of the captured audio.

`channels`

Set the number of channels in the captured audio.

`list_devices`

If set to true, print a list of devices and exit.

`list_options`

If set to true, print a list of selected device's options and exit.

video\_device\_number

Set video device number for devices with the same name (starts at 0, defaults to 0).

audio\_device\_number

Set audio device number for devices with the same name (starts at 0, defaults to 0).

pixel\_format

Select pixel format to be used by DirectShow. This may only be set when the video codec is not set or set to rawvideo.

audio\_buffer\_size

Set audio device buffer size in milliseconds (which can directly impact latency, depending on the device). Defaults to using the audio device's default buffer size (typically some multiple of 500ms).

Setting this value too low can degrade performance. See also

[http://msdn.microsoft.com/en-us/library/windows/desktop/dd377582\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/dd377582(v=vs.85).aspx)

video\_pin\_name

Select video capture pin to use by name or alternative name.

audio\_pin\_name

Select audio capture pin to use by name or alternative name.

crossbar\_video\_input\_pin\_number

Select video input pin number for crossbar device. This will be routed to the crossbar device's Video Decoder output pin. Note that changing this value can affect future invocations (sets a new default) until system reboot occurs.

crossbar\_audio\_input\_pin\_number

Select audio input pin number for crossbar device. This will be routed to the crossbar device's Audio Decoder output pin. Note that changing this value can affect future invocations (sets a new default) until system reboot occurs.

show\_video\_device\_dialog

If set to true, before capture starts, popup a display dialog to the end user, allowing them to change video filter properties and configurations manually. Note that for crossbar devices, adjusting values in this dialog may be needed at times to toggle between PAL (25 fps) and NTSC (29.97) input frame rates, sizes, interlacing, etc. Changing these values can enable different scan rates/frame rates and avoiding green bars at the bottom, flickering scan lines, etc. Note that with some devices, changing these properties can also affect future invocations (sets new defaults) until system reboot occurs.



`show_audio_device_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to change audio filter properties and configurations manually.

`show_video_crossbar_connection_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to manually modify crossbar pin routings, when it opens a video device.

`show_audio_crossbar_connection_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to manually modify crossbar pin routings, when it opens an audio device.

`show_analog_tv_tuner_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to manually modify TV channels and frequencies.

`show_analog_tv_tuner_audio_dialog`

If set to `true`, before capture starts, popup a display dialog to the end user, allowing them to manually modify TV audio (like mono vs. stereo, Language A,B or C).

`audio_device_load`

Load an audio capture filter device from file instead of searching it by name. It may load additional parameters too, if the filter supports the serialization of its properties to. To use this an audio capture source has to be specified, but it can be anything even fake one.

`audio_device_save`

Save the currently used audio capture filter device and its parameters (if the filter supports it) to a file. If a file with the same name exists it will be overwritten.

`video_device_load`

Load a video capture filter device from file instead of searching it by name. It may load additional parameters too, if the filter supports the serialization of its properties to. To use this a video capture source has to be specified, but it can be anything even fake one.

`video_device_save`

Save the currently used video capture filter device and its parameters (if the filter supports it) to a file. If a file with the same name exists it will be overwritten.

## 26.5.2 Examples# TOC

- Print the list of DirectShow supported devices and exit:

```
$ ffmpeg -list_devices true -f dshow -i dummy
```

- Open video device *Camera*:

```
$ ffmpeg -f dshow -i video="Camera"
```

- Open second video device with name *Camera*:

```
$ ffmpeg -f dshow -video_device_number 1 -i video="Camera"
```

- Open video device *Camera* and audio device *Microphone*:

```
$ ffmpeg -f dshow -i video="Camera":audio="Microphone"
```

- Print the list of supported options in selected device and exit:

```
$ ffmpeg -list_options true -f dshow -i video="Camera"
```

- Specify pin names to capture by name or alternative name, specify alternative device name:

```
$ ffmpeg -f dshow -audio_pin_name "Audio Out" -video_pin_name 2 -i video=video="device_gmp_\\?pci#ven_1a0a&dev_62021461&rev_0184&e2c7d856&a0a18[65e8773d-8f56-11d0-a3b9-00a0c9223196]\\{ca465100-dab0-4d59-818f-8c477284adf6}" :audio="Microphone"
```

- Configure a crossbar device, specifying crossbar pins, allow user to adjust video capture properties at startup:

```
$ ffmpeg -f dshow -show_video_device_dialog true -crossbar_video_input_pin_number 0  
-crossbar_audio_input_pin_number 3 -i video="AVerMedia BDA Analog Capture":audio="AVerMedia BDA Analog Capture"
```

## 26.6 dv1394# TOC

Linux DV 1394 input device.

### 26.6.1 Options# TOC

framerate

Set the frame rate. Default is 25.

standard

Available values are:

‘pal’  
‘ntsc’

Default value is ntsc.

## 26.7 fbdev# TOC

Linux framebuffer input device.

The Linux framebuffer is a graphic hardware-independent abstraction layer to show graphics on a computer monitor, typically on the console. It is accessed through a file device node, usually `/dev/fb0`.

For more detailed information read the file `Documentation/fb/framebuffer.txt` included in the Linux source tree.

See also <http://linux-fbdev.sourceforge.net/>, and `fbset(1)`.

To record from the framebuffer device `/dev/fb0` with `ffmpeg`:

```
ffmpeg -f fbdev -framerate 10 -i /dev/fb0 out.avi
```

You can take a single screenshot image with the command:

```
ffmpeg -f fbdev -framerate 1 -i /dev/fb0 -frames:v 1 screenshot.jpeg
```

### 26.7.1 Options# TOC

`framerate`

Set the frame rate. Default is 25.

## 26.8 gdigrab# TOC

Win32 GDI-based screen capture device.

This device allows you to capture a region of the display on Windows.

There are two options for the input filename:

`desktop`

or

`title=window_title`

The first option will capture the entire desktop, or a fixed region of the desktop. The second option will instead capture the contents of a single window, regardless of its position on the screen.

For example, to grab the entire desktop using `ffmpeg`:

```
ffmpeg -f gdigrab -framerate 6 -i desktop out.mpg
```

Grab a 640x480 region at position 10 , 20:

```
ffmpeg -f gdigrab -framerate 6 -offset_x 10 -offset_y 20 -video_size vga -i desktop out.mpg
```

Grab the contents of the window named "Calculator"

```
ffmpeg -f gdigrab -framerate 6 -i title=Calculator out.mpg
```

## 26.8.1 Options# TOC

`draw_mouse`

Specify whether to draw the mouse pointer. Use the value 0 to not draw the pointer. Default value is 1.

`framerate`

Set the grabbing frame rate. Default value is `ntsc`, corresponding to a frame rate of 30000/1001.

`show_region`

Show grabbed region on screen.

If `show_region` is specified with 1, then the grabbing region will be indicated on screen. With this option, it is easy to know what is being grabbed if only a portion of the screen is grabbed.

Note that `show_region` is incompatible with grabbing the contents of a single window.

For example:

```
ffmpeg -f gdigrab -show_region 1 -framerate 6 -video_size cif -offset_x 10 -offset_y 20 -i desktop out.mpg
```

`video_size`

Set the video frame size. The default is to capture the full screen if `desktop` is selected, or the full window size if `title=window_title` is selected.

`offset_x`

When capturing a region with `video_size`, set the distance from the left edge of the screen or desktop.

Note that the offset calculation is from the top left corner of the primary monitor on Windows. If you have a monitor positioned to the left of your primary monitor, you will need to use a negative `offset_x` value to move the region to that monitor.

`offset_y`

When capturing a region with `video_size`, set the distance from the top edge of the screen or desktop.

Note that the offset calculation is from the top left corner of the primary monitor on Windows. If you have a monitor positioned above your primary monitor, you will need to use a negative `offset_y` value to move the region to that monitor.

## 26.9 iec61883# TOC

FireWire DV/HDV input device using libiec61883.

To enable this input device, you need libiec61883, libraw1394 and libavc1394 installed on your system. Use the configure option `--enable-libiec61883` to compile with the device enabled.

The iec61883 capture device supports capturing from a video device connected via IEEE1394 (FireWire), using libiec61883 and the new Linux FireWire stack (juju). This is the default DV/HDV input method in Linux Kernel 2.6.37 and later, since the old FireWire stack was removed.

Specify the FireWire port to be used as input file, or "auto" to choose the first port connected.

### 26.9.1 Options# TOC

`dvtype`

Override autodetection of DV/HDV. This should only be used if auto detection does not work, or if usage of a different device type should be prohibited. Treating a DV device as HDV (or vice versa) will not work and result in undefined behavior. The values `auto`, `dv` and `hdv` are supported.

`dvbuffer`

Set maximum size of buffer for incoming data, in frames. For DV, this is an exact value. For HDV, it is not frame exact, since HDV does not have a fixed frame size.

`dvguid`

Select the capture device by specifying its GUID. Capturing will only be performed from the specified device and fails if no device with the given GUID is found. This is useful to select the input if multiple devices are connected at the same time. Look at `/sys/bus/firewire/devices` to find out the GUIDs.

### 26.9.2 Examples# TOC

- Grab and show the input of a FireWire DV/HDV device.

```
ffplay -f iec61883 -i auto
```

- Grab and record the input of a FireWire DV/HDV device, using a packet buffer of 100000 packets if the source is HDV.

```
ffmpeg -f iec61883 -i auto -hdvbuffer 100000 out.mpg
```

## 26.10 jack# TOC

JACK input device.

To enable this input device during configuration you need libjack installed on your system.

A JACK input device creates one or more JACK writable clients, one for each audio channel, with name *client\_name*:input\_*N*, where *client\_name* is the name provided by the application, and *N* is a number which identifies the channel. Each writable client will send the acquired data to the FFmpeg input device.

Once you have created one or more JACK readable clients, you need to connect them to one or more JACK writable clients.

To connect or disconnect JACK clients you can use the `jack_connect` and `jack_disconnect` programs, or do it through a graphical interface, for example with `qjackctl`.

To list the JACK clients and their properties you can invoke the command `jack_lsp`.

Follows an example which shows how to capture a JACK readable client with `ffmpeg`.

```
# Create a JACK writable client with name "ffmpeg".
$ ffmpeg -f jack -i ffmpeg -y out.wav

# Start the sample jack_metro readable client.
$ jack_metro -b 120 -d 0.2 -f 4000

# List the current JACK clients.
$ jack_lsp -c
system:capture_1
system:capture_2
system:playback_1
system:playback_2
ffmpeg:input_1
metro:120_bpm

# Connect metro to the ffmpeg writable client.
$ jack_connect metro:120_bpm ffmpeg:input_1
```

For more information read: <http://jackaudio.org/>

### 26.10.1 Options# TOC

channels

Set the number of channels. Default is 2.

## 26.11 lavfi# TOC

Libavfilter input virtual device.

This input device reads data from the open output pads of a libavfilter filtergraph.

For each filtergraph open output, the input device will create a corresponding stream which is mapped to the generated output. Currently only video data is supported. The filtergraph is specified through the option *graph*.

### 26.11.1 Options# TOC

*graph*

Specify the filtergraph to use as input. Each video open output must be labelled by a unique string of the form "outN", where *N* is a number starting from 0 corresponding to the mapped input stream generated by the device. The first unlabelled output is automatically assigned to the "out0" label, but all the others need to be specified explicitly.

The suffix "+subcc" can be appended to the output label to create an extra stream with the closed captions packets attached to that output (experimental; only for EIA-608 / CEA-708 for now). The subcc streams are created after all the normal streams, in the order of the corresponding stream. For example, if there is "out19+subcc", "out7+subcc" and up to "out42", the stream #43 is subcc for stream #7 and stream #44 is subcc for stream #19.

If not specified defaults to the filename specified for the input device.

*graph\_file*

Set the filename of the filtergraph to be read and sent to the other filters. Syntax of the filtergraph is the same as the one specified by the option *graph*.

*dumpgraph*

Dump graph to stderr.

### 26.11.2 Examples# TOC

- Create a color video stream and play it back with `ffplay`:

```
ffplay -f lavfi -graph "color=c=pink [out0]" dummy
```

- As the previous example, but use filename for specifying the graph description, and omit the "out0" label:

```
ffplay -f lavfi color=c=pink
```

- Create three different video test filtered sources and play them:

```
ffplay -f lavfi -graph "testsrc [out0]; testsrc,hflip [out1]; testsrc,negate [out2]" test3
```

- Read an audio stream from a file using the amovie source and play it back with ffplay:

```
ffplay -f lavfi "amovie=test.wav"
```

- Read an audio stream and a video stream and play it back with ffplay:

```
ffplay -f lavfi "movie=test.avi[out0];amovie=test.wav[out1]"
```

- Dump decoded frames to images and closed captions to a file (experimental):

```
ffmpeg -f lavfi -i "movie=test.ts[out0+subcc]" -map v frame%08d.png -map s -c copy -f rawvideo subcc.bin
```

## 26.12 libcdio# TOC

Audio-CD input device based on libcdio.

To enable this input device during configuration you need libcdio installed on your system. It requires the configure option `--enable-libcdio`.

This device allows playing and grabbing from an Audio-CD.

For example to copy with `ffmpeg` the entire Audio-CD in `/dev/sr0`, you may run the command:

```
ffmpeg -f libcdio -i /dev/sr0 cd.wav
```

### 26.12.1 Options# TOC

`speed`

Set drive reading speed. Default value is 0.

The speed is specified CD-ROM speed units. The speed is set through the libcdio `cdio_cddap_speed_set` function. On many CD-ROM drives, specifying a value too large will result in using the fastest speed.

`paranoia_mode`

Set paranoia recovery mode flags. It accepts one of the following values:

```
'disable'  
'verify'  
'overlap'  
'neverskip'  
'full'
```

Default value is `'disable'`.



For more information about the available recovery modes, consult the paranoia project documentation.

## 26.13 libdc1394# TOC

IIDC1394 input device, based on libdc1394 and libraw1394.

Requires the configure option `--enable-libdc1394`.

## 26.14 openal# TOC

The OpenAL input device provides audio capture on all systems with a working OpenAL 1.1 implementation.

To enable this input device during configuration, you need OpenAL headers and libraries installed on your system, and need to configure FFmpeg with `--enable-openal`.

OpenAL headers and libraries should be provided as part of your OpenAL implementation, or as an additional download (an SDK). Depending on your installation you may need to specify additional flags via the `--extra-cflags` and `--extra-ldflags` for allowing the build system to locate the OpenAL headers and libraries.

An incomplete list of OpenAL implementations follows:

### Creative

The official Windows implementation, providing hardware acceleration with supported devices and software fallback. See <http://openal.org/>.

### OpenAL Soft

Portable, open source (LGPL) software implementation. Includes backends for the most common sound APIs on the Windows, Linux, Solaris, and BSD operating systems. See <http://kcat.strangesoft.net/openal.html>.

### Apple

OpenAL is part of Core Audio, the official Mac OS X Audio interface. See <http://developer.apple.com/technologies/mac/audio-and-video.html>

This device allows one to capture from an audio input device handled through OpenAL.

You need to specify the name of the device to capture in the provided filename. If the empty string is provided, the device will automatically select the default device. You can get the list of the supported devices by using the option *list\_devices*.

## 26.14.1 Options# TOC

`channels`

Set the number of channels in the captured audio. Only the values 1 (monaural) and 2 (stereo) are currently supported. Defaults to 2.

`sample_size`

Set the sample size (in bits) of the captured audio. Only the values 8 and 16 are currently supported. Defaults to 16.

`sample_rate`

Set the sample rate (in Hz) of the captured audio. Defaults to 44.1k.

`list_devices`

If set to true, print a list of devices and exit. Defaults to false.

## 26.14.2 Examples# TOC

Print the list of OpenAL supported devices and exit:

```
$ ffmpeg -list_devices true -f openal -i dummy out.ogg
```

Capture from the OpenAL device DR-BT101 via PulseAudio:

```
$ ffmpeg -f openal -i 'DR-BT101 via PulseAudio' out.ogg
```

Capture from the default device (note the empty string "" as filename):

```
$ ffmpeg -f openal -i '' out.ogg
```

Capture from two devices simultaneously, writing to two different files, within the same `ffmpeg` command:

```
$ ffmpeg -f openal -i 'DR-BT101 via PulseAudio' out1.ogg -f openal -i 'ALSA Default' out2.ogg
```

Note: not all OpenAL implementations support multiple simultaneous capture - try the latest OpenAL Soft if the above does not work.

## 26.15 oss# TOC

Open Sound System input device.

The filename to provide to the input device is the device node representing the OSS input device, and is usually set to `/dev/dsp`.

For example to grab from `/dev/dsp` using `ffmpeg` use the command:

```
ffmpeg -f oss -i /dev/dsp /tmp/oss.wav
```

For more information about OSS see: <http://manuals.opensound.com/usersguide/dsp.html>

### **26.15.1 Options# TOC**

`sample_rate`

Set the sample rate in Hz. Default is 48000.

`channels`

Set the number of channels. Default is 2.

### **26.16 pulse# TOC**

PulseAudio input device.

To enable this output device you need to configure FFmpeg with `--enable-libpulse`.

The filename to provide to the input device is a source device or the string "default"

To list the PulseAudio source devices and their properties you can invoke the command `pactl list sources`.

More information about PulseAudio can be found on <http://www.pulseaudio.org>.

### **26.16.1 Options# TOC**

`server`

Connect to a specific PulseAudio server, specified by an IP address. Default server is used when not provided.

`name`

Specify the application name PulseAudio will use when showing active clients, by default it is the `LIBAVFORMAT_IDENT` string.

`stream_name`

Specify the stream name PulseAudio will use when showing active streams, by default it is "record".

`sample_rate`

Specify the samplerate in Hz, by default 48kHz is used.

channels

Specify the channels in use, by default 2 (stereo) is set.

frame\_size

Specify the number of bytes per frame, by default it is set to 1024.

fragment\_size

Specify the minimal buffering fragment in PulseAudio, it will affect the audio latency. By default it is unset.

wallclock

Set the initial PTS using the current time. Default is 1.

## 26.16.2 Examples# TOC

Record a stream from default device:

```
ffmpeg -f pulse -i default /tmp/pulse.wav
```

## 26.17 qtkit# TOC

QTKit input device.

The filename passed as input is parsed to contain either a device name or index. The device index can also be given by using -video\_device\_index. A given device index will override any given device name. If the desired device consists of numbers only, use -video\_device\_index to identify it. The default device will be chosen if an empty string or the device name "default" is given. The available devices can be enumerated by using -list\_devices.

```
ffmpeg -f qtkit -i "0" out.mpg
```

```
ffmpeg -f qtkit -video_device_index 0 -i "" out.mpg
```

```
ffmpeg -f qtkit -i "default" out.mpg
```

```
ffmpeg -f qtkit -list_devices true -i ""
```

### 26.17.1 Options# TOC

frame\_rate

Set frame rate. Default is 30.

`list_devices`

If set to `true`, print a list of devices and exit. Default is `false`.

`video_device_index`

Select the video device by index for devices with the same name (starts at 0).

## 26.18 sndio# TOC

sndio input device.

To enable this input device during configuration you need `libsndio` installed on your system.

The filename to provide to the input device is the device node representing the sndio input device, and is usually set to `/dev/audio0`.

For example to grab from `/dev/audio0` using `ffmpeg` use the command:

```
ffmpeg -f sndio -i /dev/audio0 /tmp/oss.wav
```

### 26.18.1 Options# TOC

`sample_rate`

Set the sample rate in Hz. Default is 48000.

`channels`

Set the number of channels. Default is 2.

## 26.19 video4linux2, v4l2# TOC

Video4Linux2 input video device.

"v4l2" can be used as alias for "video4linux2".

If FFmpeg is built with `v4l-utils` support (by using the `--enable-libv4l2` configure option), it is possible to use it with the `-use_libv4l2` input device option.

The name of the device to grab is a file device node, usually Linux systems tend to automatically create such nodes when the device (e.g. an USB webcam) is plugged into the system, and has a name of the kind `/dev/videoN`, where *N* is a number associated to the device.

Video4Linux2 devices usually support a limited set of *widthxheight* sizes and frame rates. You can check which are supported using `-list_formats all` for Video4Linux2 devices. Some devices, like TV cards, support one or more standards. It is possible to list all the supported standards using `-list_standards all`.

The time base for the timestamps is 1 microsecond. Depending on the kernel version and configuration, the timestamps may be derived from the real time clock (origin at the Unix Epoch) or the monotonic clock (origin usually at boot time, unaffected by NTP or manual changes to the clock). The `-timestamps abs` or `-ts abs` option can be used to force conversion into the real time clock.

Some usage examples of the video4linux2 device with `ffmpeg` and `ffplay`:

- List supported formats for a video4linux2 device:

```
ffplay -f video4linux2 -list_formats all /dev/video0
```

- Grab and show the input of a video4linux2 device:

```
ffplay -f video4linux2 -framerate 30 -video_size hd720 /dev/video0
```

- Grab and record the input of a video4linux2 device, leave the frame rate and size as previously set:

```
ffmpeg -f video4linux2 -input_format mjpeg -i /dev/video0 out.mpeg
```

For more information about Video4Linux, check <http://linuxtv.org/>.

### 26.19.1 Options# TOC

`standard`

Set the standard. Must be the name of a supported standard. To get a list of the supported standards, use the `list_standards` option.

`channel`

Set the input channel number. Default to -1, which means using the previously selected channel.

`video_size`

Set the video frame size. The argument must be a string in the form *WIDTHxHEIGHT* or a valid size abbreviation.

`pixel_format`

Select the pixel format (only valid for raw video input).

`input_format`

Set the preferred pixel format (for raw video) or a codec name. This option allows one to select the input format, when several are available.

`framerate`

Set the preferred video frame rate.

`list_formats`

List available formats (supported pixel formats, codecs, and frame sizes) and exit.

Available values are:

‘all’

Show all available (compressed and non-compressed) formats.

‘raw’

Show only raw video (non-compressed) formats.

‘compressed’

Show only compressed formats.

`list_standards`

List supported standards and exit.

Available values are:

‘all’

Show all supported standards.

`timestamps, ts`

Set type of timestamps for grabbed frames.

Available values are:

‘default’

Use timestamps from the kernel.

‘abs’

Use absolute timestamps (wall clock).

‘mono2abs’

Force conversion from monotonic to absolute timestamps.

Default value is `default`.

`use_libv4l2`

Use libv4l2 (v4l-utils) conversion functions. Default is 0.

## 26.20 vfwcap# TOC

VfW (Video for Windows) capture input device.

The filename passed as input is the capture driver number, ranging from 0 to 9. You may use "list" as filename to print a list of drivers. Any other filename will be interpreted as device number 0.

### 26.20.1 Options# TOC

`video_size`

Set the video frame size.

`framerate`

Set the grabbing frame rate. Default value is `ntsc`, corresponding to a frame rate of 30000/1001.

## 26.21 x11grab# TOC

X11 video input device.

To enable this input device during configuration you need libxcb installed on your system. It will be automatically detected during configuration.

Alternatively, the configure option `--enable-x11grab` exists for legacy Xlib users.

This device allows one to capture a region of an X11 display.

The filename passed as input has the syntax:

```
[hostname]:display_number.screen_number[+x_offset,y_offset]
```

*hostname:display\_number.screen\_number* specifies the X11 display name of the screen to grab from. *hostname* can be omitted, and defaults to "localhost". The environment variable `DISPLAY` contains the default display name.

*x\_offset* and *y\_offset* specify the offsets of the grabbed area with respect to the top-left border of the X11 screen. They default to 0.

Check the X11 documentation (e.g. `man X`) for more detailed information.



Use the `xdpinfo` program for getting basic information about the properties of your X11 display (e.g. `grep` for "name" or "dimensions").

For example to grab from `:0.0` using `ffmpeg`:

```
ffmpeg -f x11grab -framerate 25 -video_size cif -i :0.0 out.mpg
```

Grab at position 10,20:

```
ffmpeg -f x11grab -framerate 25 -video_size cif -i :0.0+10,20 out.mpg
```

## 26.21.1 Options# TOC

`draw_mouse`

Specify whether to draw the mouse pointer. A value of 0 specify not to draw the pointer. Default value is 1.

`follow_mouse`

Make the grabbed area follow the mouse. The argument can be `centered` or a number of pixels *PIXELS*.

When it is specified with "centered", the grabbing region follows the mouse pointer and keeps the pointer at the center of region; otherwise, the region follows only when the mouse pointer reaches within *PIXELS* (greater than zero) to the edge of region.

For example:

```
ffmpeg -f x11grab -follow_mouse centered -framerate 25 -video_size cif -i :0.0 out.mpg
```

To follow only when the mouse pointer reaches within 100 pixels to edge:

```
ffmpeg -f x11grab -follow_mouse 100 -framerate 25 -video_size cif -i :0.0 out.mpg
```

`framerate`

Set the grabbing frame rate. Default value is `ntsc`, corresponding to a frame rate of `30000/1001`.

`show_region`

Show grabbed region on screen.

If `show_region` is specified with 1, then the grabbing region will be indicated on screen. With this option, it is easy to know what is being grabbed if only a portion of the screen is grabbed.

`region_border`

Set the region border thickness if `-show_region 1` is used. Range is 1 to 128 and default is 3 (XCB-based x11grab only).

For example:

```
ffmpeg -f x11grab -show_region 1 -framerate 25 -video_size cif -i :0.0+10,20 out.mpg
```

With *follow\_mouse*:

```
ffmpeg -f x11grab -follow_mouse centered -show_region 1 -framerate 25 -video_size cif -i :0.0 out.mpg
```

`video_size`

Set the video frame size. Default value is vga.

`use_shm`

Use the MIT-SHM extension for shared memory. Default value is 1. It may be necessary to disable it for remote displays (legacy x11grab only).

### 26.21.2 *grab\_x grab\_y* AVOption# TOC

The syntax is:

```
-grab_x x_offset -grab_y y_offset
```

Set the grabbing region coordinates. They are expressed as offset from the top left corner of the X11 window. The default value is 0.

## 27 Output Devices# TOC

Output devices are configured elements in FFmpeg that can write multimedia data to an output device attached to your system.

When you configure your FFmpeg build, all the supported output devices are enabled by default. You can list all available ones using the configure option "`--list-outdevs`".

You can disable all the output devices using the configure option "`--disable-outdevs`", and selectively enable an output device using the option "`--enable-outdev=OUTDEV`", or you can disable a particular input device using the option "`--disable-outdev=OUTDEV`".

The option "`-devices`" of the ff\* tools will display the list of enabled output devices.

A description of the currently available output devices follows.

## 27.1 alsa# TOC

ALSA (Advanced Linux Sound Architecture) output device.

### 27.1.1 Examples# TOC

- Play a file on default ALSA device:

```
ffmpeg -i INPUT -f alsa default
```

- Play a file on soundcard 1, audio device 7:

```
ffmpeg -i INPUT -f alsa hw:1,7
```

## 27.2 caca# TOC

CACA output device.

This output device allows one to show a video stream in CACA window. Only one CACA window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need to configure FFmpeg with `--enable-libcaca`. libcaca is a graphics library that outputs text instead of pixels.

For more information about libcaca, check: <http://caca.zoy.org/wiki/libcaca>

### 27.2.1 Options# TOC

`window_title`

Set the CACA window title, if not specified default to the filename specified for the output device.

`window_size`

Set the CACA window size, can be a string of the form *widthxheight* or a video size abbreviation. If not specified it defaults to the size of the input video.

`driver`

Set display driver.

`algorithm`

Set dithering algorithm. Dithering is necessary because the picture being rendered has usually far more colours than the available palette. The accepted values are listed with `-list_dither_algorithms`.

antialias

Set antialias method. Antialiasing smoothens the rendered image and avoids the commonly seen staircase effect. The accepted values are listed with `-list_dither antialiases`.

charset

Set which characters are going to be used when rendering text. The accepted values are listed with `-list_dither charsets`.

color

Set color to be used when rendering text. The accepted values are listed with `-list_dither colors`.

list\_drivers

If set to true, print a list of available drivers and exit.

list\_dither

List available dither options related to the argument. The argument must be one of algorithms, antialiases, charsets, colors.

### 27.2.2 Examples# TOC

- The following command shows the ffmpeg output is an CACA window, forcing its size to 80x25:

```
ffmpeg -i INPUT -vcodec rawvideo -pix_fmt rgb24 -window_size 80x25 -f caca -
```

- Show the list of available drivers and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_drivers true -
```

- Show the list of available dither colors and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_dither colors -
```

### 27.3 decklink# TOC

The decklink output device provides playback capabilities for Blackmagic DeckLink devices.

To enable this output device, you need the Blackmagic DeckLink SDK and you need to configure with the appropriate `--extra-cflags` and `--extra-ldflags`. On Windows, you need to run the IDL files through `widl`.

DeckLink is very picky about the formats it supports. Pixel format is always uyvy422, framerate and video size must be determined for your device with `-list_formats 1`. Audio sample rate is always 48 kHz.

### 27.3.1 Options# TOC

`list_devices`

If set to `true`, print a list of devices and exit. Defaults to `false`.

`list_formats`

If set to `true`, print a list of supported formats and exit. Defaults to `false`.

`preroll`

Amount of time to preroll video in seconds. Defaults to 0.5.

### 27.3.2 Examples# TOC

- List output devices:

```
ffmpeg -i test.avi -f decklink -list_devices 1 dummy
```

- List supported formats:

```
ffmpeg -i test.avi -f decklink -list_formats 1 'DeckLink Mini Monitor'
```

- Play video clip:

```
ffmpeg -i test.avi -f decklink -pix_fmt uyvy422 'DeckLink Mini Monitor'
```

- Play video clip with non-standard framerate or video size:

```
ffmpeg -i test.avi -f decklink -pix_fmt uyvy422 -s 720x486 -r 24000/1001 'DeckLink Mini Monitor'
```

## 27.4 fbdev# TOC

Linux framebuffer output device.

The Linux framebuffer is a graphic hardware-independent abstraction layer to show graphics on a computer monitor, typically on the console. It is accessed through a file device node, usually `/dev/fb0`.

For more detailed information read the file `Documentation/fb/framebuffer.txt` included in the Linux source tree.

### 27.4.1 Options# TOC

`xoffset`

`yoffset`

Set x/y coordinate of top left corner. Default is 0.

## 27.4.2 Examples# TOC

Play a file on framebuffer device `/dev/fb0`. Required pixel format depends on current framebuffer settings.

```
ffmpeg -re -i INPUT -vcodec rawvideo -pix_fmt bgra -f fbdev /dev/fb0
```

See also <http://linux-fbdev.sourceforge.net/>, and `fbset(1)`.

## 27.5 opengl# TOC

OpenGL output device.

To enable this output device you need to configure FFmpeg with `--enable-opengl`.

This output device allows one to render to OpenGL context. Context may be provided by application or default SDL window is created.

When device renders to external context, application must implement handlers for following messages:

AV\_DEV\_TO\_APP\_CREATE\_WINDOW\_BUFFER - create OpenGL context on current thread.

AV\_DEV\_TO\_APP\_PREPARE\_WINDOW\_BUFFER - make OpenGL context current.

AV\_DEV\_TO\_APP\_DISPLAY\_WINDOW\_BUFFER - swap buffers.

AV\_DEV\_TO\_APP\_DESTROY\_WINDOW\_BUFFER - destroy OpenGL context. Application is also required to inform a device about current resolution by sending AV\_APP\_TO\_DEV\_WINDOW\_SIZE message.

### 27.5.1 Options# TOC

`background`

Set background color. Black is a default.

`no_window`

Disables default SDL window when set to non-zero value. Application must provide OpenGL context and both `window_size_cb` and `window_swap_buffers_cb` callbacks when set.

`window_title`

Set the SDL window title, if not specified default to the filename specified for the output device. Ignored when `no_window` is set.

`window_size`

Set preferred window size, can be a string of the form `widthxheight` or a video size abbreviation. If not specified it defaults to the size of the input video, downscaled according to the aspect ratio. Mostly usable when `no_window` is not set.

## 27.5.2 Examples# TOC

Play a file on SDL window using OpenGL rendering:

```
ffmpeg -i INPUT -f opengl "window title"
```

## 27.6 oss# TOC

OSS (Open Sound System) output device.

## 27.7 pulse# TOC

PulseAudio output device.

To enable this output device you need to configure FFmpeg with `--enable-libpulse`.

More information about PulseAudio can be found on <http://www.pulseaudio.org>

### 27.7.1 Options# TOC

`server`

Connect to a specific PulseAudio server, specified by an IP address. Default server is used when not provided.

`name`

Specify the application name PulseAudio will use when showing active clients, by default it is the `LIBAVFORMAT_IDENT` string.

`stream_name`

Specify the stream name PulseAudio will use when showing active streams, by default it is set to the specified output name.

`device`

Specify the device to use. Default device is used when not provided. List of output devices can be obtained with command `pactl list sinks`.

`buffer_size`

`buffer_duration`

Control the size and duration of the PulseAudio buffer. A small buffer gives more control, but requires more frequent updates.

`buffer_size` specifies size in bytes while `buffer_duration` specifies duration in milliseconds.

When both options are provided then the highest value is used (duration is recalculated to bytes using stream parameters). If they are set to 0 (which is default), the device will use the default PulseAudio duration value. By default PulseAudio set buffer duration to around 2 seconds.

`prebuf`

Specify pre-buffering size in bytes. The server does not start with playback before at least `prebuf` bytes are available in the buffer. By default this option is initialized to the same value as `buffer_size` or `buffer_duration` (whichever is bigger).

`minreq`

Specify minimum request size in bytes. The server does not request less than `minreq` bytes from the client, instead waits until the buffer is free enough to request more bytes at once. It is recommended to not set this option, which will initialize this to a value that is deemed sensible by the server.

## 27.7.2 Examples# TOC

Play a file on default device on default server:

```
ffmpeg -i INPUT -f pulse "stream name"
```

## 27.8 sdl# TOC

SDL (Simple DirectMedia Layer) output device.

This output device allows one to show a video stream in an SDL window. Only one SDL window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need `libsdl` installed on your system when configuring your build.

For more information about SDL, check: <http://www.libsdl.org/>

### 27.8.1 Options# TOC

`window_title`

Set the SDL window title, if not specified default to the filename specified for the output device.

`icon_title`

Set the name of the iconified SDL window, if not specified it is set to the same value of *window\_title*.

`window_size`



Set the SDL window size, can be a string of the form *widthxheight* or a video size abbreviation. If not specified it defaults to the size of the input video, downscaled according to the aspect ratio.

`window_fullscreen`

Set fullscreen mode when non-zero value is provided. Default value is zero.

## 27.8.2 Interactive commands# TOC

The window created by the device can be controlled through the following interactive commands.

`q`, `ESC`

Quit the device immediately.

## 27.8.3 Examples# TOC

The following command shows the `ffmpeg` output is an SDL window, forcing its size to the `qcif` format:

```
ffmpeg -i INPUT -vcodec rawvideo -pix_fmt yuv420p -window_size qcif -f sdl "SDL output"
```

## 27.9 sndio# TOC

`sndio` audio output device.

## 27.10 xv# TOC

`XV` (XVideo) output device.

This output device allows one to show a video stream in a X Window System window.

### 27.10.1 Options# TOC

`display_name`

Specify the hardware display name, which determines the display and communications domain to be used.

The display name or `DISPLAY` environment variable can be a string in the format *hostname[:number[.screen\_number]]*.

*hostname* specifies the name of the host machine on which the display is physically attached. *number* specifies the number of the display server on that host machine. *screen\_number* specifies the screen to be used on that server.

If unspecified, it defaults to the value of the `DISPLAY` environment variable.

For example, `dual-headed:0.1` would specify screen 1 of display 0 on the machine named “dual-headed”.

Check the X11 specification for more detailed information about the display name format.

`window_id`

When set to non-zero value then device doesn't create new window, but uses existing one with provided *window\_id*. By default this options is set to zero and device creates its own window.

`window_size`

Set the created window size, can be a string of the form *widthxheight* or a video size abbreviation. If not specified it defaults to the size of the input video. Ignored when *window\_id* is set.

`window_x`

`window_y`

Set the X and Y window offsets for the created window. They are both set to 0 by default. The values may be ignored by the window manager. Ignored when *window\_id* is set.

`window_title`

Set the window title, if not specified default to the filename specified for the output device. Ignored when *window\_id* is set.

For more information about XVideo see <http://www.x.org/>.

## 27.10.2 Examples# TOC

- Decode, display and encode video input with `ffmpeg` at the same time:

```
ffmpeg -i INPUT OUTPUT -f xv display
```

- Decode and display the input video to multiple X11 windows:

```
ffmpeg -i INPUT -f xv normal -vf negate -f xv negated
```

## 28 Resampler Options# TOC

The audio resampler supports the following named options.

Options may be set by specifying *-option value* in the FFmpeg tools, *option=value* for the `aresample` filter, by setting the value explicitly in the `SwrContext` options or using the `libavutil/opt.h` API for programmatic use.

`ich, in_channel_count`

Set the number of input channels. Default value is 0. Setting this value is not mandatory if the corresponding channel layout `in_channel_layout` is set.

`och, out_channel_count`

Set the number of output channels. Default value is 0. Setting this value is not mandatory if the corresponding channel layout `out_channel_layout` is set.

`uch, used_channel_count`

Set the number of used input channels. Default value is 0. This option is only used for special remapping.

`isr, in_sample_rate`

Set the input sample rate. Default value is 0.

`osr, out_sample_rate`

Set the output sample rate. Default value is 0.

`isf, in_sample_fmt`

Specify the input sample format. It is set by default to `none`.

`osf, out_sample_fmt`

Specify the output sample format. It is set by default to `none`.

`tsf, internal_sample_fmt`

Set the internal sample format. Default value is `none`. This will automatically be chosen when it is not explicitly set.

`icl, in_channel_layout`

`ocl, out_channel_layout`

Set the input/output channel layout.

See (ffmpeg-utils)the Channel Layout section in the ffmpeg-utils(1) manual for the required syntax.

`clev, center_mix_level`

Set the center mix level. It is a value expressed in deciBel, and must be in the interval `[-32,32]`.

`slev, surround_mix_level`

Set the surround mix level. It is a value expressed in deciBel, and must be in the interval `[-32,32]`.

lfe\_mix\_level

Set LFE mix into non LFE level. It is used when there is a LFE input but no LFE output. It is a value expressed in deciBel, and must be in the interval [-32,32].

rmvol, rematrix\_volume

Set rematrix volume. Default value is 1.0.

rematrix\_maxval

Set maximum output value for rematrixing. This can be used to prevent clipping vs. preventing volume reduction. A value of 1.0 prevents clipping.

flags, swr\_flags

Set flags used by the converter. Default value is 0.

It supports the following individual flags:

res

force resampling, this flag forces resampling to be used even when the input and output sample rates match.

dither\_scale

Set the dither scale. Default value is 1.

dither\_method

Set dither method. Default value is 0.

Supported values:

'rectangular'

select rectangular dither

'triangular'

select triangular dither

'triangular\_hp'

select triangular dither with high pass

'lipshitz'

select lipshitz noise shaping dither

‘shibata’

select shibata noise shaping dither

‘low\_shibata’

select low shibata noise shaping dither

‘high\_shibata’

select high shibata noise shaping dither

‘f\_weighted’

select f-weighted noise shaping dither

‘modified\_e\_weighted’

select modified-e-weighted noise shaping dither

‘improved\_e\_weighted’

select improved-e-weighted noise shaping dither

resampler

Set resampling engine. Default value is swr.

Supported values:

‘swr’

select the native SW Resampler; filter options precision and cheby are not applicable in this case.

‘soxr’

select the SoX Resampler (where available); compensation, and filter options filter\_size, phase\_shift, filter\_type & kaiser\_beta, are not applicable in this case.

filter\_size

For swr only, set resampling filter size, default value is 32.

phase\_shift

For swr only, set resampling phase shift, default value is 10, and must be in the interval [0,30].

`linear_interp`

Use Linear Interpolation if set to 1, default value is 0.

`cutoff`

Set cutoff frequency (swr: 6dB point; soxr: 0dB point) ratio; must be a float value between 0 and 1. Default value is 0.97 with swr, and 0.91 with soxr (which, with a sample-rate of 44100, preserves the entire audio band to 20kHz).

`precision`

For soxr only, the precision in bits to which the resampled signal will be calculated. The default value of 20 (which, with suitable dithering, is appropriate for a destination bit-depth of 16) gives SoX's 'High Quality'; a value of 28 gives SoX's 'Very High Quality'.

`cheby`

For soxr only, selects passband rolloff none (Chebyshev) & higher-precision approximation for 'irrational' ratios. Default value is 0.

`async`

For swr only, simple 1 parameter audio sync to timestamps using stretching, squeezing, filling and trimming. Setting this to 1 will enable filling and trimming, larger values represent the maximum amount in samples that the data may be stretched or squeezed for each second. Default value is 0, thus no compensation is applied to make the samples match the audio timestamps.

`first_pts`

For swr only, assume the first pts should be this value. The time unit is 1 / sample rate. This allows for padding/trimming at the start of stream. By default, no assumption is made about the first frame's expected pts, so no padding or trimming is done. For example, this could be set to 0 to pad the beginning with silence if an audio stream starts after the video stream or to trim any samples with a negative pts due to encoder delay.

`min_comp`

For swr only, set the minimum difference between timestamps and audio data (in seconds) to trigger stretching/squeezing/filling or trimming of the data to make it match the timestamps. The default is that stretching/squeezing/filling and trimming is disabled (`min_comp = FLT_MAX`).

`min_hard_comp`

For swr only, set the minimum difference between timestamps and audio data (in seconds) to trigger adding/dropping samples to make it match the timestamps. This option effectively is a threshold to select between hard (trim/fill) and soft (squeeze/stretch) compensation. Note that all compensation is

by default disabled through `min_comp`. The default is 0.1.

`comp_duration`

For swr only, set duration (in seconds) over which data is stretched/squeezed to make it match the timestamps. Must be a non-negative double float value, default value is 1.0.

`max_soft_comp`

For swr only, set maximum factor by which data is stretched/squeezed to make it match the timestamps. Must be a non-negative double float value, default value is 0.

`matrix_encoding`

Select matrixed stereo encoding.

It accepts the following values:

`'none'`

select none

`'dolby'`

select Dolby

`'dplii'`

select Dolby Pro Logic II

Default value is none.

`filter_type`

For swr only, select resampling filter type. This only affects resampling operations.

It accepts the following values:

`'cubic'`

select cubic

`'blackman_nuttall'`

select Blackman Nuttall Windowed Sinc

`'kaiser'`

select Kaiser Windowed Sinc

kaiser\_beta

For swr only, set Kaiser Window Beta value. Must be an integer in the interval [2,16], default value is 9.

output\_sample\_bits

For swr only, set number of used output sample bits for dithering. Must be an integer in the interval [0,64], default value is 0, which means it's not used.

## 29 Scaler Options# TOC

The video scaler supports the following named options.

Options may be set by specifying *-option value* in the FFmpeg tools. For programmatic use, they can be set explicitly in the SwsContext options or through the libavutil/opt.h API.

sws\_flags

Set the scaler flags. This is also used to set the scaling algorithm. Only a single algorithm should be selected.

It accepts the following values:

'fast\_bilinear'

Select fast bilinear scaling algorithm.

'bilinear'

Select bilinear scaling algorithm.

'bicubic'

Select bicubic scaling algorithm.

'experimental'

Select experimental scaling algorithm.

'neighbor'

Select nearest neighbor rescaling algorithm.

'area'



Select averaging area rescaling algorithm.

`'bicublin'`

Select bicubic scaling algorithm for the luma component, bilinear for chroma components.

`'gauss'`

Select Gaussian rescaling algorithm.

`'sinc'`

Select sinc rescaling algorithm.

`'lanczos'`

Select lanczos rescaling algorithm.

`'spline'`

Select natural bicubic spline rescaling algorithm.

`'print_info'`

Enable printing/debug logging.

`'accurate_rnd'`

Enable accurate rounding.

`'full_chroma_int'`

Enable full chroma interpolation.

`'full_chroma_inp'`

Select full chroma input.

`'bitexact'`

Enable bitexact output.

`srcw`

Set source width.

`srch`

Set source height.

`dstw`

Set destination width.

`dsth`

Set destination height.

`src_format`

Set source pixel format (must be expressed as an integer).

`dst_format`

Set destination pixel format (must be expressed as an integer).

`src_range`

Select source range.

`dst_range`

Select destination range.

`param0, param1`

Set scaling algorithm parameters. The specified values are specific of some scaling algorithms and ignored by others. The specified values are floating point number values.

`sws_dither`

Set the dithering algorithm. Accepts one of the following values. Default value is 'auto'.

'auto'

automatic choice

'none'

no dithering

'bayer'

bayer dither

'ed'

error diffusion dither

`'a_dither'`

arithmetic dither, based using addition

`'x_dither'`

arithmetic dither, based using xor (more random/less apparent patterning than `a_dither`).

`alphablend`

Set the alpha blending to use when the input has alpha but the output does not. Default value is `'none'`.

`'uniform_color'`

Blend onto a uniform background color

`'checkerboard'`

Blend onto a checkerboard

`'none'`

No blending

## 30 Filtering Introduction# TOC

Filtering in FFmpeg is enabled through the libavfilter library.

In libavfilter, a filter can have multiple inputs and multiple outputs. To illustrate the sorts of things that are possible, we consider the following filtergraph.

```

                        [main]
input --> split -----> overlay --> output
      |
      |[tmp]                [flip]|
      +-----> crop --> vflip -----+
```

This filtergraph splits the input stream in two streams, then sends one stream through the crop filter and the vflip filter, before merging it back with the other stream by overlaying it on top. You can use the following command to achieve this:

```
ffmpeg -i INPUT -vf "split [main][tmp]; [tmp] crop=iw:ih/2:0:0, vflip [flip]; [main][flip] overlay=0:H/2" OUTPUT
```

The result will be that the top half of the video is mirrored onto the bottom half of the output video.

Filters in the same linear chain are separated by commas, and distinct linear chains of filters are separated by semicolons. In our example, *crop,vflip* are in one linear chain, *split* and *overlay* are separately in another. The points where the linear chains join are labelled by names enclosed in square brackets. In the example, the split filter generates two outputs that are associated to the labels *[main]* and *[tmp]*.

The stream sent to the second output of *split*, labelled as *[tmp]*, is processed through the *crop* filter, which crops away the lower half part of the video, and then vertically flipped. The *overlay* filter takes in input the first unchanged output of the split filter (which was labelled as *[main]*), and overlay on its lower half the output generated by the *crop,vflip* filterchain.

Some filters take in input a list of parameters: they are specified after the filter name and an equal sign, and are separated from each other by a colon.

There exist so-called *source filters* that do not have an audio/video input, and *sink filters* that will not have audio/video output.

## 31 graph2dot# TOC

The `graph2dot` program included in the FFmpeg `tools` directory can be used to parse a filtergraph description and issue a corresponding textual representation in the dot language.

Invoke the command:

```
graph2dot -h
```

to see how to use `graph2dot`.

You can then pass the dot description to the `dot` program (from the graphviz suite of programs) and obtain a graphical representation of the filtergraph.

For example the sequence of commands:

```
echo GRAPH_DESCRIPTION | \
tools/graph2dot -o graph.tmp && \
dot -Tpng graph.tmp -o graph.png && \
display graph.png
```

can be used to create and display an image representing the graph described by the *GRAPH\_DESCRIPTION* string. Note that this string must be a complete self-contained graph, with its inputs and outputs explicitly defined. For example if your command line is of the form:

```
ffmpeg -i infile -vf scale=640:360 outfile
```

your *GRAPH\_DESCRIPTION* string will need to be of the form:

```
nullsrc,scale=640:360,nullsink
```

you may also need to set the *nullsrc* parameters and add a *format* filter in order to simulate a specific input file.

## 32 Filtergraph description# TOC

A filtergraph is a directed graph of connected filters. It can contain cycles, and there can be multiple links between a pair of filters. Each link has one input pad on one side connecting it to one filter from which it takes its input, and one output pad on the other side connecting it to one filter accepting its output.

Each filter in a filtergraph is an instance of a filter class registered in the application, which defines the features and the number of input and output pads of the filter.

A filter with no input pads is called a "source", and a filter with no output pads is called a "sink".

### 32.1 Filtergraph syntax# TOC

A filtergraph has a textual representation, which is recognized by the `-filter/-vf/-af` and `-filter_complex` options in `ffmpeg` and `-vf/-af` in `ffplay`, and by the `avfilter_graph_parse_ptr()` function defined in `libavfilter/avfilter.h`.

A filterchain consists of a sequence of connected filters, each one connected to the previous one in the sequence. A filterchain is represented by a list of `","`-separated filter descriptions.

A filtergraph consists of a sequence of filterchains. A sequence of filterchains is represented by a list of `";"`-separated filterchain descriptions.

A filter is represented by a string of the form:

`[in_link_1]...[in_link_N]filter_name=arguments[out_link_1]...[out_link_M]`

*filter\_name* is the name of the filter class of which the described filter is an instance of, and has to be the name of one of the filter classes registered in the program. The name of the filter class is optionally followed by a string `"=arguments"`.

*arguments* is a string which contains the parameters used to initialize the filter instance. It may have one of two forms:

- A `':'`-separated list of *key=value* pairs.
- A `':'`-separated list of *value*. In this case, the keys are assumed to be the option names in the order they are declared. E.g. the `fade` filter declares three options in this order – `type`, `start_frame` and `nb_frames`. Then the parameter list `in:0:30` means that the value *in* is assigned to the option `type`, `0` to `start_frame` and `30` to `nb_frames`.
- A `':'`-separated list of mixed direct *value* and long *key=value* pairs. The direct *value* must precede the *key=value* pairs, and follow the same constraints order of the previous point. The following *key=value* pairs can be set in any preferred order.

If the option value itself is a list of items (e.g. the `format` filter takes a list of pixel formats), the items in the list are usually separated by `'|'`.

The list of arguments can be quoted using the character `'` as initial and ending mark, and the character `\` for escaping the characters within the quoted text; otherwise the argument string is considered terminated when the next special character (belonging to the set `[ ] = ; , '`) is encountered.

The name and arguments of the filter are optionally preceded and followed by a list of link labels. A link label allows one to name a link and associate it to a filter output or input pad. The preceding labels *in\_link\_1* ... *in\_link\_N*, are associated to the filter input pads, the following labels *out\_link\_1* ... *out\_link\_M*, are associated to the output pads.

When two link labels with the same name are found in the filtergraph, a link between the corresponding input and output pad is created.

If an output pad is not labelled, it is linked by default to the first unlabelled input pad of the next filter in the filterchain. For example in the filterchain

```
nullsrc, split[L1], [L2]overlay, nullsink
```

the `split` filter instance has two output pads, and the `overlay` filter instance two input pads. The first output pad of `split` is labelled "L1", the first input pad of `overlay` is labelled "L2", and the second output pad of `split` is linked to the second input pad of `overlay`, which are both unlabelled.

In a filter description, if the input label of the first filter is not specified, "in" is assumed; if the output label of the last filter is not specified, "out" is assumed.

In a complete filterchain all the unlabelled filter input and output pads must be connected. A filtergraph is considered valid if all the filter input and output pads of all the filterchains are connected.

Libavfilter will automatically insert scale filters where format conversion is required. It is possible to specify `sws_flags` for those automatically inserted scalers by prepending `sws_flags=flags;` to the filtergraph description.

Here is a BNF description of the filtergraph syntax:

```
NAME          ::= sequence of alphanumeric characters and '_'
LINKLABEL     ::= "[" NAME "]"
LINKLABELS    ::= LINKLABEL [LINKLABELS]
FILTER_ARGUMENTS ::= sequence of chars (possibly quoted)
FILTER        ::= [LINKLABELS] NAME ["=" FILTER_ARGUMENTS] [LINKLABELS]
FILTERCHAIN   ::= FILTER [ , FILTERCHAIN ]
FILTERGRAPH   ::= [sws_flags=flags;] FILTERCHAIN [ ; FILTERGRAPH ]
```

## 32.2 Notes on filtergraph escaping# TOC

Filtergraph description composition entails several levels of escaping. See (ffmpeg-utils)the "Quoting and escaping" section in the ffmpeg-utils(1) manual for more information about the employed escaping procedure.

A first level escaping affects the content of each filter option value, which may contain the special character `:` used to separate values, or one of the escaping characters `\` `'`.

A second level escaping affects the whole filter description, which may contain the escaping characters `\` `'` or the special characters `[ ] , ;` used by the filtergraph description.

Finally, when you specify a filtergraph on a shell commandline, you need to perform a third level escaping for the shell special characters contained within it.

For example, consider the following string to be embedded in the drawtext filter description `text` value:

```
this is a 'string': may contain one, or more, special characters
```

This string contains the `'` special escaping character, and the `:` special character, so it needs to be escaped in this way:

```
text=this is a \'string\': may contain one, or more, special characters
```

A second level of escaping is required when embedding the filter description in a filtergraph description, in order to escape all the filtergraph special characters. Thus the example above becomes:

```
drawtext=text=this is a \\\'string\\\': may contain one\\, or more\\, special characters
```

(note that in addition to the `\` `'` escaping special characters, also `,` needs to be escaped).

Finally an additional level of escaping is needed when writing the filtergraph description in a shell command, which depends on the escaping rules of the adopted shell. For example, assuming that `\` is special and needs to be escaped with another `\`, the previous string will finally result in:

```
-vf "drawtext=text=this is a \\\\\\\\'string\\\\\\\\\'\\\\\\\\: may contain one\\\\, or more\\\\, special characters"
```

## 33 Timeline editing# TOC

Some filters support a generic `enable` option. For the filters supporting timeline editing, this option can be set to an expression which is evaluated before sending a frame to the filter. If the evaluation is non-zero, the filter will be enabled, otherwise the frame will be sent unchanged to the next filter in the filtergraph.

The expression accepts the following values:

`'t'`

timestamp expressed in seconds, NAN if the input timestamp is unknown

`'n'`

sequential number of the input frame, starting from 0

`'pos'`

the position in the file of the input frame, NAN if unknown

`'w'`

`'h'`

width and height of the input frame if video

Additionally, these filters support an `enable` command that can be used to re-define the expression.

Like any other filtering option, the `enable` option follows the same rules.

For example, to enable a blur filter (`smartblur`) from 10 seconds to 3 minutes, and a curves filter starting at 3 seconds:

```
smartblur = enable='between(t,10,3*60)',  
curves    = enable='gte(t,3)' : preset=cross_process
```

## 34 Audio Filters# TOC

When you configure your FFmpeg build, you can disable any of the existing filters using `--disable-filters`. The configure output will show the audio filters included in your build.

Below is a description of the currently available audio filters.

### 34.1 acrossfade# TOC

Apply cross fade from one input audio stream to another input audio stream. The cross fade is applied for specified duration near the end of first stream.

The filter accepts the following options:

`nb_samples, ns`

Specify the number of samples for which the cross fade effect has to last. At the end of the cross fade effect the first input audio will be completely silent. Default is 44100.

`duration, d`

Specify the duration of the cross fade effect. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. By default the duration is determined by `nb_samples`. If set this option is used instead of `nb_samples`.

`overlap, o`

Should first stream end overlap with second stream start. Default is enabled.



curve1

Set curve for cross fade transition for first stream.

curve2

Set curve for cross fade transition for second stream.

For description of available curve types see afade filter description.

### 34.1.1 Examples# TOC

- Cross fade from one input to another:

```
ffmpeg -i first.flac -i second.flac -filter_complex acrossfade=d=10:c1=exp:c2=exp output.flac
```

- Cross fade from one input to another but without overlapping:

```
ffmpeg -i first.flac -i second.flac -filter_complex acrossfade=d=10:o=0:c1=exp:c2=exp output.flac
```

## 34.2 adelay# TOC

Delay one or more audio channels.

Samples in delayed channel are filled with silence.

The filter accepts the following option:

delays

Set list of delays in milliseconds for each channel separated by '|'. At least one delay greater than 0 should be provided. Unused delays will be silently ignored. If number of given delays is smaller than number of channels all remaining channels will not be delayed.

### 34.2.1 Examples# TOC

- Delay first channel by 1.5 seconds, the third channel by 0.5 seconds and leave the second channel (and any other channels that may be present) unchanged.

```
adelay=1500|0|500
```

## 34.3 aecho# TOC

Apply echoing to the input audio.

Echoes are reflected sound and can occur naturally amongst mountains (and sometimes large buildings) when talking or shouting; digital echo effects emulate this behaviour and are often used to help fill out the sound of a single instrument or vocal. The time difference between the original signal and the reflection is the `delay`, and the loudness of the reflected signal is the `decay`. Multiple echoes can have different delays and decays.

A description of the accepted parameters follows.

`in_gain`

Set input gain of reflected signal. Default is 0.6.

`out_gain`

Set output gain of reflected signal. Default is 0.3.

`delays`

Set list of time intervals in milliseconds between original signal and reflections separated by '|'. Allowed range for each delay is (0 - 90000.0]. Default is 1000.

`decays`

Set list of loudnesses of reflected signals separated by '|'. Allowed range for each decay is (0 - 1.0]. Default is 0.5.

### 34.3.1 Examples# TOC

- Make it sound as if there are twice as many instruments as are actually playing:

```
aecho=0.8:0.88:60:0.4
```

- If delay is very short, then it sound like a (metallic) robot playing music:

```
aecho=0.8:0.88:6:0.4
```

- A longer delay will sound like an open air concert in the mountains:

```
aecho=0.8:0.9:1000:0.3
```

- Same as above but with one more mountain:

```
aecho=0.8:0.9:1000|1800:0.3|0.25
```

## 34.4 aeval# TOC

Modify an audio signal according to the specified expressions.

This filter accepts one or more expressions (one for each channel), which are evaluated and used to modify a corresponding audio signal.

It accepts the following parameters:

`exprs`

Set the '|' -separated expressions list for each separate channel. If the number of input channels is greater than the number of expressions, the last specified expression is used for the remaining output channels.

`channel_layout, c`

Set output channel layout. If not specified, the channel layout is specified by the number of expressions. If set to 'same', it will use by default the same input channel layout.

Each expression in *exprs* can contain the following constants and functions:

`ch`

channel number of the current expression

`n`

number of the evaluated sample, starting from 0

`s`

sample rate

`t`

time of the evaluated sample expressed in seconds

`nb_in_channels`

`nb_out_channels`

input and output number of channels

`val(CH)`

the value of input channel with number *CH*

Note: this filter is slow. For faster processing you should use a dedicated filter.

### 34.4.1 Examples# TOC

- Half volume:

```
aeval=val(ch)/2:c=same
```

- Invert phase of the second channel:

```
aeval=val(0)|-val(1)
```

## 34.5 afade# TOC

Apply fade-in/out effect to input audio.

A description of the accepted parameters follows.

`type, t`

Specify the effect type, can be either `in` for fade-in, or `out` for a fade-out effect. Default is `in`.

`start_sample, ss`

Specify the number of the start sample for starting to apply the fade effect. Default is 0.

`nb_samples, ns`

Specify the number of samples for which the fade effect has to last. At the end of the fade-in effect the output audio will have the same volume as the input audio, at the end of the fade-out transition the output audio will be silence. Default is 44100.

`start_time, st`

Specify the start time of the fade effect. Default is 0. The value must be specified as a time duration; see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. If set this option is used instead of *start\_sample*.

`duration, d`

Specify the duration of the fade effect. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. At the end of the fade-in effect the output audio will have the same volume as the input audio, at the end of the fade-out transition the output audio will be silence. By default the duration is determined by *nb\_samples*. If set this option is used instead of *nb\_samples*.

`curve`

Set curve for fade transition.

It accepts the following values:

`tri`

select triangular, linear slope (default)

`qsin`

select quarter of sine wave

hsin

select half of sine wave

esin

select exponential sine wave

log

select logarithmic

ipar

select inverted parabola

qua

select quadratic

cub

select cubic

squ

select square root

cbr

select cubic root

par

select parabola

exp

select exponential

iqsin

select inverted quarter of sine wave

ihsin

select inverted half of sine wave

dese

```
select double-exponential seat
desi
select double-exponential sigmoid
```

### 34.5.1 Examples# TOC

- Fade in first 15 seconds of audio:

```
afade=t=in:ss=0:d=15
```

- Fade out last 25 seconds of a 900 seconds audio:

```
afade=t=out:st=875:d=25
```

### 34.6 aformat# TOC

Set output format constraints for the input audio. The framework will negotiate the most appropriate format to minimize conversions.

It accepts the following parameters:

`sample_fmts`

A '|' -separated list of requested sample formats.

`sample_rates`

A '|' -separated list of requested sample rates.

`channel_layouts`

A '|' -separated list of requested channel layouts.

See (ffmpeg-utils)the Channel Layout section in the ffmpeg-utils(1) manual for the required syntax.

If a parameter is omitted, all values are allowed.

Force the output to either unsigned 8-bit or signed 16-bit stereo

```
aformat=sample_fmts=u8|s16:channel_layouts=stereo
```

### 34.7 allpass# TOC

Apply a two-pole all-pass filter with central frequency (in Hz) *frequency*, and filter-width *width*. An all-pass filter changes the audio's frequency to phase relationship without changing its frequency to amplitude relationship.

The filter accepts the following options:

`frequency, f`

Set frequency in Hz.

`width_type`

Set method to specify band-width of filter.

`h`

Hz

`q`

Q-Factor

`o`

octave

`s`

slope

`width, w`

Specify the band-width of a filter in `width_type` units.

## 34.8 amerge# TOC

Merge two or more audio streams into a single multi-channel stream.

The filter accepts the following options:

`inputs`

Set the number of inputs. Default is 2.

If the channel layouts of the inputs are disjoint, and therefore compatible, the channel layout of the output will be set accordingly and the channels will be reordered as necessary. If the channel layouts of the inputs are not disjoint, the output will have all the channels of the first input then all the channels of the second input, in that order, and the channel layout of the output will be the default value corresponding to the total number of channels.

For example, if the first input is in 2.1 (FL+FR+LF) and the second input is FC+BL+BR, then the output will be in 5.1, with the channels in the following order: a1, a2, b1, a3, b2, b3 (a1 is the first channel of the first input, b1 is the first channel of the second input).

On the other hand, if both input are in stereo, the output channels will be in the default order: a1, a2, b1, b2, and the channel layout will be arbitrarily set to 4.0, which may or may not be the expected value.

All inputs must have the same sample rate, and format.

If inputs do not have the same duration, the output will stop with the shortest.

### 34.8.1 Examples# TOC

- Merge two mono files into a stereo stream:

```
amovie=left.wav [l] ; amovie=right.mp3 [r] ; [l] [r] amerge
```

- Multiple merges assuming 1 video stream and 6 audio streams in `input.mkv`:

```
ffmpeg -i input.mkv -filter_complex "[0:1][0:2][0:3][0:4][0:5][0:6] amerge=inputs=6" -c:a pcm_s16le output.mkv
```

## 34.9 amix# TOC

Mixes multiple audio inputs into a single output.

Note that this filter only supports float samples (the *amerge* and *pan* audio filters support many formats). If the *amix* input has integer samples then *aresample* will be automatically inserted to perform the conversion to float samples.

For example

```
ffmpeg -i INPUT1 -i INPUT2 -i INPUT3 -filter_complex amix=inputs=3:duration=first:dropout_transition=3 OUTPUT
```

will mix 3 input audio streams to a single output with the same duration as the first input and a dropout transition time of 3 seconds.

It accepts the following parameters:

`inputs`

The number of inputs. If unspecified, it defaults to 2.

`duration`

How to determine the end-of-stream.

`longest`

The duration of the longest input. (default)

`shortest`



The duration of the shortest input.

`first`

The duration of the first input.

`dropout_transition`

The transition time, in seconds, for volume renormalization when an input stream ends. The default value is 2 seconds.

## 34.10 `anull#` TOC

Pass the audio source unchanged to the output.

## 34.11 `apad#` TOC

Pad the end of an audio stream with silence.

This can be used together with `ffmpeg -shortest` to extend audio streams to the same length as the video stream.

A description of the accepted options follows.

`packet_size`

Set silence packet size. Default value is 4096.

`pad_len`

Set the number of samples of silence to add to the end. After the value is reached, the stream is terminated. This option is mutually exclusive with `whole_len`.

`whole_len`

Set the minimum total number of samples in the output audio stream. If the value is longer than the input audio length, silence is added to the end, until the value is reached. This option is mutually exclusive with `pad_len`.

If neither the `pad_len` nor the `whole_len` option is set, the filter will add silence to the end of the input stream indefinitely.

### 34.11.1 `Examples#` TOC

- Add 1024 samples of silence to the end of the input:

```
apad=pad_len=1024
```

- Make sure the audio output will contain at least 10000 samples, pad the input with silence if required:

```
apad=whole_len=10000
```

- Use `ffmpeg` to pad the audio input with silence, so that the video stream will always result the shortest and will be converted until the end in the output file when using the `shortest` option:

```
ffmpeg -i VIDEO -i AUDIO -filter_complex "[1:0]apad" -shortest OUTPUT
```

## 34.12 aphaser# TOC

Add a phasing effect to the input audio.

A phaser filter creates series of peaks and troughs in the frequency spectrum. The position of the peaks and troughs are modulated so that they vary over time, creating a sweeping effect.

A description of the accepted parameters follows.

`in_gain`

Set input gain. Default is 0.4.

`out_gain`

Set output gain. Default is 0.74

`delay`

Set delay in milliseconds. Default is 3.0.

`decay`

Set decay. Default is 0.4.

`speed`

Set modulation speed in Hz. Default is 0.5.

`type`

Set modulation type. Default is triangular.

It accepts the following values:

`'triangular, t'`

`'sinusoidal, s'`

## 34.13 aresample# TOC

Resample the input audio to the specified parameters, using the libswresample library. If none are specified then the filter will automatically convert between its input and output.

This filter is also able to stretch/squeeze the audio data to make it match the timestamps or to inject silence / cut out audio to make it match the timestamps, do a combination of both or do neither.

The filter accepts the syntax `[sample_rate:]resampler_options`, where *sample\_rate* expresses a sample rate and *resampler\_options* is a list of *key=value* pairs, separated by ":". See the ffmpeg-resampler manual for the complete list of supported options.

### 34.13.1 Examples# TOC

- Resample the input audio to 44100Hz:

```
aresample=44100
```

- Stretch/squeeze samples to the given timestamps, with a maximum of 1000 samples per second compensation:

```
aresample=async=1000
```

## 34.14 asetnsamples# TOC

Set the number of samples per each output audio frame.

The last output packet may contain a different number of samples, as the filter will flush all the remaining samples when the input audio signal its end.

The filter accepts the following options:

`nb_out_samples, n`

Set the number of frames per each output audio frame. The number is intended as the number of samples *per each channel*. Default value is 1024.

`pad, p`

If set to 1, the filter will pad the last audio frame with zeroes, so that the last frame will contain the same number of samples as the previous ones. Default value is 1.

For example, to set the number of per-frame samples to 1234 and disable padding for the last frame, use:

```
asetnsamples=n=1234:p=0
```

## 34.15 asetrate# TOC

Set the sample rate without altering the PCM data. This will result in a change of speed and pitch.

The filter accepts the following options:

`sample_rate, r`

Set the output sample rate. Default is 44100 Hz.

## 34.16 ashowinfo# TOC

Show a line containing various information for each input audio frame. The input audio is not modified.

The shown line contains a sequence of key/value pairs of the form *key:value*.

The following values are shown in the output:

`n`

The (sequential) number of the input frame, starting from 0.

`pts`

The presentation timestamp of the input frame, in time base units; the time base depends on the filter input pad, and is usually  $1/\text{sample\_rate}$ .

`pts_time`

The presentation timestamp of the input frame in seconds.

`pos`

position of the frame in the input stream, -1 if this information is unavailable and/or meaningless (for example in case of synthetic audio)

`fmt`

The sample format.

`chlayout`

The channel layout.

`rate`

The sample rate for the audio frame.

`nb_samples`

The number of samples (per channel) in the frame.

`checksum`

The Adler-32 checksum (printed in hexadecimal) of the audio data. For planar audio, the data is treated as if all the planes were concatenated.

`plane_checksums`

A list of Adler-32 checksums for each data plane.

## 34.17 `astats# TOC`

Display time domain statistical information about the audio channels. Statistics are calculated and displayed for each audio channel and, where applicable, an overall figure is also given.

It accepts the following option:

`length`

Short window length in seconds, used for peak and trough RMS measurement. Default is 0.05 (50 milliseconds). Allowed range is [0.1 - 10].

`metadata`

Set metadata injection. All the metadata keys are prefixed with `lavfi.astats.X`, where X is channel number starting from 1 or string `Overall`. Default is disabled.

Available keys for each channel are: `DC_offset` `Min_level` `Max_level` `Min_difference` `Max_difference` `Mean_difference` `Peak_level` `RMS_peak` `RMS_trough` `Crest_factor` `Flat_factor` `Peak_count` `Bit_depth`

and for Overall: `DC_offset` `Min_level` `Max_level` `Min_difference` `Max_difference` `Mean_difference` `Peak_level` `RMS_level` `RMS_peak` `RMS_trough` `Flat_factor` `Peak_count` `Bit_depth` `Number_of_samples`

For example full key look like this `lavfi.astats.1.DC_offset` or this `lavfi.astats.Overall.Peak_count`.

For description what each key means read below.

`reset`

Set number of frame after which stats are going to be recalculated. Default is disabled.

A description of each shown parameter follows:

DC offset

Mean amplitude displacement from zero.

Min level

Minimal sample level.

Max level

Maximal sample level.

Min difference

Minimal difference between two consecutive samples.

Max difference

Maximal difference between two consecutive samples.

Mean difference

Mean difference between two consecutive samples. The average of each difference between two consecutive samples.

Peak level dB

RMS level dB

Standard peak and RMS level measured in dBFS.

RMS peak dB

RMS trough dB

Peak and trough values for RMS level measured over a short window.

Crest factor

Standard ratio of peak to RMS level (note: not in dB).

Flat factor

Flatness (i.e. consecutive samples with the same value) of the signal at its peak levels (i.e. either *Min level* or *Max level*).

Peak count

Number of occasions (not the number of samples) that the signal attained either *Min level* or *Max level*.

Bit depth

Overall bit depth of audio. Number of bits used for each sample.

## 34.18 astreamsync# TOC

Forward two audio streams and control the order the buffers are forwarded.

The filter accepts the following options:

`expr, e`

Set the expression deciding which stream should be forwarded next: if the result is negative, the first stream is forwarded; if the result is positive or zero, the second stream is forwarded. It can use the following variables:

*b1 b2*

number of buffers forwarded so far on each stream

*s1 s2*

number of samples forwarded so far on each stream

*t1 t2*

current timestamp of each stream

The default value is  $t1 - t2$ , which means to always forward the stream that has a smaller timestamp.

### 34.18.1 Examples# TOC

Stress-test `amerge` by randomly sending buffers on the wrong input, while avoiding too much of a desynchronization:

```
amovie=file.ogg [a] ; amovie=file.mp3 [b] ;  
[a] [b] astreamsync=(2*random(1))-1+tanh(5*(t1-t2)) [a2] [b2] ;  
[a2] [b2] amerge
```

## 34.19 asyncts# TOC

Synchronize audio data with timestamps by squeezing/stretching it and/or dropping samples/adding silence when needed.

This filter is not built by default, please use aresample to do squeezing/stretching.

It accepts the following parameters:

`compensate`

Enable stretching/squeezing the data to make it match the timestamps. Disabled by default. When disabled, time gaps are covered with silence.

`min_delta`

The minimum difference between timestamps and audio data (in seconds) to trigger adding/dropping samples. The default value is 0.1. If you get an imperfect sync with this filter, try setting this parameter to 0.

`max_comp`

The maximum compensation in samples per second. Only relevant with `compensate=1`. The default value is 500.

`first_pts`

Assume that the first PTS should be this value. The time base is 1 / sample rate. This allows for padding/trimming at the start of the stream. By default, no assumption is made about the first frame's expected PTS, so no padding or trimming is done. For example, this could be set to 0 to pad the beginning with silence if an audio stream starts after the video stream or to trim any samples with a negative PTS due to encoder delay.

## 34.20 atempo# TOC

Adjust audio tempo.

The filter accepts exactly one parameter, the audio tempo. If not specified then the filter will assume nominal 1.0 tempo. Tempo must be in the [0.5, 2.0] range.

### 34.20.1 Examples# TOC

- Slow down audio to 80% tempo:

`atempo=0.8`

- To speed up audio to 125% tempo:

`atempo=1.25`



## 34.21 atrim# TOC

Trim the input so that the output contains one continuous subpart of the input.

It accepts the following parameters:

`start`

Timestamp (in seconds) of the start of the section to keep. I.e. the audio sample with the timestamp *start* will be the first sample in the output.

`end`

Specify time of the first audio sample that will be dropped, i.e. the audio sample immediately preceding the one with the timestamp *end* will be the last sample in the output.

`start_pts`

Same as *start*, except this option sets the start timestamp in samples instead of seconds.

`end_pts`

Same as *end*, except this option sets the end timestamp in samples instead of seconds.

`duration`

The maximum duration of the output in seconds.

`start_sample`

The number of the first sample that should be output.

`end_sample`

The number of the first sample that should be dropped.

`start`, `end`, and `duration` are expressed as time duration specifications; see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

Note that the first two sets of the `start/end` options and the `duration` option look at the frame timestamp, while the `_sample` options simply count the samples that pass through the filter. So `start/end_pts` and `start/end_sample` will give different results when the timestamps are wrong, inexact or do not start at zero. Also note that this filter does not modify the timestamps. If you wish to have the output timestamps start at zero, insert the `asetpts` filter after the `atrim` filter.

If multiple `start` or `end` options are set, this filter tries to be greedy and keep all samples that match at least one of the specified constraints. To keep only the part that matches all the constraints at once, chain multiple `atrim` filters.

The defaults are such that all the input is kept. So it is possible to set e.g. just the end values to keep everything before the specified time.

Examples:

- Drop everything except the second minute of input:

```
ffmpeg -i INPUT -af atrim=60:120
```

- Keep only the first 1000 samples:

```
ffmpeg -i INPUT -af atrim=end_sample=1000
```

## 34.22 bandpass# TOC

Apply a two-pole Butterworth band-pass filter with central frequency *frequency*, and (3dB-point) band-width *width*. The *csg* option selects a constant skirt gain (peak gain = Q) instead of the default: constant 0dB peak gain. The filter roll off at 6dB per octave (20dB per decade).

The filter accepts the following options:

*frequency*, *f*

Set the filter's central frequency. Default is 3000.

*csg*

Constant skirt gain if set to 1. Defaults to 0.

*width\_type*

Set method to specify band-width of filter.

*h*

Hz

*q*

Q-Factor

*o*

octave

*s*

slope

`width, w`

Specify the band-width of a filter in `width_type` units.

### 34.23 bandreject# TOC

Apply a two-pole Butterworth band-reject filter with central frequency *frequency*, and (3dB-point) band-width *width*. The filter roll off at 6dB per octave (20dB per decade).

The filter accepts the following options:

`frequency, f`

Set the filter's central frequency. Default is 3000.

`width_type`

Set method to specify band-width of filter.

`h`

Hz

`q`

Q-Factor

`o`

octave

`s`

slope

`width, w`

Specify the band-width of a filter in `width_type` units.

### 34.24 bass# TOC

Boost or cut the bass (lower) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

The filter accepts the following options:

`gain, g`

Give the gain at 0 Hz. Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of clipping when using a positive gain.

frequency, *f*

Set the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 100 Hz.

width\_type

Set method to specify band-width of filter.

h

Hz

q

Q-Factor

o

octave

s

slope

width, *w*

Determine how steep is the filter's shelf transition.

## 34.25 biquad# TOC

Apply a biquad IIR filter with the given coefficients. Where *b0*, *b1*, *b2* and *a0*, *a1*, *a2* are the numerator and denominator coefficients respectively.

## 34.26 bs2b# TOC

Bauer stereo to binaural transformation, which improves headphone listening of stereo audio records.

It accepts the following parameters:

profile

Pre-defined crossfeed level.

default

Default level (fcut=700, feed=50).

cmoy

Chu Moy circuit (fcut=700, feed=60).

jmeier

Jan Meier circuit (fcut=650, feed=95).

fcut

Cut frequency (in Hz).

feed

Feed level (in Hz).

## 34.27 channelmap# TOC

Remap input channels to new locations.

It accepts the following parameters:

channel\_layout

The channel layout of the output stream.

map

Map channels from input to output. The argument is a '|'-separated list of mappings, each in the *in\_channel-out\_channel* or *in\_channel* form. *in\_channel* can be either the name of the input channel (e.g. FL for front left) or its index in the input channel layout. *out\_channel* is the name of the output channel or its index in the output channel layout. If *out\_channel* is not given then it is implicitly an index, starting with zero and increasing by one for each mapping.

If no mapping is present, the filter will implicitly map input channels to output channels, preserving indices.

For example, assuming a 5.1+downmix input MOV file,

```
ffmpeg -i in.mov -filter 'channelmap=map=DL-FL|DR-FR' out.wav
```

will create an output WAV file tagged as stereo from the downmix channels of the input.

To fix a 5.1 WAV improperly encoded in AAC's native channel order

```
ffmpeg -i in.wav -filter 'channelmap=1|2|0|5|3|4:5.1' out.wav
```

## 34.28 channelsplit# TOC

Split each channel from an input audio stream into a separate output stream.

It accepts the following parameters:

`channel_layout`

The channel layout of the input stream. The default is "stereo".

For example, assuming a stereo input MP3 file,

```
ffmpeg -i in.mp3 -filter_complex channelsplit out.mkv
```

will create an output Matroska file with two audio streams, one containing only the left channel and the other the right channel.

Split a 5.1 WAV file into per-channel files:

```
ffmpeg -i in.wav -filter_complex  
'channelsplit=channel_layout=5.1[FL][FR][FC][LFE][SL][SR]'  
-map '[FL]' front_left.wav -map '[FR]' front_right.wav -map '[FC]'  
front_center.wav -map '[LFE]' lfe.wav -map '[SL]' side_left.wav -map '[SR]'  
side_right.wav
```

## 34.29 chorus# TOC

Add a chorus effect to the audio.

Can make a single vocal sound like a chorus, but can also be applied to instrumentation.

Chorus resembles an echo effect with a short delay, but whereas with echo the delay is constant, with chorus, it is varied using sinusoidal or triangular modulation. The modulation depth defines the range the modulated delay is played before or after the delay. Hence the delayed sound will sound slower or faster, that is the delayed sound tuned around the original one, like in a chorus where some vocals are slightly off key.

It accepts the following parameters:

`in_gain`

Set input gain. Default is 0.4.

`out_gain`

Set output gain. Default is 0.4.

delays

Set delays. A typical delay is around 40ms to 60ms.

decays

Set decays.

speeds

Set speeds.

depths

Set depths.

### 34.29.1 Examples# TOC

- A single delay:

```
chorus=0.7:0.9:55:0.4:0.25:2
```

- Two delays:

```
chorus=0.6:0.9:50|60:0.4|0.32:0.25|0.4:2|1.3
```

- Fuller sounding chorus with three delays:

```
chorus=0.5:0.9:50|60|40:0.4|0.32|0.3:0.25|0.4|0.3:2|2.3|1.3
```

### 34.30 compand# TOC

Compress or expand the audio's dynamic range.

It accepts the following parameters:

attacks

decays

A list of times in seconds for each channel over which the instantaneous level of the input signal is averaged to determine its volume. *attacks* refers to increase of volume and *decays* refers to decrease of volume. For most situations, the attack time (response to the audio getting louder) should be shorter than the decay time, because the human ear is more sensitive to sudden loud audio than sudden soft audio. A typical value for attack is 0.3 seconds and a typical value for decay is 0.8 seconds. If specified number of attacks & decays is lower than number of channels, the last set attack/decay will be used for all remaining channels.

#### points

A list of points for the transfer function, specified in dB relative to the maximum possible signal amplitude. Each key points list must be defined using the following syntax:

`x0/y0 | x1/y1 | x2/y2 | . . . .` or `x0/y0 x1/y1 x2/y2 . . . .`

The input values must be in strictly increasing order but the transfer function does not have to be monotonically rising. The point 0/0 is assumed but may be overridden (by 0/out-dBn). Typical values for the transfer function are -70/-70 | -60/-20.

#### soft-knee

Set the curve radius in dB for all joints. It defaults to 0.01.

#### gain

Set the additional gain in dB to be applied at all points on the transfer function. This allows for easy adjustment of the overall gain. It defaults to 0.

#### volume

Set an initial volume, in dB, to be assumed for each channel when filtering starts. This permits the user to supply a nominal level initially, so that, for example, a very large gain is not applied to initial signal levels before the companding has begun to operate. A typical value for audio which is initially quiet is -90 dB. It defaults to 0.

#### delay

Set a delay, in seconds. The input audio is analyzed immediately, but audio is delayed before being fed to the volume adjuster. Specifying a delay approximately equal to the attack/decay times allows the filter to effectively operate in predictive rather than reactive mode. It defaults to 0.

### 34.30.1 Examples# TOC

- Make music with both quiet and loud passages suitable for listening to in a noisy environment:

```
compand=.3|.3:1|1:-90/-60|-60/-40|-40/-30|-20/-20:6:0:-90:0.2
```

Another example for audio with whisper and explosion parts:

```
compand=0|0:1|1:-90/-900|-70/-70|-30/-9|0/-3:6:0:0:0
```

- A noise gate for when the noise is at a lower level than the signal:

```
compand=.1|.1:.2|.2:-900/-900|-50.1/-900|-50/-50:.01:0:-90:.1
```

- Here is another noise gate, this time for when the noise is at a higher level than the signal (making it, in some ways, similar to squelch):



```
compand=.1|.1:.1|.1:-45.1/-45.1|-45/-900|0/-900:.01:45:-90:.1
```

### 34.31 dcshift# TOC

Apply a DC shift to the audio.

This can be useful to remove a DC offset (caused perhaps by a hardware problem in the recording chain) from the audio. The effect of a DC offset is reduced headroom and hence volume. The `astats` filter can be used to determine if a signal has a DC offset.

`shift`

Set the DC shift, allowed range is [-1, 1]. It indicates the amount to shift the audio.

`limitergain`

Optional. It should have a value much less than 1 (e.g. 0.05 or 0.02) and is used to prevent clipping.

### 34.32 dynaudnorm# TOC

Dynamic Audio Normalizer.

This filter applies a certain amount of gain to the input audio in order to bring its peak magnitude to a target level (e.g. 0 dBFS). However, in contrast to more "simple" normalization algorithms, the Dynamic Audio Normalizer *dynamically* re-adjusts the gain factor to the input audio. This allows for applying extra gain to the "quiet" sections of the audio while avoiding distortions or clipping the "loud" sections. In other words: The Dynamic Audio Normalizer will "even out" the volume of quiet and loud sections, in the sense that the volume of each section is brought to the same target level. Note, however, that the Dynamic Audio Normalizer achieves this goal *without* applying "dynamic range compressing". It will retain 100% of the dynamic range *within* each section of the audio file.

`f`

Set the frame length in milliseconds. In range from 10 to 8000 milliseconds. Default is 500 milliseconds. The Dynamic Audio Normalizer processes the input audio in small chunks, referred to as frames. This is required, because a peak magnitude has no meaning for just a single sample value. Instead, we need to determine the peak magnitude for a contiguous sequence of sample values. While a "standard" normalizer would simply use the peak magnitude of the complete file, the Dynamic Audio Normalizer determines the peak magnitude individually for each frame. The length of a frame is specified in milliseconds. By default, the Dynamic Audio Normalizer uses a frame length of 500 milliseconds, which has been found to give good results with most files. Note that the exact frame length, in number of samples, will be determined automatically, based on the sampling rate of the individual input audio file.

`g`

Set the Gaussian filter window size. In range from 3 to 301, must be odd number. Default is 31. Probably the most important parameter of the Dynamic Audio Normalizer is the `window_size` of the Gaussian smoothing filter. The filter's window size is specified in frames, centered around the current frame. For the sake of simplicity, this must be an odd number. Consequently, the default value of 31 takes into account the current frame, as well as the 15 preceding frames and the 15 subsequent frames. Using a larger window results in a stronger smoothing effect and thus in less gain variation, i.e. slower gain adaptation. Conversely, using a smaller window results in a weaker smoothing effect and thus in more gain variation, i.e. faster gain adaptation. In other words, the more you increase this value, the more the Dynamic Audio Normalizer will behave like a "traditional" normalization filter. On the contrary, the more you decrease this value, the more the Dynamic Audio Normalizer will behave like a dynamic range compressor.

p

Set the target peak value. This specifies the highest permissible magnitude level for the normalized audio input. This filter will try to approach the target peak magnitude as closely as possible, but at the same time it also makes sure that the normalized signal will never exceed the peak magnitude. A frame's maximum local gain factor is imposed directly by the target peak magnitude. The default value is 0.95 and thus leaves a headroom of 5%\*. It is not recommended to go above this value.

m

Set the maximum gain factor. In range from 1.0 to 100.0. Default is 10.0. The Dynamic Audio Normalizer determines the maximum possible (local) gain factor for each input frame, i.e. the maximum gain factor that does not result in clipping or distortion. The maximum gain factor is determined by the frame's highest magnitude sample. However, the Dynamic Audio Normalizer additionally bounds the frame's maximum gain factor by a predetermined (global) maximum gain factor. This is done in order to avoid excessive gain factors in "silent" or almost silent frames. By default, the maximum gain factor is 10.0, For most inputs the default value should be sufficient and it usually is not recommended to increase this value. Though, for input with an extremely low overall volume level, it may be necessary to allow even higher gain factors. Note, however, that the Dynamic Audio Normalizer does not simply apply a "hard" threshold (i.e. cut off values above the threshold). Instead, a "sigmoid" threshold function will be applied. This way, the gain factors will smoothly approach the threshold value, but never exceed that value.

r

Set the target RMS. In range from 0.0 to 1.0. Default is 0.0 - disabled. By default, the Dynamic Audio Normalizer performs "peak" normalization. This means that the maximum local gain factor for each frame is defined (only) by the frame's highest magnitude sample. This way, the samples can be amplified as much as possible without exceeding the maximum signal level, i.e. without clipping. Optionally, however, the Dynamic Audio Normalizer can also take into account the frame's root mean square, abbreviated RMS. In electrical engineering, the RMS is commonly used to determine the power of a time-varying signal. It is therefore considered that the RMS is a better approximation of the "perceived loudness" than just looking at the signal's peak magnitude. Consequently, by adjusting all frames to a constant RMS value, a uniform "perceived loudness" can be established. If a target RMS value has been specified, a frame's local gain factor is defined as the factor that would result in exactly that RMS value. Note, however, that the maximum local gain factor is still restricted

by the frame's highest magnitude sample, in order to prevent clipping.

n

Enable channels coupling. By default is enabled. By default, the Dynamic Audio Normalizer will amplify all channels by the same amount. This means the same gain factor will be applied to all channels, i.e. the maximum possible gain factor is determined by the "loudest" channel. However, in some recordings, it may happen that the volume of the different channels is uneven, e.g. one channel may be "quieter" than the other one(s). In this case, this option can be used to disable the channel coupling. This way, the gain factor will be determined independently for each channel, depending only on the individual channel's highest magnitude sample. This allows for harmonizing the volume of the different channels.

c

Enable DC bias correction. By default is disabled. An audio signal (in the time domain) is a sequence of sample values. In the Dynamic Audio Normalizer these sample values are represented in the -1.0 to 1.0 range, regardless of the original input format. Normally, the audio signal, or "waveform", should be centered around the zero point. That means if we calculate the mean value of all samples in a file, or in a single frame, then the result should be 0.0 or at least very close to that value. If, however, there is a significant deviation of the mean value from 0.0, in either positive or negative direction, this is referred to as a DC bias or DC offset. Since a DC bias is clearly undesirable, the Dynamic Audio Normalizer provides optional DC bias correction. With DC bias correction enabled, the Dynamic Audio Normalizer will determine the mean value, or "DC correction" offset, of each input frame and subtract that value from all of the frame's sample values which ensures those samples are centered around 0.0 again. Also, in order to avoid "gaps" at the frame boundaries, the DC correction offset values will be interpolated smoothly between neighbouring frames.

b

Enable alternative boundary mode. By default is disabled. The Dynamic Audio Normalizer takes into account a certain neighbourhood around each frame. This includes the preceding frames as well as the subsequent frames. However, for the "boundary" frames, located at the very beginning and at the very end of the audio file, not all neighbouring frames are available. In particular, for the first few frames in the audio file, the preceding frames are not known. And, similarly, for the last few frames in the audio file, the subsequent frames are not known. Thus, the question arises which gain factors should be assumed for the missing frames in the "boundary" region. The Dynamic Audio Normalizer implements two modes to deal with this situation. The default boundary mode assumes a gain factor of exactly 1.0 for the missing frames, resulting in a smooth "fade in" and "fade out" at the beginning and at the end of the input, respectively.

s

Set the compress factor. In range from 0.0 to 30.0. Default is 0.0. By default, the Dynamic Audio Normalizer does not apply "traditional" compression. This means that signal peaks will not be pruned and thus the full dynamic range will be retained within each local neighbourhood. However, in some cases it may be desirable to combine the Dynamic Audio Normalizer's normalization algorithm with a more "traditional" compression. For this purpose, the Dynamic Audio Normalizer provides an

optional compression (thresholding) function. If (and only if) the compression feature is enabled, all input frames will be processed by a soft knee thresholding function prior to the actual normalization process. Put simply, the thresholding function is going to prune all samples whose magnitude exceeds a certain threshold value. However, the Dynamic Audio Normalizer does not simply apply a fixed threshold value. Instead, the threshold value will be adjusted for each individual frame. In general, smaller parameters result in stronger compression, and vice versa. Values below 3.0 are not recommended, because audible distortion may appear.

### 34.33 earwax# TOC

Make audio easier to listen to on headphones.

This filter adds ‘cues’ to 44.1kHz stereo (i.e. audio CD format) audio so that when listened to on headphones the stereo image is moved from inside your head (standard for headphones) to outside and in front of the listener (standard for speakers).

Ported from SoX.

### 34.34 equalizer# TOC

Apply a two-pole peaking equalisation (EQ) filter. With this filter, the signal-level at and around a selected frequency can be increased or decreased, whilst (unlike bandpass and bandreject filters) that at all other frequencies is unchanged.

In order to produce complex equalisation curves, this filter can be given several times, each with a different central frequency.

The filter accepts the following options:

`frequency, f`

Set the filter’s central frequency in Hz.

`width_type`

Set method to specify band-width of filter.

`h`

Hz

`q`

Q-Factor

`o`

octave

s

slope

width, w

Specify the band-width of a filter in width\_type units.

gain, g

Set the required gain or attenuation in dB. Beware of clipping when using a positive gain.

### 34.34.1 Examples# TOC

- Attenuate 10 dB at 1000 Hz, with a bandwidth of 200 Hz:

```
equalizer=f=1000:width_type=h:width=200:g=-10
```

- Apply 2 dB gain at 1000 Hz with Q 1 and attenuate 5 dB at 100 Hz with Q 2:

```
equalizer=f=1000:width_type=q:width=1:g=2,equalizer=f=100:width_type=q:width=2:g=-5
```

### 34.35 flanger# TOC

Apply a flanging effect to the audio.

The filter accepts the following options:

delay

Set base delay in milliseconds. Range from 0 to 30. Default value is 0.

depth

Set added swep delay in milliseconds. Range from 0 to 10. Default value is 2.

regen

Set percentage regeneration (delayed signal feedback). Range from -95 to 95. Default value is 0.

width

Set percentage of delayed signal mixed with original. Range from 0 to 100. Default value is 71.

speed

Set sweeps per second (Hz). Range from 0.1 to 10. Default value is 0.5.

shape

Set swept wave shape, can be *triangular* or *sinusoidal*. Default value is *sinusoidal*.

phase

Set swept wave percentage-shift for multi channel. Range from 0 to 100. Default value is 25.

interp

Set delay-line interpolation, *linear* or *quadratic*. Default is *linear*.

### 34.36 highpass# TOC

Apply a high-pass filter with 3dB point frequency. The filter can be either single-pole, or double-pole (the default). The filter roll off at 6dB per pole per octave (20dB per pole per decade).

The filter accepts the following options:

frequency, f

Set frequency in Hz. Default is 3000.

poles, p

Set number of poles. Default is 2.

width\_type

Set method to specify band-width of filter.

h

Hz

q

Q-Factor

o

octave

s

slope

width, w

Specify the band-width of a filter in width\_type units. Applies only to double-pole filter. The default is 0.707q and gives a Butterworth response.

## 34.37 join# TOC

Join multiple input streams into one multi-channel stream.

It accepts the following parameters:

inputs

The number of input streams. It defaults to 2.

channel\_layout

The desired output channel layout. It defaults to stereo.

map

Map channels from inputs to output. The argument is a '|'-separated list of mappings, each in the *input\_idx.in\_channel-out\_channel* form. *input\_idx* is the 0-based index of the input stream. *in\_channel* can be either the name of the input channel (e.g. FL for front left) or its index in the specified input stream. *out\_channel* is the name of the output channel.

The filter will attempt to guess the mappings when they are not specified explicitly. It does so by first trying to find an unused matching input channel and if that fails it picks the first unused input channel.

Join 3 inputs (with properly set channel layouts):

```
ffmpeg -i INPUT1 -i INPUT2 -i INPUT3 -filter_complex join=inputs=3 OUTPUT
```

Build a 5.1 output from 6 single-channel streams:

```
ffmpeg -i fl -i fr -i fc -i sl -i sr -i lfe -filter_complex  
'join=inputs=6:channel_layout=5.1:map=0.0-FL|1.0-FR|2.0-FC|3.0-SL|4.0-SR|5.0-LFE'  
out
```

## 34.38 ladspa# TOC

Load a LADSPA (Linux Audio Developer's Simple Plugin API) plugin.

To enable compilation of this filter you need to configure FFmpeg with `--enable-ladspa`.

file, f

Specifies the name of LADSPA plugin library to load. If the environment variable LADSPA\_PATH is defined, the LADSPA plugin is searched in each one of the directories specified by the colon separated list in LADSPA\_PATH, otherwise in the standard LADSPA paths, which are in this order:  
HOME/.ladspa/lib/, /usr/local/lib/ladspa/, /usr/lib/ladspa/.

plugin, p

Specifies the plugin within the library. Some libraries contain only one plugin, but others contain many of them. If this is not set filter will list all available plugins within the specified library.

controls, c

Set the '|' separated list of controls which are zero or more floating point values that determine the behavior of the loaded plugin (for example delay, threshold or gain). Controls need to be defined using the following syntax: c0=value0|c1=value1|c2=value2|..., where *valuei* is the value set on the *i*-th control. If controls is set to help, all available controls and their valid ranges are printed.

sample\_rate, s

Specify the sample rate, default to 44100. Only used if plugin have zero inputs.

nb\_samples, n

Set the number of samples per channel per each output frame, default is 1024. Only used if plugin have zero inputs.

duration, d

Set the minimum duration of the sourced audio. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. Note that the resulting duration may be greater than the specified duration, as the generated audio is always cut at the end of a complete frame. If not specified, or the expressed duration is negative, the audio is supposed to be generated forever. Only used if plugin have zero inputs.

### 34.38.1 Examples# TOC

- List all available plugins within amp (LADSPA example plugin) library:

```
ladspa=file=amp
```

- List all available controls and their valid ranges for vcf\_notch plugin from VCF library:

```
ladspa=f=vcf:p=vcf_notch:c=help
```

- Simulate low quality audio equipment using Computer Music Toolkit (CMT) plugin library:

```
ladspa=file=cmt:plugin=lofi:controls=c0=22|c1=12|c2=12
```

- Add reverberation to the audio using TAP-plugins (Tom's Audio Processing plugins):



```
ladspa=file=tap_reverb:tap_reverb
```

- Generate white noise, with 0.2 amplitude:

```
ladspa=file=cmt:noise_source_white:c=c0=.2
```

- Generate 20 bpm clicks using plugin C\* Click - Metronome from the C\* Audio Plugin Suite (CAPS) library:

```
ladspa=file=caps:Click:c=c1=20'
```

- Apply C\* Eq10X2 - Stereo 10-band equaliser effect:

```
ladspa=caps:Eq10X2:c=c0=-48|c9=-24|c3=12|c4=2
```

### 34.38.2 Commands# TOC

This filter supports the following commands:

`cN`

Modify the  $N$ -th control value.

If the specified value is not valid, it is ignored and prior one is kept.

### 34.39 lowpass# TOC

Apply a low-pass filter with 3dB point frequency. The filter can be either single-pole or double-pole (the default). The filter roll off at 6dB per pole per octave (20dB per pole per decade).

The filter accepts the following options:

`frequency, f`

Set frequency in Hz. Default is 500.

`poles, p`

Set number of poles. Default is 2.

`width_type`

Set method to specify band-width of filter.

`h`

Hz

`q`

Q-Factor

o

octave

s

slope

width, w

Specify the band-width of a filter in width\_type units. Applies only to double-pole filter. The default is 0.707q and gives a Butterworth response.

## 34.40 pan# TOC

Mix channels with specific gain levels. The filter accepts the output channel layout followed by a set of channels definitions.

This filter is also designed to efficiently remap the channels of an audio stream.

The filter accepts parameters of the form: "*l*|*outdef*|*outdef*|..."

*l*

output channel layout or number of channels

*outdef*

output channel specification, of the form: "*out\_name*=[*gain*\*]*in\_name*[+[*gain*\*]*in\_name*...]"

*out\_name*

output channel to define, either a channel name (FL, FR, etc.) or a channel number (c0, c1, etc.)

*gain*

multiplicative coefficient for the channel, 1 leaving the volume unchanged

*in\_name*

input channel to use, see *out\_name* for details; it is not possible to mix named and numbered input channels

If the '=' in a channel specification is replaced by '<', then the gains for that specification will be renormalized so that the total is 1, thus avoiding clipping noise.

### 34.40.1 Mixing examples# TOC

For example, if you want to down-mix from stereo to mono, but with a bigger factor for the left channel:

```
pan=lc|c0=0.9*c0+0.1*c1
```

A customized down-mix to stereo that works automatically for 3-, 4-, 5- and 7-channels surround:

```
pan=stereo| FL < FL + 0.5*FC + 0.6*BL + 0.6*SL | FR < FR + 0.5*FC + 0.6*BR + 0.6*SR
```

Note that `ffmpeg` integrates a default down-mix (and up-mix) system that should be preferred (see `"-ac"` option) unless you have very specific needs.

### 34.40.2 Remapping examples# TOC

The channel remapping will be effective if, and only if:

- gain coefficients are zeroes or ones,
- only one input per channel output,

If all these conditions are satisfied, the filter will notify the user ("Pure channel mapping detected"), and use an optimized and lossless method to do the remapping.

For example, if you have a 5.1 source and want a stereo audio stream by dropping the extra channels:

```
pan="stereo| c0=FL | c1=FR"
```

Given the same source, you can also switch front left and front right channels and keep the input channel layout:

```
pan="5.1| c0=c1 | c1=c0 | c2=c2 | c3=c3 | c4=c4 | c5=c5"
```

If the input is a stereo audio stream, you can mute the front left channel (and still keep the stereo channel layout) with:

```
pan="stereo|c1=c1"
```

Still with a stereo audio stream input, you can copy the right channel in both front left and right:

```
pan="stereo| c0=FR | c1=FR"
```

## 34.41 replaygain# TOC

ReplayGain scanner filter. This filter takes an audio stream as an input and outputs it unchanged. At end of filtering it displays `track_gain` and `track_peak`.

## 34.42 resample# TOC

Convert the audio sample format, sample rate and channel layout. It is not meant to be used directly.

## 34.43 sidechaincompress# TOC

This filter acts like normal compressor but has the ability to compress detected signal using second input signal. It needs two input streams and returns one output stream. First input stream will be processed depending on second stream signal. The filtered signal then can be filtered with other filters in later stages of processing. See pan and amerge filter.

The filter accepts the following options:

threshold

If a signal of second stream raises above this level it will affect the gain reduction of first stream. By default is 0.125. Range is between 0.00097563 and 1.

ratio

Set a ratio about which the signal is reduced. 1:2 means that if the level raised 4dB above the threshold, it will be only 2dB above after the reduction. Default is 2. Range is between 1 and 20.

attack

Amount of milliseconds the signal has to rise above the threshold before gain reduction starts. Default is 20. Range is between 0.01 and 2000.

release

Amount of milliseconds the signal has to fall below the threshold before reduction is decreased again. Default is 250. Range is between 0.01 and 9000.

makeup

Set the amount by how much signal will be amplified after processing. Default is 2. Range is from 1 and 64.

knee

Curve the sharp knee around the threshold to enter gain reduction more softly. Default is 2.82843. Range is between 1 and 8.

link

Choose if the average level between all channels of side-chain stream or the louder(maximum) channel of side-chain stream affects the reduction. Default is average.

detection

Should the exact signal be taken in case of `peak` or an RMS one in case of `rms`. Default is `rms` which is mainly smoother.

### 34.43.1 Examples# TOC

- Full `ffmpeg` example taking 2 audio inputs, 1st input to be compressed depending on the signal of 2nd input and later compressed signal to be merged with 2nd input:

```
ffmpeg -i main.flac -i sidechain.flac -filter_complex "[1:a]asplit=2[sc][mix];[0:a][sc]sidechaincompress[compr];[compr][mix]amerge"
```

### 34.44 silencedetect# TOC

Detect silence in an audio stream.

This filter logs a message when it detects that the input audio volume is less or equal to a noise tolerance value for a duration greater or equal to the minimum detected noise duration.

The printed times and duration are expressed in seconds.

The filter accepts the following options:

`duration, d`

Set silence duration until notification (default is 2 seconds).

`noise, n`

Set noise tolerance. Can be specified in dB (in case "dB" is appended to the specified value) or amplitude ratio. Default is -60dB, or 0.001.

#### 34.44.1 Examples# TOC

- Detect 5 seconds of silence with -50dB noise tolerance:

```
silencedetect=n=-50dB:d=5
```

- Complete example with `ffmpeg` to detect silence with 0.0001 noise tolerance in `silence.mp3`:

```
ffmpeg -i silence.mp3 -af silencedetect=noise=0.0001 -f null -
```

### 34.45 silenceremove# TOC

Remove silence from the beginning, middle or end of the audio.

The filter accepts the following options:

`start_periods`

This value is used to indicate if audio should be trimmed at beginning of the audio. A value of zero indicates no silence should be trimmed from the beginning. When specifying a non-zero value, it trims audio up until it finds non-silence. Normally, when trimming silence from beginning of audio the *start\_periods* will be 1 but it can be increased to higher values to trim all audio up to specific count of non-silence periods. Default value is 0.

`start_duration`

Specify the amount of time that non-silence must be detected before it stops trimming audio. By increasing the duration, bursts of noises can be treated as silence and trimmed off. Default value is 0.

`start_threshold`

This indicates what sample value should be treated as silence. For digital audio, a value of 0 may be fine but for audio recorded from analog, you may wish to increase the value to account for background noise. Can be specified in dB (in case "dB" is appended to the specified value) or amplitude ratio. Default value is 0.

`stop_periods`

Set the count for trimming silence from the end of audio. To remove silence from the middle of a file, specify a *stop\_periods* that is negative. This value is then treated as a positive value and is used to indicate the effect should restart processing as specified by *start\_periods*, making it suitable for removing periods of silence in the middle of the audio. Default value is 0.

`stop_duration`

Specify a duration of silence that must exist before audio is not copied any more. By specifying a higher duration, silence that is wanted can be left in the audio. Default value is 0.

`stop_threshold`

This is the same as *start\_threshold* but for trimming silence from the end of audio. Can be specified in dB (in case "dB" is appended to the specified value) or amplitude ratio. Default value is 0.

`leave_silence`

This indicate that *stop\_duration* length of audio should be left intact at the beginning of each period of silence. For example, if you want to remove long pauses between words but do not want to remove the pauses completely. Default value is 0.

### 34.45.1 Examples# TOC

- The following example shows how this filter can be used to start a recording that does not contain the delay at the start which usually occurs between pressing the record button and the start of the performance:

```
silenceremove=1:5:0.02
```

### 34.46 treble# TOC

Boost or cut treble (upper) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

The filter accepts the following options:

gain, g

Give the gain at whichever is the lower of ~22 kHz and the Nyquist frequency. Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of clipping when using a positive gain.

frequency, f

Set the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 3000 Hz.

width\_type

Set method to specify band-width of filter.

h

Hz

q

Q-Factor

o

octave

s

slope

width, w

Determine how steep is the filter's shelf transition.

## 34.47 volume# TOC

Adjust the input audio volume.

It accepts the following parameters:

`volume`

Set audio volume expression.

Output values are clipped to the maximum value.

The output audio volume is given by the relation:

$$\text{output\_volume} = \text{volume} * \text{input\_volume}$$

The default value for *volume* is "1.0".

`precision`

This parameter represents the mathematical precision.

It determines which input sample formats will be allowed, which affects the precision of the volume scaling.

`fixed`

8-bit fixed-point; this limits input sample format to U8, S16, and S32.

`float`

32-bit floating-point; this limits input sample format to FLT. (default)

`double`

64-bit floating-point; this limits input sample format to DBL.

`replaygain`

Choose the behaviour on encountering ReplayGain side data in input frames.

`drop`

Remove ReplayGain side data, ignoring its contents (the default).

`ignore`



Ignore ReplayGain side data, but leave it in the frame.

track

Prefer the track gain, if present.

album

Prefer the album gain, if present.

replaygain\_preamp

Pre-amplification gain in dB to apply to the selected replaygain gain.

Default value for *replaygain\_preamp* is 0.0.

eval

Set when the volume expression is evaluated.

It accepts the following values:

‘once’

only evaluate expression once during the filter initialization, or when the ‘volume’ command is sent

‘frame’

evaluate expression for each incoming frame

Default value is ‘once’.

The volume expression can contain the following parameters.

n

frame number (starting at zero)

nb\_channels

number of channels

nb\_consumed\_samples

number of samples consumed by the filter

nb\_samples

number of samples in the current frame

`pos`

original frame position in the file

`pts`

frame PTS

`sample_rate`

sample rate

`startpts`

PTS at start of stream

`startt`

time at start of stream

`t`

frame time

`tb`

timestamp timebase

`volume`

last set volume value

Note that when `eval` is set to ‘once’ only the *sample\_rate* and *tb* variables are available, all other variables will evaluate to NAN.

### 34.47.1 Commands# TOC

This filter supports the following commands:

`volume`

Modify the volume expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

`replaygain_noclip`

Prevent clipping by limiting the gain applied.

Default value for *replaygain\_noclip* is 1.

### 34.47.2 Examples# TOC

- Halve the input audio volume:

```
volume=volume=0.5
volume=volume=1/2
volume=volume=-6.0206dB
```

In all the above example the named key for *volume* can be omitted, for example like in:

```
volume=0.5
```

- Increase input audio power by 6 decibels using fixed-point precision:

```
volume=volume=6dB:precision=fixed
```

- Fade volume after time 10 with an annihilation period of 5 seconds:

```
volume='if(1t(t,10),1,max(1-(t-10)/5,0))':eval=frame
```

### 34.48 volumedetect# TOC

Detect the volume of the input video.

The filter has no parameters. The input is not modified. Statistics about the volume will be printed in the log when the input stream end is reached.

In particular it will show the mean volume (root mean square), maximum volume (on a per-sample basis), and the beginning of a histogram of the registered volume values (from the maximum value to a cumulated 1/1000 of the samples).

All volumes are in decibels relative to the maximum PCM value.

#### 34.48.1 Examples# TOC

Here is an excerpt of the output:

```
[Parsed_volumedetect_0 0xa23120] mean_volume: -27 dB
[Parsed_volumedetect_0 0xa23120] max_volume: -4 dB
[Parsed_volumedetect_0 0xa23120] histogram_4db: 6
[Parsed_volumedetect_0 0xa23120] histogram_5db: 62
[Parsed_volumedetect_0 0xa23120] histogram_6db: 286
[Parsed_volumedetect_0 0xa23120] histogram_7db: 1042
[Parsed_volumedetect_0 0xa23120] histogram_8db: 2551
[Parsed_volumedetect_0 0xa23120] histogram_9db: 4609
[Parsed_volumedetect_0 0xa23120] histogram_10db: 8409
```

It means that:

- The mean square energy is approximately -27 dB, or  $10^{-2.7}$ .
- The largest sample is at -4 dB, or more precisely between -4 dB and -5 dB.
- There are 6 samples at -4 dB, 62 at -5 dB, 286 at -6 dB, etc.

In other words, raising the volume by +4 dB does not cause any clipping, raising it by +5 dB causes clipping for 6 samples, etc.

## 35 Audio Sources# TOC

Below is a description of the currently available audio sources.

### 35.1 abuffer# TOC

Buffer audio frames, and make them available to the filter chain.

This source is mainly intended for a programmatic use, in particular through the interface defined in `libavfilter/asrc_abuffer.h`.

It accepts the following parameters:

`time_base`

The timebase which will be used for timestamps of submitted frames. It must be either a floating-point number or in *numerator/denominator* form.

`sample_rate`

The sample rate of the incoming audio buffers.

`sample_fmt`

The sample format of the incoming audio buffers. Either a sample format name or its corresponding integer representation from the enum `AVSampleFormat` in `libavutil/samplefmt.h`

`channel_layout`

The channel layout of the incoming audio buffers. Either a channel layout name from `channel_layout_map` in `libavutil/channel_layout.c` or its corresponding integer representation from the `AV_CH_LAYOUT_*` macros in `libavutil/channel_layout.h`

`channels`

The number of channels of the incoming audio buffers. If both *channels* and *channel\_layout* are specified, then they must be consistent.

### 35.1.1 Examples# TOC

```
abuffer=sample_rate=44100:sample_fmt=s16p:channel_layout=stereo
```

will instruct the source to accept planar 16bit signed stereo at 44100Hz. Since the sample format with name "s16p" corresponds to the number 6 and the "stereo" channel layout corresponds to the value 0x3, this is equivalent to:

```
abuffer=sample_rate=44100:sample_fmt=6:channel_layout=0x3
```

## 35.2 aevalsrc# TOC

Generate an audio signal specified by an expression.

This source accepts in input one or more expressions (one for each channel), which are evaluated and used to generate a corresponding audio signal.

This source accepts the following options:

`exprs`

Set the `'|'`-separated expressions list for each separate channel. In case the `channel_layout` option is not specified, the selected channel layout depends on the number of provided expressions. Otherwise the last specified expression is applied to the remaining output channels.

`channel_layout, c`

Set the channel layout. The number of channels in the specified layout must be equal to the number of specified expressions.

`duration, d`

Set the minimum duration of the sourced audio. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax. Note that the resulting duration may be greater than the specified duration, as the generated audio is always cut at the end of a complete frame.

If not specified, or the expressed duration is negative, the audio is supposed to be generated forever.

`nb_samples, n`

Set the number of samples per channel per each output frame, default to 1024.

`sample_rate, s`

Specify the sample rate, default to 44100.

Each expression in *exprs* can contain the following constants:

n

number of the evaluated sample, starting from 0

t

time of the evaluated sample expressed in seconds, starting from 0

s

sample rate

### 35.2.1 Examples# TOC

- Generate silence:

```
aevalsrc=0
```

- Generate a sin signal with frequency of 440 Hz, set sample rate to 8000 Hz:

```
aevalsrc="sin(440*2*PI*t):s=8000"
```

- Generate a two channels signal, specify the channel layout (Front Center + Back Center) explicitly:

```
aevalsrc="sin(420*2*PI*t)|cos(430*2*PI*t):c=FC|BC"
```

- Generate white noise:

```
aevalsrc="-2+random(0)"
```

- Generate an amplitude modulated signal:

```
aevalsrc="sin(10*2*PI*t)*sin(880*2*PI*t)"
```

- Generate 2.5 Hz binaural beats on a 360 Hz carrier:

```
aevalsrc="0.1*sin(2*PI*(360-2.5/2)*t)|0.1*sin(2*PI*(360+2.5/2)*t)"
```

### 35.3 anullsrc# TOC

The null audio source, return unprocessed audio frames. It is mainly useful as a template and to be employed in analysis / debugging tools, or as the source for filters which ignore the input data (for example the sox synth filter).

This source accepts the following options:

`channel_layout, cl`

Specifies the channel layout, and can be either an integer or a string representing a channel layout. The default value of *channel\_layout* is "stereo".

Check the `channel_layout_map` definition in `libavutil/channel_layout.c` for the mapping between strings and channel layout values.

`sample_rate, r`

Specifies the sample rate, and defaults to 44100.

`nb_samples, n`

Set the number of samples per requested frames.

### 35.3.1 Examples# TOC

- Set the sample rate to 48000 Hz and the channel layout to `AV_CH_LAYOUT_MONO`.

```
anullsrc=r=48000:cl=4
```

- Do the same operation with a more obvious syntax:

```
anullsrc=r=48000:cl=mono
```

All the parameters need to be explicitly defined.

## 35.4 flite# TOC

Synthesize a voice utterance using the `libflite` library.

To enable compilation of this filter you need to configure FFmpeg with `--enable-libflite`.

Note that the flite library is not thread-safe.

The filter accepts the following options:

`list_voices`

If set to 1, list the names of the available voices and exit immediately. Default value is 0.

`nb_samples, n`

Set the maximum number of samples per frame. Default value is 512.

`textfile`

Set the filename containing the text to speak.

`text`

Set the text to speak.

`voice, v`

Set the voice to use for the speech synthesis. Default value is `kal`. See also the *list\_voices* option.

### 35.4.1 Examples# TOC

- Read from file `speech.txt`, and synthesize the text using the standard flite voice:

```
flite=textfile=speech.txt
```

- Read the specified text selecting the `slt` voice:

```
flite=text='So fare thee well, poor devil of a Sub-Sub, whose commentator I am':voice=slt
```

- Input text to `ffmpeg`:

```
ffmpeg -f lavfi -i flite=text='So fare thee well, poor devil of a Sub-Sub, whose commentator I am':voice=slt
```

- Make `ffplay` speak the specified text, using `flite` and the `lavfi` device:

```
ffplay -f lavfi flite=text='No more be grieved for which that thou hast done.'
```

For more information about libflite, check: <http://www.speech.cs.cmu.edu/flite/>

## 35.5 sine# TOC

Generate an audio signal made of a sine wave with amplitude 1/8.

The audio signal is bit-exact.

The filter accepts the following options:

`frequency, f`

Set the carrier frequency. Default is 440 Hz.

`beep_factor, b`

Enable a periodic beep every second with frequency *beep\_factor* times the carrier frequency. Default is 0, meaning the beep is disabled.

`sample_rate, r`

Specify the sample rate, default is 44100.

`duration, d`

Specify the duration of the generated audio stream.



`samples_per_frame`

Set the number of samples per output frame, default is 1024.

### 35.5.1 Examples# TOC

- Generate a simple 440 Hz sine wave:

```
sine
```

- Generate a 220 Hz sine wave with a 880 Hz beep each second, for 5 seconds:

```
sine=220:4:d=5  
sine=f=220:b=4:d=5  
sine=frequency=220:beep_factor=4:duration=5
```

## 36 Audio Sinks# TOC

Below is a description of the currently available audio sinks.

### 36.1 abuffersink# TOC

Buffer audio frames, and make them available to the end of filter chain.

This sink is mainly intended for programmatic use, in particular through the interface defined in `libavfilter/buffersink.h` or the options system.

It accepts a pointer to an `AVABufferSinkContext` structure, which defines the incoming buffers' formats, to be passed as the `opaque` parameter to `avfilter_init_filter` for initialization.

### 36.2 anullsink# TOC

Null audio sink; do absolutely nothing with the input audio. It is mainly useful as a template and for use in analysis / debugging tools.

## 37 Video Filters# TOC

When you configure your FFmpeg build, you can disable any of the existing filters using `--disable-filters`. The configure output will show the video filters included in your build.

Below is a description of the currently available video filters.

### 37.1 alphaextract# TOC

Extract the alpha component from the input as a grayscale video. This is especially useful with the *alphamerge* filter.

## 37.2 alphamerge# TOC

Add or replace the alpha component of the primary input with the grayscale value of a second input. This is intended for use with *alphaextract* to allow the transmission or storage of frame sequences that have alpha in a format that doesn't support an alpha channel.

For example, to reconstruct full frames from a normal YUV-encoded video and a separate video created with *alphaextract*, you might use:

```
movie=in_alpha.mkv [alpha]; [in][alpha] alphamerge [out]
```

Since this filter is designed for reconstruction, it operates on frame sequences without considering timestamps, and terminates when either input reaches end of stream. This will cause problems if your encoding pipeline drops frames. If you're trying to apply an image as an overlay to a video stream, consider the *overlay* filter instead.

## 37.3 ass# TOC

Same as the subtitles filter, except that it doesn't require libavcodec and libavformat to work. On the other hand, it is limited to ASS (Advanced Substation Alpha) subtitles files.

This filter accepts the following option in addition to the common options from the subtitles filter:

shaping

Set the shaping engine

Available values are:

'auto'

The default libass shaping engine, which is the best available.

'simple'

Fast, font-agnostic shaper that can do only substitutions

'complex'

Slower shaper using OpenType for substitutions and positioning

The default is `auto`.

## 37.4 atadenoise# TOC

Apply an Adaptive Temporal Averaging Denoiser to the video input.

The filter accepts the following options:

0a

Set threshold A for 1st plane. Default is 0.02. Valid range is 0 to 0.3.

0b

Set threshold B for 1st plane. Default is 0.04. Valid range is 0 to 5.

1a

Set threshold A for 2nd plane. Default is 0.02. Valid range is 0 to 0.3.

1b

Set threshold B for 2nd plane. Default is 0.04. Valid range is 0 to 5.

2a

Set threshold A for 3rd plane. Default is 0.02. Valid range is 0 to 0.3.

2b

Set threshold B for 3rd plane. Default is 0.04. Valid range is 0 to 5.

Threshold A is designed to react on abrupt changes in the input signal and threshold B is designed to react on continuous changes in the input signal.

s

Set number of frames filter will use for averaging. Default is 33. Must be odd number in range [5, 129].

## 37.5 bbox# TOC

Compute the bounding box for the non-black pixels in the input frame luminance plane.

This filter computes the bounding box containing all the pixels with a luminance value greater than the minimum allowed value. The parameters describing the bounding box are printed on the filter log.

The filter accepts the following option:

min\_val

Set the minimal luminance value. Default is 16.

## 37.6 blackdetect# TOC

Detect video intervals that are (almost) completely black. Can be useful to detect chapter transitions, commercials, or invalid recordings. Output lines contains the time for the start, end and duration of the detected black interval expressed in seconds.

In order to display the output lines, you need to set the loglevel at least to the AV\_LOG\_INFO value.

The filter accepts the following options:

`black_min_duration, d`

Set the minimum detected black duration expressed in seconds. It must be a non-negative floating point number.

Default value is 2.0.

`picture_black_ratio_th, pic_th`

Set the threshold for considering a picture "black". Express the minimum value for the ratio:

*`nb_black_pixels / nb_pixels`*

for which a picture is considered black. Default value is 0.98.

`pixel_black_th, pix_th`

Set the threshold for considering a pixel "black".

The threshold expresses the maximum pixel luminance value for which a pixel is considered "black". The provided value is scaled according to the following equation:

*`absolute_threshold = luminance_minimum_value + pixel_black_th * luminance_range_size`*

*`luminance_range_size`* and *`luminance_minimum_value`* depend on the input video format, the range is [0-255] for YUV full-range formats and [16-235] for YUV non full-range formats.

Default value is 0.10.

The following example sets the maximum pixel threshold to the minimum value, and detects only black intervals of 2 or more seconds:

```
blackdetect=d=2:pix_th=0.00
```

## 37.7 blackframe# TOC

Detect frames that are (almost) completely black. Can be useful to detect chapter transitions or commercials. Output lines consist of the frame number of the detected frame, the percentage of blackness, the position in the file if known or -1 and the timestamp in seconds.

In order to display the output lines, you need to set the loglevel at least to the AV\_LOG\_INFO value.

It accepts the following parameters:

`amount`

The percentage of the pixels that have to be below the threshold; it defaults to 98.

`threshold, thresh`

The threshold below which a pixel value is considered black; it defaults to 32.

## 37.8 `blend, tblend`# TOC

Blend two video frames into each other.

The `blend` filter takes two input streams and outputs one stream, the first input is the "top" layer and second input is "bottom" layer. Output terminates when shortest input terminates.

The `tblend` (time blend) filter takes two consecutive frames from one single stream, and outputs the result obtained by blending the new frame on top of the old frame.

A description of the accepted options follows.

`c0_mode`

`c1_mode`

`c2_mode`

`c3_mode`

`all_mode`

Set blend mode for specific pixel component or all pixel components in case of *all\_mode*. Default value is `normal`.

Available values for component modes are:

`'addition'`

`'and'`

`'average'`

`'burn'`

`'darken'`

`'difference'`

`'difference128'`

`'divide'`

`'dodge'`

`'exclusion'`

`'glow'`

`'hardlight'`

'hardmix'  
'lighten'  
'linearlight'  
'multiply'  
'negation'  
'normal'  
'or'  
'overlay'  
'phoenix'  
'pinlight'  
'reflect'  
'screen'  
'softlight'  
'subtract'  
'vividlight'  
'xor'  
c0\_opacity  
c1\_opacity  
c2\_opacity  
c3\_opacity  
all\_opacity

Set blend opacity for specific pixel component or all pixel components in case of *all\_opacity*. Only used in combination with pixel component blend modes.

c0\_expr  
c1\_expr  
c2\_expr  
c3\_expr  
all\_expr

Set blend expression for specific pixel component or all pixel components in case of *all\_expr*. Note that related mode options will be ignored if those are set.

The expressions can use the following variables:

N

The sequential number of the filtered frame, starting from 0.

X

Y

the coordinates of the current sample

W

H

the width and height of currently filtered plane

SW

SH

Width and height scale depending on the currently filtered plane. It is the ratio between the corresponding luma plane number of pixels and the current plane ones. E.g. for YUV4:2:0 the values are 1, 1 for the luma plane, and 0.5, 0.5 for chroma planes.

T

Time of the current frame, expressed in seconds.

TOP, A

Value of pixel component at current location for first video frame (top layer).

BOTTOM, B

Value of pixel component at current location for second video frame (bottom layer).

shortest

Force termination when the shortest input terminates. Default is 0. This option is only defined for the blend filter.

repeatlast

Continue applying the last bottom frame after the end of the stream. A value of 0 disable the filter after the last frame of the bottom layer is reached. Default is 1. This option is only defined for the blend filter.

### 37.8.1 Examples# TOC

- Apply transition from bottom layer to top layer in first 10 seconds:

```
blend=all_expr='A*(if(gte(T,10),1,T/10))+B*(1-(if(gte(T,10),1,T/10)))'
```

- Apply 1x1 checkerboard effect:

```
blend=all_expr='if(eq(mod(X,2),mod(Y,2)),A,B)'
```

- Apply uncover left effect:

```
blend=all_expr='if(gte(N*SW+X,W),A,B)'
```

- Apply uncover down effect:

```
blend=all_expr='if(gte(Y-N*SH,0),A,B)'
```

- Apply uncover up-left effect:

```
blend=all_expr='if(gte(T*SH*40+Y,H)*gte((T*40*SW+X)*W/H,W),A,B)'
```

- Display differences between the current and the previous frame:

```
tblend=all_mode=difference128
```

## 37.9 boxblur# TOC

Apply a boxblur algorithm to the input video.

It accepts the following parameters:

```
luma_radius, lr  
luma_power, lp  
chroma_radius, cr  
chroma_power, cp  
alpha_radius, ar  
alpha_power, ap
```

A description of the accepted options follows.

```
luma_radius, lr  
chroma_radius, cr  
alpha_radius, ar
```

Set an expression for the box radius in pixels used for blurring the corresponding input plane.

The radius value must be a non-negative number, and must not be greater than the value of the expression  $\min(w, h) / 2$  for the luma and alpha planes, and of  $\min(cw, ch) / 2$  for the chroma planes.

Default value for `luma_radius` is "2". If not specified, `chroma_radius` and `alpha_radius` default to the corresponding value set for `luma_radius`.

The expressions can contain the following constants:

```
w  
h
```

The input width and height in pixels.

```
cw  
ch
```



The input chroma image width and height in pixels.

`hsub`  
`vsub`

The horizontal and vertical chroma subsample values. For example, for the pixel format "yuv422p", *hsub* is 2 and *vsub* is 1.

`luma_power, lp`  
`chroma_power, cp`  
`alpha_power, ap`

Specify how many times the boxblur filter is applied to the corresponding plane.

Default value for `luma_power` is 2. If not specified, `chroma_power` and `alpha_power` default to the corresponding value set for `luma_power`.

A value of 0 will disable the effect.

### 37.9.1 Examples# TOC

- Apply a boxblur filter with the luma, chroma, and alpha radii set to 2:

```
boxblur=luma_radius=2:luma_power=1  
boxblur=2:1
```

- Set the luma radius to 2, and alpha and chroma radius to 0:

```
boxblur=2:1:cr=0:ar=0
```

- Set the luma and chroma radii to a fraction of the video dimension:

```
boxblur=luma_radius=min(h\,w)/10:luma_power=1:chroma_radius=min(cw\,ch)/10:chroma_power=1
```

### 37.10 codecview# TOC

Visualize information exported by some codecs.

Some codecs can export information through frames using side-data or other means. For example, some MPEG based codecs export motion vectors through the *export\_mvs* flag in the codec *flags2* option.

The filter accepts the following option:

`mv`

Set motion vectors to visualize.

Available flags for *mv* are:

‘pf’

forward predicted MVs of P-frames

‘bf’

forward predicted MVs of B-frames

‘bb’

backward predicted MVs of B-frames

### 37.10.1 Examples# TOC

- Visualizes multi-directionals MVs from P and B-Frames using `ffplay`:

```
ffplay -flags2 +export_mvs input.mpg -vf codecview=mv=pf+bf+bb
```

### 37.11 colorbalance# TOC

Modify intensity of primary colors (red, green and blue) of input frames.

The filter allows an input frame to be adjusted in the shadows, midtones or highlights regions for the red-cyan, green-magenta or blue-yellow balance.

A positive adjustment value shifts the balance towards the primary color, a negative value towards the complementary color.

The filter accepts the following options:

rs  
gs  
bs

Adjust red, green and blue shadows (darkest pixels).

rm  
gm  
bm

Adjust red, green and blue midtones (medium pixels).

rh  
gh  
bh

Adjust red, green and blue highlights (brightest pixels).

Allowed ranges for options are  $[-1.0, 1.0]$ . Defaults are 0.

### 37.11.1 Examples# TOC

- Add red color cast to shadows:

```
colorbalance=rs=.3
```

## 37.12 colorkey# TOC

RGB colorspace color keying.

The filter accepts the following options:

**color**

The color which will be replaced with transparency.

**similarity**

Similarity percentage with the key color.

0.01 matches only the exact key color, while 1.0 matches everything.

**blend**

Blend percentage.

0.0 makes pixels either fully transparent, or not transparent at all.

Higher values result in semi-transparent pixels, with a higher transparency the more similar the pixels color is to the key color.

### 37.12.1 Examples# TOC

- Make every green pixel in the input image transparent:

```
ffmpeg -i input.png -vf colorkey=green out.png
```

- Overlay a greenscreen-video on top of a static background image.

```
ffmpeg -i background.png -i video.mp4 -filter_complex "[1:v]colorkey=0x3BBD1E:0.3:0.2[ckout];[0:v][ckout]overlay[out]" -map "[out]" output.flv
```

## 37.13 colorlevels# TOC

Adjust video input frames using levels.

The filter accepts the following options:

rimin  
gimin  
bimin  
aimin

Adjust red, green, blue and alpha input black point. Allowed ranges for options are  $[-1.0, 1.0]$ . Defaults are 0.

rimax  
gimax  
bimax  
aimax

Adjust red, green, blue and alpha input white point. Allowed ranges for options are  $[-1.0, 1.0]$ . Defaults are 1.

Input levels are used to lighten highlights (bright tones), darken shadows (dark tones), change the balance of bright and dark tones.

romin  
gomin  
bomin  
aomin

Adjust red, green, blue and alpha output black point. Allowed ranges for options are  $[0, 1.0]$ . Defaults are 0.

romax  
gomax  
bomax  
aomax

Adjust red, green, blue and alpha output white point. Allowed ranges for options are  $[0, 1.0]$ . Defaults are 1.

Output levels allows manual selection of a constrained output level range.

### 37.13.1 Examples# TOC

- Make video output darker:

```
colorlevels=rimin=0.058:gimin=0.058:bimin=0.058
```

- Increase contrast:

```
colorlevels=rimin=0.039:gimin=0.039:bimin=0.039:rimax=0.96:gimax=0.96:bimax=0.96
```

- Make video output lighter:

```
colorlevels=rimax=0.902:gimax=0.902:bimax=0.902
```

- Increase brightness:

```
colorlevels=romin=0.5:gomin=0.5:bomin=0.5
```

## 37.14 colorchannelmixer# TOC

Adjust video input frames by re-mixing color channels.

This filter modifies a color channel by adding the values associated to the other channels of the same pixels. For example if the value to modify is red, the output value will be:

$$red = red * rr + blue * rb + green * rg + alpha * ra$$

The filter accepts the following options:

rr  
rg  
rb  
ra

Adjust contribution of input red, green, blue and alpha channels for output red channel. Default is 1 for *rr*, and 0 for *rg*, *rb* and *ra*.

gr  
gg  
gb  
ga

Adjust contribution of input red, green, blue and alpha channels for output green channel. Default is 1 for *gg*, and 0 for *gr*, *gb* and *ga*.

br  
bg  
bb  
ba

Adjust contribution of input red, green, blue and alpha channels for output blue channel. Default is 1 for *bb*, and 0 for *br*, *bg* and *ba*.

ar  
ag  
ab  
aa

Adjust contribution of input red, green, blue and alpha channels for output alpha channel. Default is 1 for *aa*, and 0 for *ar*, *ag* and *ab*.

Allowed ranges for options are  $[-2.0, 2.0]$ .

### 37.14.1 Examples# TOC

- Convert source to grayscale:

```
colorchannelmixer=.3:.4:.3:0:.3:.4:.3:0:.3:.4:.3
```

- Simulate sepia tones:

```
colorchannelmixer=.393:.769:.189:0:.349:.686:.168:0:.272:.534:.131
```

### 37.15 colormatrix# TOC

Convert color matrix.

The filter accepts the following options:

```
src  
dst
```

Specify the source and destination color matrix. Both values must be specified.

The accepted values are:

‘bt709’

BT.709

‘bt601’

BT.601

‘smpte240m’

SMPTE-240M

‘fcc’

FCC

For example to convert from BT.601 to SMPTE-240M, use the command:

```
colormatrix=bt601:smpte240m
```

## 37.16 copy# TOC

Copy the input source unchanged to the output. This is mainly useful for testing purposes.

## 37.17 crop# TOC

Crop the input video to given dimensions.

It accepts the following parameters:

`w, out_w`

The width of the output video. It defaults to `iw`. This expression is evaluated only once during the filter configuration, or when the ‘`w`’ or ‘`out_w`’ command is sent.

`h, out_h`

The height of the output video. It defaults to `ih`. This expression is evaluated only once during the filter configuration, or when the ‘`h`’ or ‘`out_h`’ command is sent.

`x`

The horizontal position, in the input video, of the left edge of the output video. It defaults to  $(in\_w - out\_w) / 2$ . This expression is evaluated per-frame.

`y`

The vertical position, in the input video, of the top edge of the output video. It defaults to  $(in\_h - out\_h) / 2$ . This expression is evaluated per-frame.

`keep_aspect`

If set to 1 will force the output display aspect ratio to be the same of the input, by changing the output sample aspect ratio. It defaults to 0.

The `out_w`, `out_h`, `x`, `y` parameters are expressions containing the following constants:

`x`

`y`

The computed values for `x` and `y`. They are evaluated for each new frame.

`in_w`

`in_h`

The input width and height.

`iw`  
`ih`

These are the same as *in\_w* and *in\_h*.

`out_w`  
`out_h`

The output (cropped) width and height.

`ow`  
`oh`

These are the same as *out\_w* and *out\_h*.

`a`

same as *iw / ih*

`sar`

input sample aspect ratio

`dar`

input display aspect ratio, it is the same as  $(iw / ih) * sar$

`hsub`  
`vsub`

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

`n`

The number of the input frame, starting from 0.

`pos`

the position in the file of the input frame, NAN if unknown

`t`

The timestamp expressed in seconds. It's NAN if the input timestamp is unknown.

The expression for *out\_w* may depend on the value of *out\_h*, and the expression for *out\_h* may depend on *out\_w*, but they cannot depend on *x* and *y*, as *x* and *y* are evaluated after *out\_w* and *out\_h*.



The  $x$  and  $y$  parameters specify the expressions for the position of the top-left corner of the output (non-cropped) area. They are evaluated for each frame. If the evaluated value is not valid, it is approximated to the nearest valid value.

The expression for  $x$  may depend on  $y$ , and the expression for  $y$  may depend on  $x$ .

### 37.17.1 Examples# TOC

- Crop area with size 100x100 at position (12,34).

```
crop=100:100:12:34
```

Using named options, the example above becomes:

```
crop=w=100:h=100:x=12:y=34
```

- Crop the central input area with size 100x100:

```
crop=100:100
```

- Crop the central input area with size 2/3 of the input video:

```
crop=2/3*in_w:2/3*in_h
```

- Crop the input video central square:

```
crop=out_w=in_h  
crop=in_h
```

- Delimit the rectangle with the top-left corner placed at position 100:100 and the right-bottom corner corresponding to the right-bottom corner of the input image.

```
crop=in_w-100:in_h-100:100:100
```

- Crop 10 pixels from the left and right borders, and 20 pixels from the top and bottom borders

```
crop=in_w-2*10:in_h-2*20
```

- Keep only the bottom right quarter of the input image:

```
crop=in_w/2:in_h/2:in_w/2:in_h/2
```

- Crop height for getting Greek harmony:

```
crop=in_w:1/PHI*in_w
```

- Apply trembling effect:

```
crop=in_w/2:in_h/2:(in_w-out_w)/2+((in_w-out_w)/2)*sin(n/10):(in_h-out_h)/2+((in_h-out_h)/2)*sin(n/7)
```

- Apply erratic camera effect depending on timestamp:

```
crop=in_w/2:in_h/2:(in_w-out_w)/2+((in_w-out_w)/2)*sin(t*10):(in_h-out_h)/2 +((in_h-out_h)/2)*sin(t*13)"
```

- Set x depending on the value of y:

```
crop=in_w/2:in_h/2:y:10+10*sin(n/10)
```

### 37.17.2 Commands# TOC

This filter supports the following commands:

```
w, out_w  
h, out_h  
x  
y
```

Set width/height of the output video and the horizontal/vertical position in the input video. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.18 cropdetect# TOC

Auto-detect the crop size.

It calculates the necessary cropping parameters and prints the recommended parameters via the logging system. The detected dimensions correspond to the non-black area of the input video.

It accepts the following parameters:

```
limit
```

Set higher black value threshold, which can be optionally specified from nothing (0) to everything (255 for 8bit based formats). An intensity value greater to the set value is considered non-black. It defaults to 24. You can also specify a value between 0.0 and 1.0 which will be scaled depending on the bitdepth of the pixel format.

```
round
```

The value which the width/height should be divisible by. It defaults to 16. The offset is automatically adjusted to center the video. Use 2 to get only even dimensions (needed for 4:2:2 video). 16 is best when encoding to most video codecs.

```
reset_count, reset
```

Set the counter that determines after how many frames cropdetect will reset the previously detected largest video area and start over to detect the current optimal crop area. Default value is 0.

This can be useful when channel logos distort the video area. 0 indicates 'never reset', and returns the largest area encountered during playback.

## 37.19 curves# TOC

Apply color adjustments using curves.

This filter is similar to the Adobe Photoshop and GIMP curves tools. Each component (red, green and blue) has its values defined by  $N$  key points tied from each other using a smooth curve. The x-axis represents the pixel values from the input frame, and the y-axis the new pixel values to be set for the output frame.

By default, a component curve is defined by the two points  $(0;0)$  and  $(1;1)$ . This creates a straight line where each original pixel value is "adjusted" to its own value, which means no change to the image.

The filter allows you to redefine these two points and add some more. A new curve (using a natural cubic spline interpolation) will be defined to pass smoothly through all these new coordinates. The new defined points need to be strictly increasing over the x-axis, and their x and y values must be in the  $[0;1]$  interval. If the computed curves happened to go outside the vector spaces, the values will be clipped accordingly.

If there is no key point defined in  $x=0$ , the filter will automatically insert a  $(0;0)$  point. In the same way, if there is no key point defined in  $x=1$ , the filter will automatically insert a  $(1;1)$  point.

The filter accepts the following options:

`preset`

Select one of the available color presets. This option can be used in addition to the `r`, `g`, `b` parameters; in this case, the later options take priority on the preset values. Available presets are:

```
'none'
'color_negative'
'cross_process'
'darker'
'increase_contrast'
'lighter'
'linear_contrast'
'medium_contrast'
'negative'
'strong_contrast'
'vintage'
```

Default is none.

`master, m`

Set the master key points. These points will define a second pass mapping. It is sometimes called a "luminance" or "value" mapping. It can be used with `r`, `g`, `b` or `all` since it acts like a post-processing LUT.

`red, r`

Set the key points for the red component.

`green, g`

Set the key points for the green component.

`blue, b`

Set the key points for the blue component.

`all`

Set the key points for all components (not including master). Can be used in addition to the other key points component options. In this case, the unset component(s) will fallback on this `all` setting.

`psfile`

Specify a Photoshop curves file (`.asv`) to import the settings from.

To avoid some filtergraph syntax conflicts, each key points list need to be defined using the following syntax: `x0/y0 x1/y1 x2/y2 ....`

### 37.19.1 Examples# TOC

- Increase slightly the middle level of blue:

```
curves=blue='0.5/0.58'
```

- Vintage effect:

```
curves=r='0/0.11 .42/.51 1/0.95':g='0.50/0.48':b='0/0.22 .49/.44 1/0.8'
```

Here we obtain the following coordinates for each components:

*red*

```
(0;0.11) (0.42;0.51) (1;0.95)
```

*green*

```
(0;0) (0.50;0.48) (1;1)
```

*blue*

```
(0;0.22) (0.49;0.44) (1;0.80)
```

- The previous example can also be achieved with the associated built-in preset:

```
curves=preset=vintage
```

- Or simply:

```
curves=vintage
```

- Use a Photoshop preset and redefine the points of the green component:

```
curves=psfile='MyCurvesPresets/purple.asv':green='0.45/0.53'
```

## 37.20 dctdnoiz# TOC

Denoise frames using 2D DCT (frequency domain filtering).

This filter is not designed for real time.

The filter accepts the following options:

`sigma, s`

Set the noise sigma constant.

This *sigma* defines a hard threshold of  $3 * \text{sigma}$ ; every DCT coefficient (absolute value) below this threshold will be dropped.

If you need a more advanced filtering, see `expr`.

Default is 0.

`overlap`

Set number overlapping pixels for each block. Since the filter can be slow, you may want to reduce this value, at the cost of a less effective filter and the risk of various artefacts.

If the overlapping value doesn't permit processing the whole input width or height, a warning will be displayed and according borders won't be denoised.

Default value is *blocksize-1*, which is the best possible setting.

`expr, e`

Set the coefficient factor expression.

For each coefficient of a DCT block, this expression will be evaluated as a multiplier value for the coefficient.

If this option is set, the `sigma` option will be ignored.

The absolute value of the coefficient can be accessed through the `c` variable.

`n`

Set the *blocksize* using the number of bits.  $1 < n$  defines the *blocksize*, which is the width and height of the processed blocks.

The default value is 3 (8x8) and can be raised to 4 for a *blocksize* of 16x16. Note that changing this setting has huge consequences on the speed processing. Also, a larger block size does not necessarily means a better de-noising.

### 37.20.1 Examples# TOC

Apply a denoise with a `sigma` of 4.5:

```
dctdnoiz=4.5
```

The same operation can be achieved using the expression system:

```
dctdnoiz=e='gte(c, 4.5*3)'
```

Violent denoise using a block size of 16x16:

```
dctdnoiz=15:n=4
```

### 37.21 deband# TOC

Remove banding artifacts from input video. It works by replacing banded pixels with average value of referenced pixels.

The filter accepts the following options:

```
1thr  
2thr  
3thr  
4thr
```

Set banding detection threshold for each plane. Default is 0.02. Valid range is 0.00003 to 0.5. If difference between current pixel and reference pixel is less than threshold, it will be considered as banded.

`range, r`

Banding detection range in pixels. Default is 16. If positive, random number in range 0 to set value will be used. If negative, exact absolute value will be used. The range defines square of four pixels around current pixel.

`direction, d`

Set direction in radians from which four pixel will be compared. If positive, random direction from 0 to set direction will be picked. If negative, exact of absolute value will be picked. For example direction 0,  $-\pi$  or  $-2\pi$  radians will pick only pixels on same row and  $-\pi/2$  will pick only pixels on same column.

`blur`

If enabled, current pixel is compared with average value of all four surrounding pixels. The default is enabled. If disabled current pixel is compared with all four surrounding pixels. The pixel is considered banded if only all four differences with surrounding pixels are less than threshold.

## 37.22 decimate# TOC

Drop duplicated frames at regular intervals.

The filter accepts the following options:

`cycle`

Set the number of frames from which one will be dropped. Setting this to  $N$  means one frame in every batch of  $N$  frames will be dropped. Default is 5.

`dupthresh`

Set the threshold for duplicate detection. If the difference metric for a frame is less than or equal to this value, then it is declared as duplicate. Default is 1.1

`scthresh`

Set scene change threshold. Default is 15.

`blockx`

`blocky`

Set the size of the x and y-axis blocks used during metric calculations. Larger blocks give better noise suppression, but also give worse detection of small movements. Must be a power of two. Default is 32.

`ppsrc`

Mark main input as a pre-processed input and activate clean source input stream. This allows the input to be pre-processed with various filters to help the metrics calculation while keeping the frame selection lossless. When set to 1, the first stream is for the pre-processed input, and the second stream is the clean source from where the kept frames are chosen. Default is 0.

chroma

Set whether or not chroma is considered in the metric calculations. Default is 1.

### 37.23 deflate# TOC

Apply deflate effect to the video.

This filter replaces the pixel by the local(3x3) average by taking into account only values lower than the pixel.

It accepts the following options:

threshold0  
threshold1  
threshold2  
threshold3

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

### 37.24 dejudder# TOC

Remove judder produced by partially interlaced telecined content.

Judder can be introduced, for instance, by pullup filter. If the original source was partially telecined content then the output of `pullup`, `dejudder` will have a variable frame rate. May change the recorded frame rate of the container. Aside from that change, this filter will not affect constant frame rate video.

The option available in this filter is:

cycle

Specify the length of the window over which the judder repeats.

Accepts any integer greater than 1. Useful values are:

‘4’

If the original was telecined from 24 to 30 fps (Film to NTSC).

‘5’

If the original was telecined from 25 to 30 fps (PAL to NTSC).

‘20’

If a mixture of the two.



The default is '4'.

## 37.25 delogo# TOC

Suppress a TV station logo by a simple interpolation of the surrounding pixels. Just set a rectangle covering the logo and watch it disappear (and sometimes something even uglier appear - your mileage may vary).

It accepts the following parameters:

`x`  
`y`

Specify the top left corner coordinates of the logo. They must be specified.

`w`  
`h`

Specify the width and height of the logo to clear. They must be specified.

`band`, `t`

Specify the thickness of the fuzzy edge of the rectangle (added to `w` and `h`). The default value is 4.

`show`

When set to 1, a green rectangle is drawn on the screen to simplify finding the right `x`, `y`, `w`, and `h` parameters. The default value is 0.

The rectangle is drawn on the outermost pixels which will be (partly) replaced with interpolated values. The values of the next pixels immediately outside this rectangle in each direction will be used to compute the interpolated pixel values inside the rectangle.

### 37.25.1 Examples# TOC

- Set a rectangle covering the area with top left corner coordinates 0,0 and size 100x77, and a band of size 10:

```
delogo=x=0:y=0:w=100:h=77:band=10
```

## 37.26 deshake# TOC

Attempt to fix small changes in horizontal and/or vertical shift. This filter helps remove camera shake from hand-holding a camera, bumping a tripod, moving on a vehicle, etc.

The filter accepts the following options:

x  
y  
w  
h

Specify a rectangular area where to limit the search for motion vectors. If desired the search for motion vectors can be limited to a rectangular area of the frame defined by its top left corner, width and height. These parameters have the same meaning as the drawbox filter which can be used to visualise the position of the bounding box.

This is useful when simultaneous movement of subjects within the frame might be confused for camera motion by the motion vector search.

If any or all of  $x$ ,  $y$ ,  $w$  and  $h$  are set to -1 then the full frame is used. This allows later options to be set without specifying the bounding box for the motion vector search.

Default - search the whole frame.

rx  
ry

Specify the maximum extent of movement in x and y directions in the range 0-64 pixels. Default 16.

edge

Specify how to generate pixels to fill blanks at the edge of the frame. Available values are:

`'blank, 0'`

Fill zeroes at blank locations

`'original, 1'`

Original image at blank locations

`'clamp, 2'`

Extruded edge value at blank locations

`'mirror, 3'`

Mirrored edge at blank locations

Default value is `'mirror'`.

blocksize

Specify the blocksize to use for motion search. Range 4-128 pixels, default 8.

contrast

Specify the contrast threshold for blocks. Only blocks with more than the specified contrast (difference between darkest and lightest pixels) will be considered. Range 1-255, default 125.

search

Specify the search strategy. Available values are:

`'exhaustive, 0'`

Set exhaustive search

`'less, 1'`

Set less exhaustive search.

Default value is `'exhaustive'`.

filename

If set then a detailed log of the motion search is written to the specified file.

opencl

If set to 1, specify using OpenCL capabilities, only available if FFmpeg was configured with `--enable-opencl`. Default value is 0.

## 37.27 detelecine# TOC

Apply an exact inverse of the telecine operation. It requires a predefined pattern specified using the pattern option which must be the same as that passed to the telecine filter.

This filter accepts the following options:

first\_field

`'top, t'`

top field first

`'bottom, b'`

bottom field first The default value is top.

pattern

A string of numbers representing the pulldown pattern you wish to apply. The default value is 23.

`start_frame`

A number representing position of the first frame with respect to the telecine pattern. This is to be used if the stream is cut. The default value is 0.

## 37.28 dilation# TOC

Apply dilation effect to the video.

This filter replaces the pixel by the local(3x3) maximum.

It accepts the following options:

`threshold0`  
`threshold1`  
`threshold2`  
`threshold3`

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

`coordinates`

Flag which specifies the pixel to refer to. Default is 255 i.e. all eight pixels are used.

Flags to local 3x3 coordinates maps like this:

1 2 3 4 5 6 7 8

## 37.29 drawbox# TOC

Draw a colored box on the input image.

It accepts the following parameters:

`x`  
`y`

The expressions which specify the top left corner coordinates of the box. It defaults to 0.

`width, w`  
`height, h`

The expressions which specify the width and height of the box; if 0 they are interpreted as the input width and height. It defaults to 0.

`color, c`

Specify the color of the box to write. For the general syntax of this option, check the "Color" section in the ffmpeg-utils manual. If the special value `invert` is used, the box edge color is the same as the video with inverted luma.

`thickness, t`

The expression which sets the thickness of the box edge. Default value is 3.

See below for the list of accepted constants.

The parameters for  $x$ ,  $y$ ,  $w$  and  $h$  and  $t$  are expressions containing the following constants:

`dar`

The input display aspect ratio, it is the same as  $(w / h) * sar$ .

`hsub`

`vsub`

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

`in_h, ih`

`in_w, iw`

The input width and height.

`sar`

The input sample aspect ratio.

`x`

`y`

The x and y offset coordinates where the box is drawn.

`w`

`h`

The width and height of the drawn box.

`t`

The thickness of the drawn box.

These constants allow the  $x$ ,  $y$ ,  $w$ ,  $h$  and  $t$  expressions to refer to each other, so you may for example specify  $y=x/dar$  or  $h=w/dar$ .

### 37.29.1 Examples# TOC

- Draw a black box around the edge of the input image:

```
drawbox
```

- Draw a box with color red and an opacity of 50%:

```
drawbox=10:20:200:60:red@0.5
```

The previous example can be specified as:

```
drawbox=x=10:y=20:w=200:h=60:color=red@0.5
```

- Fill the box with pink color:

```
drawbox=x=10:y=10:w=100:h=100:color=pink@0.5:t=max
```

- Draw a 2-pixel red 2.40:1 mask:

```
drawbox=x=-t:y=0.5*(ih-iw/2.4)-t:w=iw+t*2:h=iw/2.4+t*2:t=2:c=red
```

### 37.30 drawgraph, adrawgraph# TOC

Draw a graph using input video or audio metadata.

It accepts the following parameters:

m1

Set 1st frame metadata key from which metadata values will be used to draw a graph.

fg1

Set 1st foreground color expression.

m2

Set 2nd frame metadata key from which metadata values will be used to draw a graph.

fg2

Set 2nd foreground color expression.

m3

Set 3rd frame metadata key from which metadata values will be used to draw a graph.

fg3

Set 3rd foreground color expression.

m4

Set 4th frame metadata key from which metadata values will be used to draw a graph.

fg4

Set 4th foreground color expression.

min

Set minimal value of metadata value.

max

Set maximal value of metadata value.

bg

Set graph background color. Default is white.

mode

Set graph mode.

Available values for mode is:

‘bar’  
‘dot’  
‘line’

Default is line.

slide

Set slide mode.

Available values for slide is:

‘frame’

Draw new frame when right border is reached.

‘replace’

Replace old columns with new ones.

`'scroll'`

Scroll from right to left.

`'rscroll'`

Scroll from left to right.

Default is `frame`.

`size`

Set size of graph video. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. The default value is 900x256.

The foreground color expressions can use the following variables:

`MIN`

Minimal value of metadata value.

`MAX`

Maximal value of metadata value.

`VAL`

Current metadata key value.

The color is defined as 0xAABBGRR.

Example using metadata from signalstats filter:

```
signalstats,drawgraph=lavfi.signalstats.YAVG:min=0:max=255
```

Example using metadata from ebur128 filter:

```
ebur128=metadata=1,adrawgraph=lavfi.r128.M:min=-120:max=5
```

## 37.31 drawgrid# TOC

Draw a grid on the input image.

It accepts the following parameters:

`x`

`y`



The expressions which specify the coordinates of some point of grid intersection (meant to configure offset). Both default to 0.

width, *w*  
height, *h*

The expressions which specify the width and height of the grid cell, if 0 they are interpreted as the input width and height, respectively, minus *thickness*, so image gets framed. Default to 0.

color, *c*

Specify the color of the grid. For the general syntax of this option, check the "Color" section in the ffmpeg-utils manual. If the special value *invert* is used, the grid color is the same as the video with inverted luma.

thickness, *t*

The expression which sets the thickness of the grid line. Default value is 1.

See below for the list of accepted constants.

The parameters for *x*, *y*, *w* and *h* and *t* are expressions containing the following constants:

*dar*

The input display aspect ratio, it is the same as  $(w / h) * sar$ .

*hsub*  
*vsub*

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

*in\_h*, *ih*  
*in\_w*, *iw*

The input grid cell width and height.

*sar*

The input sample aspect ratio.

*x*  
*y*

The x and y coordinates of some point of grid intersection (meant to configure offset).

*w*

*h*

The width and height of the drawn cell.

*t*

The thickness of the drawn cell.

These constants allow the *x*, *y*, *w*, *h* and *t* expressions to refer to each other, so you may for example specify *y*=*x*/*dar* or *h*=*w*/*dar*.

### 37.31.1 Examples# TOC

- Draw a grid with cell 100x100 pixels, thickness 2 pixels, with color red and an opacity of 50%:

```
drawgrid=width=100:height=100:thickness=2:color=red@0.5
```

- Draw a white 3x3 grid with an opacity of 50%:

```
drawgrid=w=iw/3:h=ih/3:t=2:c=white@0.5
```

### 37.32 drawtext# TOC

Draw a text string or text from a specified file on top of a video, using the libfreetype library.

To enable compilation of this filter, you need to configure FFmpeg with `--enable-libfreetype`. To enable default font fallback and the *font* option you need to configure FFmpeg with `--enable-libfontconfig`. To enable the *text\_shaping* option, you need to configure FFmpeg with `--enable-libfribidi`.

#### 37.32.1 Syntax# TOC

It accepts the following parameters:

*box*

Used to draw a box around text using the background color. The value must be either 1 (enable) or 0 (disable). The default value of *box* is 0.

*boxborderw*

Set the width of the border to be drawn around the box using *boxcolor*. The default value of *boxborderw* is 0.

*boxcolor*

The color to be used for drawing box around text. For the syntax of this option, check the "Color" section in the ffmpeg-utils manual.

The default value of *boxcolor* is "white".

`borderw`

Set the width of the border to be drawn around the text using *bordercolor*. The default value of *borderw* is 0.

`bordercolor`

Set the color to be used for drawing border around text. For the syntax of this option, check the "Color" section in the ffmpeg-utils manual.

The default value of *bordercolor* is "black".

`expansion`

Select how the *text* is expanded. Can be either none, `strftime` (deprecated) or `normal` (default). See the Text expansion section below for details.

`fix_bounds`

If true, check and fix text coords to avoid clipping.

`fontcolor`

The color to be used for drawing fonts. For the syntax of this option, check the "Color" section in the ffmpeg-utils manual.

The default value of *fontcolor* is "black".

`fontcolor_expr`

String which is expanded the same way as *text* to obtain dynamic *fontcolor* value. By default this option has empty value and is not processed. When this option is set, it overrides *fontcolor* option.

`font`

The font family to be used for drawing text. By default Sans.

`fontfile`

The font file to be used for drawing text. The path must be included. This parameter is mandatory if the fontconfig support is disabled.

`draw`

This option does not exist, please see the timeline system

alpha

Draw the text applying alpha blending. The value can be either a number between 0.0 and 1.0 The expression accepts the same variables *x*, *y* do. The default value is 1. Please see `fontcolor_expr`

fontsize

The font size to be used for drawing text. The default value of *fontsize* is 16.

text\_shaping

If set to 1, attempt to shape the text (for example, reverse the order of right-to-left text and join Arabic characters) before drawing it. Otherwise, just draw the text exactly as given. By default 1 (if supported).

ft\_load\_flags

The flags to be used for loading the fonts.

The flags map the corresponding flags supported by libfreetype, and are a combination of the following values:

*default*  
*no\_scale*  
*no\_hinting*  
*render*  
*no\_bitmap*  
*vertical\_layout*  
*force\_autohint*  
*crop\_bitmap*  
*pedantic*  
*ignore\_global\_advance\_width*  
*no\_recurse*  
*ignore\_transform*  
*monochrome*  
*linear\_design*  
*no\_autohint*

Default value is "default".

For more information consult the documentation for the `FT_LOAD_*` libfreetype flags.

shadowcolor

The color to be used for drawing a shadow behind the drawn text. For the syntax of this option, check the "Color" section in the `ffmpeg-utils` manual.

The default value of *shadowcolor* is "black".

shadowx  
shadowy

The x and y offsets for the text shadow position with respect to the position of the text. They can be either positive or negative values. The default value for both is "0".

start\_number

The starting frame number for the *n/frame\_num* variable. The default value is "0".

tabsize

The size in number of spaces to use for rendering the tab. Default value is 4.

timecode

Set the initial timecode representation in "hh:mm:ss[;.:]ff" format. It can be used with or without text parameter. *timecode\_rate* option must be specified.

timecode\_rate, rate, r

Set the timecode frame rate (timecode only).

text

The text string to be drawn. The text must be a sequence of UTF-8 encoded characters. This parameter is mandatory if no file is specified with the parameter *textfile*.

textfile

A text file containing text to be drawn. The text must be a sequence of UTF-8 encoded characters.

This parameter is mandatory if no text string is specified with the parameter *text*.

If both *text* and *textfile* are specified, an error is thrown.

reload

If set to 1, the *textfile* will be reloaded before each frame. Be sure to update it atomically, or it may be read partially, or even fail.

x  
y

The expressions which specify the offsets where text will be drawn within the video frame. They are relative to the top/left border of the output image.

The default value of  $x$  and  $y$  is "0".

See below for the list of accepted constants and functions.

The parameters for  $x$  and  $y$  are expressions containing the following constants and functions:

`dar`

input display aspect ratio, it is the same as  $(w / h) * sar$

`hsub`

`vsub`

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

`line_h, lh`

the height of each text line

`main_h, h, H`

the input height

`main_w, w, W`

the input width

`max_glyph_a, ascent`

the maximum distance from the baseline to the highest/upper grid coordinate used to place a glyph outline point, for all the rendered glyphs. It is a positive value, due to the grid's orientation with the Y axis upwards.

`max_glyph_d, descent`

the maximum distance from the baseline to the lowest grid coordinate used to place a glyph outline point, for all the rendered glyphs. This is a negative value, due to the grid's orientation, with the Y axis upwards.

`max_glyph_h`

maximum glyph height, that is the maximum height for all the glyphs contained in the rendered text, it is equivalent to *ascent - descent*.

`max_glyph_w`

maximum glyph width, that is the maximum width for all the glyphs contained in the rendered text

`n`

the number of input frame, starting from 0

`rand(min, max)`

return a random number included between *min* and *max*

`sar`

The input sample aspect ratio.

`t`

timestamp expressed in seconds, NAN if the input timestamp is unknown

`text_h, th`

the height of the rendered text

`text_w, tw`

the width of the rendered text

`x`

`y`

the x and y offset coordinates where the text is drawn.

These parameters allow the *x* and *y* expressions to refer each other, so you can for example specify `y=x/dar`.

### 37.32.2 Text expansion# TOC

If `expansion` is set to `strftime`, the filter recognizes `strftime()` sequences in the provided text and expands them accordingly. Check the documentation of `strftime()`. This feature is deprecated.

If `expansion` is set to `none`, the text is printed verbatim.

If `expansion` is set to `normal` (which is the default), the following expansion mechanism is used.

The backslash character `'\'`, followed by any character, always expands to the second character.

Sequence of the form `%{ . . . }` are expanded. The text between the braces is a function name, possibly followed by arguments separated by `':'`. If the arguments contain special characters or delimiters (`':'` or `'`}), they should be escaped.

Note that they probably must also be escaped as the value for the `text` option in the filter argument string and as the filter argument in the filtergraph description, and possibly also for the shell, that makes up to four levels of escaping; using a text file avoids these problems.

The following functions are available:

`expr, e`

The expression evaluation result.

It must take one argument specifying the expression to be evaluated, which accepts the same constants and functions as the *x* and *y* values. Note that not all constants should be used, for example the text size is not known when evaluating the expression, so the constants *text\_w* and *text\_h* will have an undefined value.

`expr_int_format, eif`

Evaluate the expression's value and output as formatted integer.

The first argument is the expression to be evaluated, just as for the *expr* function. The second argument specifies the output format. Allowed values are 'x', 'X', 'd' and 'u'. They are treated exactly as in the `printf` function. The third parameter is optional and sets the number of positions taken by the output. It can be used to add padding with zeros from the left.

`gmtime`

The time at which the filter is running, expressed in UTC. It can accept an argument: a `strftime()` format string.

`localtime`

The time at which the filter is running, expressed in the local time zone. It can accept an argument: a `strftime()` format string.

`metadata`

Frame metadata. It must take one argument specifying metadata key.

`n, frame_num`

The frame number, starting from 0.

`pict_type`

A 1 character description of the current picture type.

`pts`

The timestamp of the current frame. It can take up to two arguments.

The first argument is the format of the timestamp; it defaults to `flt` for seconds as a decimal number with microsecond accuracy; `hms` stands for a formatted `[-]HH:MM:SS.mmm` timestamp with millisecond accuracy.



The second argument is an offset added to the timestamp.

### 37.32.3 Examples# TOC

- Draw "Test Text" with font FreeSerif, using the default values for the optional parameters.

```
drawtext="fontfile=/usr/share/fonts/truetype/freefont/FreeSerif.ttf: text='Test Text' "
```

- Draw 'Test Text' with font FreeSerif of size 24 at position x=100 and y=50 (counting from the top-left corner of the screen), text is yellow with a red box around it. Both the text and the box have an opacity of 20%.

```
drawtext="fontfile=/usr/share/fonts/truetype/freefont/FreeSerif.ttf: text='Test Text':\
x=100: y=50: fontsize=24: fontcolor=yellow@0.2: box=1: boxcolor=red@0.2"
```

Note that the double quotes are not necessary if spaces are not used within the parameter list.

- Show the text at the center of the video frame:

```
drawtext="fontsize=30:fontfile=FreeSerif.ttf:text='hello world':x=(w-text_w)/2:y=(h-text_h-line_h)/2"
```

- Show a text line sliding from right to left in the last row of the video frame. The file LONG\_LINE is assumed to contain a single line with no newlines.

```
drawtext="fontsize=15:fontfile=FreeSerif.ttf:text=LONG_LINE:y=h-line_h:x=-50*t"
```

- Show the content of file CREDITS off the bottom of the frame and scroll up.

```
drawtext="fontsize=20:fontfile=FreeSerif.ttf:textfile=CREDITS:y=h-20*t"
```

- Draw a single green letter "g", at the center of the input video. The glyph baseline is placed at half screen height.

```
drawtext="fontsize=60:fontfile=FreeSerif.ttf:fontcolor=green:text=g:x=(w-max_glyph_w)/2:y=h/2-ascent"
```

- Show text for 1 second every 3 seconds:

```
drawtext="fontfile=FreeSerif.ttf:fontcolor=white:x=100:y=x/dar:enable=1t(mod(t\,3)\,1):text='blink' "
```

- Use fontconfig to set the font. Note that the colons need to be escaped.

```
drawtext='fontfile=Linux Libertine O-40\:style=Semibold:text=FFmpeg'
```

- Print the date of a real-time encoding (see strftime(3)):

```
drawtext='fontfile=FreeSans.ttf:text=%{localtime\:%a %b %d %Y}'
```

- Show text fading in and out (appearing/disappearing):

```
#!/bin/sh
SD=1.0 # display start
DE=10.0 # display end
FD=1.5 # fade in duration
FD=1 # fade out duration
ffplay -f lavdi "color,drawtext=test?test?fontsize=50:fontfile=FreeSerif.ttf:fontcolor=gray:ff0000[if\\:\\ clip(255*(1-between(t\\, SDG + SPD\\, DDE - SPD\\) * ((t - SDG)/SPD2)*between(t\\, SDG\\, DDE + SPD\\) * ((t - DDE)/SPD2)*between(t\\, DDE\\, DDE - SPD\\) * ((t - DDE)/SPD2)*between(t\\, DDE - SPD\\, DDE) /\\, 0\\, 255) \\:\\ a\\:\\ 2 ]"
```

For more information about libfreetype, check: <http://www.freetype.org/>.

For more information about fontconfig, check:  
<http://freedesktop.org/software/fontconfig/fontconfig-user.html>.

For more information about libfribidi, check: <http://fribidi.org/>.

## 37.33 edgedetect# TOC

Detect and draw edges. The filter uses the Canny Edge Detection algorithm.

The filter accepts the following options:

`low`  
`high`

Set low and high threshold values used by the Canny thresholding algorithm.

The high threshold selects the "strong" edge pixels, which are then connected through 8-connectivity with the "weak" edge pixels selected by the low threshold.

*low* and *high* threshold values must be chosen in the range [0,1], and *low* should be lesser or equal to *high*.

Default value for *low* is 20 / 255, and default value for *high* is 50 / 255.

`mode`

Define the drawing mode.

`'wires'`

Draw white/gray wires on black background.

`'colormix'`

Mix the colors to create a paint/cartoon effect.

Default value is *wires*.

### 37.33.1 Examples# TOC

- Standard edge detection with custom values for the hysteresis thresholding:

```
edgedetect=low=0.1:high=0.4
```

- Painting effect without thresholding:

edgedetect=mode=colormix:high=0

## 37.34 eq# TOC

Set brightness, contrast, saturation and approximate gamma adjustment.

The filter accepts the following options:

contrast

Set the contrast expression. The value must be a float value in range  $-2.0$  to  $2.0$ . The default value is "0".

brightness

Set the brightness expression. The value must be a float value in range  $-1.0$  to  $1.0$ . The default value is "0".

saturation

Set the saturation expression. The value must be a float in range  $0.0$  to  $3.0$ . The default value is "1".

gamma

Set the gamma expression. The value must be a float in range  $0.1$  to  $10.0$ . The default value is "1".

gamma\_r

Set the gamma expression for red. The value must be a float in range  $0.1$  to  $10.0$ . The default value is "1".

gamma\_g

Set the gamma expression for green. The value must be a float in range  $0.1$  to  $10.0$ . The default value is "1".

gamma\_b

Set the gamma expression for blue. The value must be a float in range  $0.1$  to  $10.0$ . The default value is "1".

gamma\_weight

Set the gamma weight expression. It can be used to reduce the effect of a high gamma value on bright image areas, e.g. keep them from getting overamplified and just plain white. The value must be a float in range  $0.0$  to  $1.0$ . A value of  $0.0$  turns the gamma correction all the way down while  $1.0$  leaves it at its full strength. Default is "1".

eval

Set when the expressions for brightness, contrast, saturation and gamma expressions are evaluated.

It accepts the following values:

`'init'`

only evaluate expressions once during the filter initialization or when a command is processed

`'frame'`

evaluate expressions for each incoming frame

Default value is `'init'`.

The expressions accept the following parameters:

n

frame count of the input frame starting from 0

pos

byte position of the corresponding packet in the input file, NAN if unspecified

r

frame rate of the input video, NAN if the input frame rate is unknown

t

timestamp expressed in seconds, NAN if the input timestamp is unknown

### **37.34.1 Commands# TOC**

The filter supports the following commands:

contrast

Set the contrast expression.

brightness

Set the brightness expression.

saturation

Set the saturation expression.

gamma

Set the gamma expression.

gamma\_r

Set the gamma\_r expression.

gamma\_g

Set gamma\_g expression.

gamma\_b

Set gamma\_b expression.

gamma\_weight

Set gamma\_weight expression.

The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

## **37.35 erosion# TOC**

Apply erosion effect to the video.

This filter replaces the pixel by the local(3x3) minimum.

It accepts the following options:

threshold0

threshold1

threshold2

threshold3

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

coordinates

Flag which specifies the pixel to refer to. Default is 255 i.e. all eight pixels are used.

Flags to local 3x3 coordinates maps like this:

1 2 3 4 5 6 7 8

## 37.36 extractplanes# TOC

Extract color channel components from input video stream into separate grayscale video streams.

The filter accepts the following option:

`planes`

Set plane(s) to extract.

Available values for planes are:

`'y'`  
`'u'`  
`'v'`  
`'a'`  
`'r'`  
`'g'`  
`'b'`

Choosing planes not available in the input will result in an error. That means you cannot select `r`, `g`, `b` planes with `y`, `u`, `v` planes at same time.

### 37.36.1 Examples# TOC

- Extract luma, u and v color channel component from input video frame into 3 grayscale outputs:

```
ffmpeg -i video.avi -filter_complex 'extractplanes=y+u+v[y][u][v]' -map '[y]' y.avi -map '[u]' u.avi -map '[v]' v.avi
```

## 37.37 elbg# TOC

Apply a posterize effect using the ELBG (Enhanced LBG) algorithm.

For each input image, the filter will compute the optimal mapping from the input to the output given the codebook length, that is the number of distinct output colors.

This filter accepts the following options.

`codebook_length, l`

Set codebook length. The value must be a positive integer, and represents the number of distinct output colors. Default value is 256.

`nb_steps, n`

Set the maximum number of iterations to apply for computing the optimal mapping. The higher the value the better the result and the higher the computation time. Default value is 1.

`seed, s`

Set a random seed, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to -1, the filter will try to use a good random seed on a best effort basis.

`pal8`

Set pal8 output pixel format. This option does not work with codebook length greater than 256.

## 37.38 fade# TOC

Apply a fade-in/out effect to the input video.

It accepts the following parameters:

`type, t`

The effect type can be either "in" for a fade-in, or "out" for a fade-out effect. Default is in.

`start_frame, s`

Specify the number of the frame to start applying the fade effect at. Default is 0.

`nb_frames, n`

The number of frames that the fade effect lasts. At the end of the fade-in effect, the output video will have the same intensity as the input video. At the end of the fade-out transition, the output video will be filled with the selected `color`. Default is 25.

`alpha`

If set to 1, fade only alpha channel, if one exists on the input. Default value is 0.

`start_time, st`

Specify the timestamp (in seconds) of the frame to start to apply the fade effect. If both `start_frame` and `start_time` are specified, the fade will start at whichever comes last. Default is 0.

`duration, d`

The number of seconds for which the fade effect has to last. At the end of the fade-in effect the output video will have the same intensity as the input video, at the end of the fade-out transition the output video will be filled with the selected `color`. If both `duration` and `nb_frames` are specified, `duration` is used. Default is 0 (`nb_frames` is used by default).

`color, c`

Specify the color of the fade. Default is "black".

### 37.38.1 Examples# TOC

- Fade in the first 30 frames of video:

```
fade=in:0:30
```

The command above is equivalent to:

```
fade=t=in:s=0:n=30
```

- Fade out the last 45 frames of a 200-frame video:

```
fade=out:155:45  
fade=type=out:start_frame=155:nb_frames=45
```

- Fade in the first 25 frames and fade out the last 25 frames of a 1000-frame video:

```
fade=in:0:25, fade=out:975:25
```

- Make the first 5 frames yellow, then fade in from frame 5-24:

```
fade=in:5:20:color=yellow
```

- Fade in alpha over first 25 frames of video:

```
fade=in:0:25:alpha=1
```

- Make the first 5.5 seconds black, then fade in for 0.5 seconds:

```
fade=t=in:st=5.5:d=0.5
```

### 37.39 fftfilt# TOC

Apply arbitrary expressions to samples in frequency domain

`dc_Y`

Adjust the dc value (gain) of the luma plane of the image. The filter accepts an integer value in range 0 to 1000. The default value is set to 0.

`dc_U`

Adjust the dc value (gain) of the 1st chroma plane of the image. The filter accepts an integer value in range 0 to 1000. The default value is set to 0.

`dc_V`



Adjust the dc value (gain) of the 2nd chroma plane of the image. The filter accepts an integer value in range 0 to 1000. The default value is set to 0.

`weight_Y`

Set the frequency domain weight expression for the luma plane.

`weight_U`

Set the frequency domain weight expression for the 1st chroma plane.

`weight_V`

Set the frequency domain weight expression for the 2nd chroma plane.

The filter accepts the following variables:

`X`

`Y`

The coordinates of the current sample.

`W`

`H`

The width and height of the image.

### 37.39.1 Examples# TOC

- High-pass:

```
fftfilt=dc_Y=128:weight_Y='squish(1-(Y+X)/100)'
```

- Low-pass:

```
fftfilt=dc_Y=0:weight_Y='squish((Y+X)/100-1)'
```

- Sharpen:

```
fftfilt=dc_Y=0:weight_Y='1+squish(1-(Y+X)/100)'
```

### 37.40 field# TOC

Extract a single field from an interlaced image using stride arithmetic to avoid wasting CPU time. The output frames are marked as non-interlaced.

The filter accepts the following options:

type

Specify whether to extract the top (if the value is 0 or `top`) or the bottom field (if the value is 1 or `bottom`).

## 37.41 fieldmatch# TOC

Field matching filter for inverse telecine. It is meant to reconstruct the progressive frames from a telecined stream. The filter does not drop duplicated frames, so to achieve a complete inverse telecine `fieldmatch` needs to be followed by a decimation filter such as `decimate` in the filtergraph.

The separation of the field matching and the decimation is notably motivated by the possibility of inserting a de-interlacing filter fallback between the two. If the source has mixed telecined and real interlaced content, `fieldmatch` will not be able to match fields for the interlaced parts. But these remaining combed frames will be marked as interlaced, and thus can be de-interlaced by a later filter such as `yadif` before decimation.

In addition to the various configuration options, `fieldmatch` can take an optional second stream, activated through the `ppsrc` option. If enabled, the frames reconstruction will be based on the fields and frames from this second stream. This allows the first input to be pre-processed in order to help the various algorithms of the filter, while keeping the output lossless (assuming the fields are matched properly). Typically, a field-aware denoiser, or brightness/contrast adjustments can help.

Note that this filter uses the same algorithms as TIVTC/TFM (AviSynth project) and VIVTC/VFM (VapourSynth project). The later is a light clone of TFM from which `fieldmatch` is based on. While the semantic and usage are very close, some behaviour and options names can differ.

The `decimate` filter currently only works for constant frame rate input. If your input has mixed telecined (30fps) and progressive content with a lower framerate like 24fps use the following filterchain to produce the necessary cfr stream: `dejudder, fps=30000/1001, fieldmatch, decimate`.

The filter accepts the following options:

order

Specify the assumed field order of the input stream. Available values are:

`'auto'`

Auto detect parity (use FFmpeg's internal parity value).

`'bff'`

Assume bottom field first.

`'tff'`

Assume top field first.

Note that it is sometimes recommended not to trust the parity announced by the stream.

Default value is *auto*.

mode

Set the matching mode or strategy to use. `pc` mode is the safest in the sense that it won't risk creating jerkiness due to duplicate frames when possible, but if there are bad edits or blended fields it will end up outputting combed frames when a good match might actually exist. On the other hand, `pcn_ub` mode is the most risky in terms of creating jerkiness, but will almost always find a good frame if there is one. The other values are all somewhere in between `pc` and `pcn_ub` in terms of risking jerkiness and creating duplicate frames versus finding good matches in sections with bad edits, orphaned fields, blended fields, etc.

More details about `p/c/n/u/b` are available in `p/c/n/u/b` meaning section.

Available values are:

`'pc'`

2-way matching (`p/c`)

`'pc_n'`

2-way matching, and trying 3rd match if still combed (`p/c + n`)

`'pc_u'`

2-way matching, and trying 3rd match (same order) if still combed (`p/c + u`)

`'pc_n_ub'`

2-way matching, trying 3rd match if still combed, and trying 4th/5th matches if still combed (`p/c + n + u/b`)

`'pcn'`

3-way matching (`p/c/n`)

`'pcn_ub'`

3-way matching, and trying 4th/5th matches if all 3 of the original matches are detected as combed (`p/c/n + u/b`)

The parenthesis at the end indicate the matches that would be used for that mode assuming `order=tff` (and `field` on *auto* or *top*).

In terms of speed `pc` mode is by far the fastest and `pcn_ub` is the slowest.

Default value is `pc_n`.

#### `ppsrc`

Mark the main input stream as a pre-processed input, and enable the secondary input stream as the clean source to pick the fields from. See the filter introduction for more details. It is similar to the `clip2` feature from VFM/TFM.

Default value is 0 (disabled).

#### `field`

Set the field to match from. It is recommended to set this to the same value as `order` unless you experience matching failures with that setting. In certain circumstances changing the field that is used to match from can have a large impact on matching performance. Available values are:

`'auto'`

Automatic (same value as `order`).

`'bottom'`

Match from the bottom field.

`'top'`

Match from the top field.

Default value is *auto*.

#### `mchroma`

Set whether or not chroma is included during the match comparisons. In most cases it is recommended to leave this enabled. You should set this to 0 only if your clip has bad chroma problems such as heavy rainbowing or other artifacts. Setting this to 0 could also be used to speed things up at the cost of some accuracy.

Default value is 1.

#### `y0`

#### `y1`

These define an exclusion band which excludes the lines between `y0` and `y1` from being included in the field matching decision. An exclusion band can be used to ignore subtitles, a logo, or other things that may interfere with the matching. `y0` sets the starting scan line and `y1` sets the ending line; all lines in between `y0` and `y1` (including `y0` and `y1`) will be ignored. Setting `y0` and `y1` to the same value will disable the feature. `y0` and `y1` defaults to 0.

## scthresh

Set the scene change detection threshold as a percentage of maximum change on the luma plane. Good values are in the [ 8.0 , 14.0 ] range. Scene change detection is only relevant in case `combatch=sc`. The range for `scthresh` is [ 0.0 , 100.0 ].

Default value is 12.0.

## combatch

When `combatch` is not *none*, `fieldmatch` will take into account the combed scores of matches when deciding what match to use as the final match. Available values are:

‘none’

No final matching based on combed scores.

‘sc’

Combed scores are only used when a scene change is detected.

‘full’

Use combed scores all the time.

Default is *sc*.

## combdbg

Force `fieldmatch` to calculate the combed metrics for certain matches and print them. This setting is known as `micout` in TFM/VFM vocabulary. Available values are:

‘none’

No forced calculation.

‘pcn’

Force p/c/n calculations.

‘pcnub’

Force p/c/n/u/b calculations.

Default value is *none*.

## cthresh

This is the area combing threshold used for combed frame detection. This essentially controls how "strong" or "visible" combing must be to be detected. Larger values mean combing must be more visible and smaller values mean combing can be less visible or strong and still be detected. Valid settings are from -1 (every pixel will be detected as combed) to 255 (no pixel will be detected as combed). This is basically a pixel difference value. A good range is [ 8 , 12 ].

Default value is 9.

chroma

Sets whether or not chroma is considered in the combed frame decision. Only disable this if your source has chroma problems (rainbowing, etc.) that are causing problems for the combed frame detection with chroma enabled. Actually, using `chroma=0` is usually more reliable, except for the case where there is chroma only combing in the source.

Default value is 0.

blockx

blocky

Respectively set the x-axis and y-axis size of the window used during combed frame detection. This has to do with the size of the area in which `combpel` pixels are required to be detected as combed for a frame to be declared combed. See the `combpel` parameter description for more info. Possible values are any number that is a power of 2 starting at 4 and going up to 512.

Default value is 16.

combpel

The number of combed pixels inside any of the `blocky` by `blockx` size blocks on the frame for the frame to be detected as combed. While `cthresh` controls how "visible" the combing must be, this setting controls "how much" combing there must be in any localized area (a window defined by the `blockx` and `blocky` settings) on the frame. Minimum value is 0 and maximum is `blocky * blockx` (at which point no frames will ever be detected as combed). This setting is known as MI in TFM/VFM vocabulary.

Default value is 80.

### 37.41.1 p/c/n/u/b meaning# TOC

#### 37.41.1.1 p/c/n# TOC

We assume the following telecined stream:

```
Top fields:      1 2 2 3 4
Bottom fields:  1 2 3 4 4
```

The numbers correspond to the progressive frame the fields relate to. Here, the first two frames are progressive, the 3rd and 4th are combed, and so on.

When `fieldmatch` is configured to run a matching from bottom (`field=bottom`) this is how this input stream get transformed:

```
Input stream:
      T      1 2 2 3 4
      B      1 2 3 4 4  <-- matching reference

Matches:
      c c n n c

Output stream:
      T      1 2 3 4 4
      B      1 2 3 4 4
```

As a result of the field matching, we can see that some frames get duplicated. To perform a complete inverse telecine, you need to rely on a decimation filter after this operation. See for instance the decimate filter.

The same operation now matching from top fields (`field=top`) looks like this:

```
Input stream:
      T      1 2 2 3 4  <-- matching reference
      B      1 2 3 4 4

Matches:
      c c p p c

Output stream:
      T      1 2 2 3 4
      B      1 2 2 3 4
```

In these examples, we can see what *p*, *c* and *n* mean; basically, they refer to the frame and field of the opposite parity:

- *p* matches the field of the opposite parity in the previous frame
- *c* matches the field of the opposite parity in the current frame
- *n* matches the field of the opposite parity in the next frame

### 37.41.1.2 u/b# TOC

The *u* and *b* matching are a bit special in the sense that they match from the opposite parity flag. In the following examples, we assume that we are currently matching the 2nd frame (Top:2, bottom:2). According to the match, a 'x' is placed above and below each matched fields.

With bottom matching (`field=bottom`):

Match:	c	p	n	b	u
	x	x	x	x	x
Top	1 2 2	1 2 2	1 2 2	1 2 2	1 2 2
Bottom	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
	x	x	x	x	x

Output frames:	2	1	2	2	2
	2	2	2	1	3

With top matching (`field=top`):

Match:	c	p	n	b	u
	x	x	x	x	x
Top	1 2 2	1 2 2	1 2 2	1 2 2	1 2 2
Bottom	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
	x	x	x	x	x

Output frames:	2	2	2	1	2
	2	1	3	2	2

## 37.41.2 Examples# TOC

Simple IVTC of a top field first telecined stream:

```
fieldmatch=order=tff:combmatch=none, decimate
```

Advanced IVTC, with fallback on yadif for still combed frames:

```
fieldmatch=order=tff:combmatch=full, yadif=deint=interlaced, decimate
```

## 37.42 fieldorder# TOC

Transform the field order of the input video.

It accepts the following parameters:

`order`

The output field order. Valid values are *tff* for top field first or *bff* for bottom field first.

The default value is 'tff'.

The transformation is done by shifting the picture content up or down by one line, and filling the remaining line with appropriate picture content. This method is consistent with most broadcast field order converters.



If the input video is not flagged as being interlaced, or it is already flagged as being of the required output field order, then this filter does not alter the incoming video.

It is very useful when converting to or from PAL DV material, which is bottom field first.

For example:

```
ffmpeg -i in.vob -vf "fieldorder=bff" out.dv
```

## 37.43 fifo# TOC

Buffer input images and send them when they are requested.

It is mainly useful when auto-inserted by the libavfilter framework.

It does not take parameters.

## 37.44 find\_rect# TOC

Find a rectangular object

It accepts the following options:

`object`

Filepath of the object image, needs to be in gray8.

`threshold`

Detection threshold, default is 0.5.

`mipmaps`

Number of mipmaps, default is 3.

`xmin, ymin, xmax, ymax`

Specifies the rectangle in which to search.

### 37.44.1 Examples# TOC

- Generate a representative palette of a given video using `ffmpeg`:

```
ffmpeg -i file.ts -vf find_rect=newref.pgm,cover_rect=cover.jpg:mode=cover new.mkv
```

## 37.45 cover\_rect# TOC

Cover a rectangular object

It accepts the following options:

`cover`

Filepath of the optional cover image, needs to be in yuv420.

`mode`

Set covering mode.

It accepts the following values:

`'cover'`

cover it by the supplied image

`'blur'`

cover it by interpolating the surrounding pixels

Default value is *blur*.

### 37.45.1 Examples# TOC

- Generate a representative palette of a given video using `ffmpeg`:

```
ffmpeg -i file.ts -vf find_rect=newref.pgm,cover_rect=cover.jpg:mode=cover new.mkv
```

## 37.46 format# TOC

Convert the input video to one of the specified pixel formats. Libavfilter will try to pick one that is suitable as input to the next filter.

It accepts the following parameters:

`pix_fmts`

A `'|'`-separated list of pixel format names, such as `"pix_fmts=yuv420p|monow|rgb24"`.

### 37.46.1 Examples# TOC

- Convert the input video to the *yuv420p* format

```
format=pix_fmts=yuv420p
```

Convert the input video to any of the formats in the list

```
format=pix_fmts=yuv420p|yuv444p|yuv410p
```

## 37.47 fps# TOC

Convert the video to specified constant frame rate by duplicating or dropping frames as necessary.

It accepts the following parameters:

`fps`

The desired output frame rate. The default is 25.

`round`

Rounding method.

Possible values are:

`zero`

zero round towards 0

`inf`

round away from 0

`down`

round towards -infinity

`up`

round towards +infinity

`near`

round to nearest

The default is `near`.

`start_time`

Assume the first PTS should be the given value, in seconds. This allows for padding/trimming at the start of stream. By default, no assumption is made about the first frame's expected PTS, so no padding or trimming is done. For example, this could be set to 0 to pad the beginning with duplicates of the first frame if a video stream starts after the audio stream or to trim any frames with a negative

PTS.

Alternatively, the options can be specified as a flat string: *fps[:round]*.

See also the `setpts` filter.

### 37.47.1 Examples# TOC

- A typical usage in order to set the fps to 25:

```
fps=fps=25
```

- Sets the fps to 24, using abbreviation and rounding method to round to nearest:

```
fps=fps=film:round=near
```

## 37.48 framepack# TOC

Pack two different video streams into a stereoscopic video, setting proper metadata on supported codecs. The two views should have the same size and framerate and processing will stop when the shorter video ends. Please note that you may conveniently adjust view properties with the `scale` and `fps` filters.

It accepts the following parameters:

`format`

The desired packing format. Supported values are:

`sbs`

The views are next to each other (default).

`tab`

The views are on top of each other.

`lines`

The views are packed by line.

`columns`

The views are packed by column.

`frameseq`

The views are temporally interleaved.

Some examples:

```
# Convert left and right views into a frame-sequential video
ffmpeg -i LEFT -i RIGHT -filter_complex framepack=frameseq OUTPUT

# Convert views into a side-by-side video with the same output resolution as the input
ffmpeg -i LEFT -i RIGHT -filter_complex [0:v]scale=wiw/2[left],[1:v]scale=wiw/2[right],[left][right]framepack=sbs OUTPUT
```

## 37.49 framerate# TOC

Change the frame rate by interpolating new video output frames from the source frames.

This filter is not designed to function correctly with interlaced media. If you wish to change the frame rate of interlaced media then you are required to deinterlace before this filter and re-interlace after this filter.

A description of the accepted options follows.

fps

Specify the output frames per second. This option can also be specified as a value alone. The default is 50.

interp\_start

Specify the start of a range where the output frame will be created as a linear interpolation of two frames. The range is [0-255], the default is 15.

interp\_end

Specify the end of a range where the output frame will be created as a linear interpolation of two frames. The range is [0-255], the default is 240.

scene

Specify the level at which a scene change is detected as a value between 0 and 100 to indicate a new scene; a low value reflects a low probability for the current frame to introduce a new scene, while a higher value means the current frame is more likely to be one. The default is 7.

flags

Specify flags influencing the filter process.

Available value for *flags* is:

scene\_change\_detect, scd

Enable scene change detection using the value of the option *scene*. This flag is enabled by default.

## 37.50 framestep# TOC

Select one frame every N-th frame.

This filter accepts the following option:

`step`

Select frame after every `step` frames. Allowed values are positive integers higher than 0. Default value is 1.

## 37.51 frei0r# TOC

Apply a frei0r effect to the input video.

To enable the compilation of this filter, you need to install the frei0r header and configure FFmpeg with `--enable-frei0r`.

It accepts the following parameters:

`filter_name`

The name of the frei0r effect to load. If the environment variable `FREI0R_PATH` is defined, the frei0r effect is searched for in each of the directories specified by the colon-separated list in `FREI0R_PATH`. Otherwise, the standard frei0r paths are searched, in this order:  
`HOME/.frei0r-1/lib/`, `/usr/local/lib/frei0r-1/`, `/usr/lib/frei0r-1/`.

`filter_params`

A `'|'`-separated list of parameters to pass to the frei0r effect.

A frei0r effect parameter can be a boolean (its value is either "y" or "n"), a double, a color (specified as *R/G/B*, where *R*, *G*, and *B* are floating point numbers between 0.0 and 1.0, inclusive) or by a color description specified in the "Color" section in the ffmpeg-utils manual), a position (specified as *X/Y*, where *X* and *Y* are floating point numbers) and/or a string.

The number and types of parameters depend on the loaded effect. If an effect parameter is not specified, the default value is set.

### 37.51.1 Examples# TOC

- Apply the distort0r effect, setting the first two double parameters:

```
frei0r=filter_name=distort0r:filter_params=0.5|0.01
```

- Apply the colordistance effect, taking a color as the first parameter:

```
frei0r=colordistance:0.2/0.3/0.4
frei0r=colordistance:violet
frei0r=colordistance:0x112233
```

- Apply the perspective effect, specifying the top left and top right image positions:

```
frei0r=perspective:0.2/0.2|0.8/0.2
```

For more information, see <http://frei0r.dyne.org>

## 37.52 fspp# TOC

Apply fast and simple postprocessing. It is a faster version of spp.

It splits (I)DCT into horizontal/vertical passes. Unlike the simple post- processing filter, one of them is performed once per block, not per pixel. This allows for much higher speed.

The filter accepts the following options:

quality

Set quality. This option defines the number of levels for averaging. It accepts an integer in the range 4-5. Default value is 4.

qp

Force a constant quantization parameter. It accepts an integer in range 0-63. If not set, the filter will use the QP from the video stream (if available).

strength

Set filter strength. It accepts an integer in range -15 to 32. Lower values mean more details but also more artifacts, while higher values make the image smoother but also blurrier. Default value is 0 â PSNR optimal.

use\_bframe\_qp

Enable the use of the QP from the B-Frames if set to 1. Using this option may cause flicker since the B-Frames have often larger QP. Default is 0 (not enabled).

## 37.53 geq# TOC

The filter accepts the following options:

lum\_expr, lum

Set the luminance expression.

`cb_expr, cb`

Set the chrominance blue expression.

`cr_expr, cr`

Set the chrominance red expression.

`alpha_expr, a`

Set the alpha expression.

`red_expr, r`

Set the red expression.

`green_expr, g`

Set the green expression.

`blue_expr, b`

Set the blue expression.

The colorspace is selected according to the specified options. If one of the `lum_expr`, `cb_expr`, or `cr_expr` options is specified, the filter will automatically select a YCbCr colorspace. If one of the `red_expr`, `green_expr`, or `blue_expr` options is specified, it will select an RGB colorspace.

If one of the chrominance expression is not defined, it falls back on the other one. If no alpha expression is specified it will evaluate to opaque value. If none of chrominance expressions are specified, they will evaluate to the luminance expression.

The expressions can use the following variables and functions:

`N`

The sequential number of the filtered frame, starting from 0.

`X`

`Y`

The coordinates of the current sample.

`W`

`H`

The width and height of the image.



SW  
SH

Width and height scale depending on the currently filtered plane. It is the ratio between the corresponding luma plane number of pixels and the current plane ones. E.g. for YUV4:2:0 the values are 1, 1 for the luma plane, and 0.5, 0.5 for chroma planes.

T

Time of the current frame, expressed in seconds.

$p(x, y)$

Return the value of the pixel at location (x,y) of the current plane.

$lum(x, y)$

Return the value of the pixel at location (x,y) of the luminance plane.

$cb(x, y)$

Return the value of the pixel at location (x,y) of the blue-difference chroma plane. Return 0 if there is no such plane.

$cr(x, y)$

Return the value of the pixel at location (x,y) of the red-difference chroma plane. Return 0 if there is no such plane.

$r(x, y)$

$g(x, y)$

$b(x, y)$

Return the value of the pixel at location (x,y) of the red/green/blue component. Return 0 if there is no such component.

$alpha(x, y)$

Return the value of the pixel at location (x,y) of the alpha plane. Return 0 if there is no such plane.

For functions, if  $x$  and  $y$  are outside the area, the value will be automatically clipped to the closer edge.

### 37.53.1 Examples# TOC

- Flip the image horizontally:

$geq=p(W-X\backslash, Y)$

- Generate a bidimensional sine wave, with angle  $\pi/3$  and a wavelength of 100 pixels:

```
geq=128 + 100*sin(2*(PI/100)*(cos(PI/3)*(X-50*T) + sin(PI/3)*Y)):128:128
```

- Generate a fancy enigmatic moving light:

```
nullsrc=s=256x256,geq=random(1)/hypot(X-cos(N*0.07)*W/2-W/2,Y-sin(N*0.09)*H/2-H/2)^2*1000000*sin(N*0.02):128:128
```

- Generate a quick emboss effect:

```
format=gray,geq=lum_expr='(p(X,Y)+(256-p(X-4,Y-4)))/2'
```

- Modify RGB components depending on pixel position:

```
geq=r='X/W*r(X,Y)':g='(1-X/W)*g(X,Y)':b='(H-Y)/H*b(X,Y)'
```

- Create a radial gradient that is the same size as the input (also see the vignette filter):

```
geq=lum=255*gauss((X/W-0.5)*3)*gauss((Y/H-0.5)*3)/gauss(0)/gauss(0),format=gray
```

- Create a linear gradient to use as a mask for another filter, then compose with overlay. In this example the video will gradually become more blurry from the top to the bottom of the y-axis as defined by the linear gradient:

```
ffmpeg -i input.mp4 -filter_complex "geq=lum=255*(Y/H),format=gray[grad];[0:v]boxblur=4[blur];[blur][grad]alphaseg[alpha];[0:v][alpha]overlay" output.mp4
```

## 37.54 gradfun# TOC

Fix the banding artifacts that are sometimes introduced into nearly flat regions by truncation to 8bit color depth. Interpolate the gradients that should go where the bands are, and dither them.

It is designed for playback only. Do not use it prior to lossy compression, because compression tends to lose the dither and bring back the bands.

It accepts the following parameters:

**strength**

The maximum amount by which the filter will change any one pixel. This is also the threshold for detecting nearly flat regions. Acceptable values range from .51 to 64; the default value is 1.2. Out-of-range values will be clipped to the valid range.

**radius**

The neighborhood to fit the gradient to. A larger radius makes for smoother gradients, but also prevents the filter from modifying the pixels near detailed regions. Acceptable values are 8-32; the default value is 16. Out-of-range values will be clipped to the valid range.

Alternatively, the options can be specified as a flat string: *strength[:radius]*

### 37.54.1 Examples# TOC

- Apply the filter with a 3.5 strength and radius of 8:

```
gradfun=3.5:8
```

- Specify radius, omitting the strength (which will fall-back to the default value):

```
gradfun=radius=8
```

### 37.55 haldclut# TOC

Apply a Hald CLUT to a video stream.

First input is the video stream to process, and second one is the Hald CLUT. The Hald CLUT input can be a simple picture or a complete video stream.

The filter accepts the following options:

`shortest`

Force termination when the shortest input terminates. Default is 0.

`repeatlast`

Continue applying the last CLUT after the end of the stream. A value of 0 disable the filter after the last frame of the CLUT is reached. Default is 1.

`haldclut` also has the same interpolation options as `lut3d` (both filters share the same internals).

More information about the Hald CLUT can be found on Eskil Steenberg's website (Hald CLUT author) at <http://www.quelsolaar.com/technology/clut.html>.

### 37.55.1 Workflow examples# TOC

#### 37.55.1.1 Hald CLUT video stream# TOC

Generate an identity Hald CLUT stream altered with various effects:

```
ffmpeg -f lavfi -i haldclutsrc=8 -vf "hue=H=2*PI*t:s=sin(2*PI*t)+1, curves=cross_process" -t 10 -c:v ffv1 clut.nut
```

Note: make sure you use a lossless codec.

Then use it with `haldclut` to apply it on some random stream:

```
ffmpeg -f lavfi -i mandelbrot -i clut.nut -filter_complex '[0][1] haldclut' -t 20 mandelclut.mkv
```

The Hald CLUT will be applied to the 10 first seconds (duration of `clut.nut`), then the latest picture of that CLUT stream will be applied to the remaining frames of the `mandelbrot` stream.

### 37.55.1.2 Hald CLUT with preview# TOC

A Hald CLUT is supposed to be a squared image of `Level*Level*Level` by `Level*Level*Level` pixels. For a given Hald CLUT, FFmpeg will select the biggest possible square starting at the top left of the picture. The remaining padding pixels (bottom or right) will be ignored. This area can be used to add a preview of the Hald CLUT.

Typically, the following generated Hald CLUT will be supported by the `haldclut` filter:

```
ffmpeg -f lavfi -i haldclutsrc=8 -vf "  
    pad=iw+320 [padded_clut];  
    smptebars=s=320x256, split [a][b];  
    [padded_clut][a] overlay=W-320:h, curves=color_negative [main];  
    [main][b] overlay=W-320" -frames:v 1 clut.png
```

It contains the original and a preview of the effect of the CLUT: SMPTE color bars are displayed on the right-top, and below the same color bars processed by the color changes.

Then, the effect of this Hald CLUT can be visualized with:

```
ffplay input.mkv -vf "movie=clut.png, [in] haldclut"
```

### 37.56 hflip# TOC

Flip the input video horizontally.

For example, to horizontally flip the input video with `ffmpeg`:

```
ffmpeg -i in.avi -vf "hflip" out.avi
```

### 37.57 histeq# TOC

This filter applies a global color histogram equalization on a per-frame basis.

It can be used to correct video that has a compressed range of pixel intensities. The filter redistributes the pixel intensities to equalize their distribution across the intensity range. It may be viewed as an "automatically adjusting contrast filter". This filter is useful only for correcting degraded or poorly captured source video.

The filter accepts the following options:

`strength`

Determine the amount of equalization to be applied. As the strength is reduced, the distribution of pixel intensities more-and-more approaches that of the input frame. The value must be a float number in the range [0,1] and defaults to 0.200.

`intensity`

Set the maximum intensity that can generated and scale the output values appropriately. The strength should be set as desired and then the intensity can be limited if needed to avoid washing-out. The value must be a float number in the range [0,1] and defaults to 0.210.

`antibanding`

Set the antibanding level. If enabled the filter will randomly vary the luminance of output pixels by a small amount to avoid banding of the histogram. Possible values are `none`, `weak` or `strong`. It defaults to `none`.

## 37.58 histogram# TOC

Compute and draw a color distribution histogram for the input video.

The computed histogram is a representation of the color component distribution in an image.

The filter accepts the following options:

`mode`

Set histogram mode.

It accepts the following values:

`'levels'`

Standard histogram that displays the color components distribution in an image. Displays color graph for each color component. Shows distribution of the Y, U, V, A or R, G, B components, depending on input format, in the current frame. Below each graph a color component scale meter is shown.

`'color'`

Displays chroma values (U/V color placement) in a two dimensional graph (which is called a vectorscope). The brighter a pixel in the vectorscope, the more pixels of the input frame correspond to that pixel (i.e., more pixels have this chroma value). The V component is displayed on the horizontal (X) axis, with the leftmost side being  $V = 0$  and the rightmost side being  $V = 255$ . The U component is displayed on the vertical (Y) axis, with the top representing  $U = 0$  and the bottom representing  $U = 255$ .

The position of a white pixel in the graph corresponds to the chroma value of a pixel of the input clip. The graph can therefore be used to read the hue (color flavor) and the saturation (the dominance of the hue in the color). As the hue of a color changes, it moves around the square. At the center of the square the saturation is zero, which means that the corresponding pixel has no color. If the amount of a specific color is increased (while leaving the other colors unchanged) the saturation increases, and the indicator moves towards the edge of the square.

`'color2'`

Chroma values in vectorscope, similar as `color` but actual chroma values are displayed.

`'waveform'`

Per row/column color component graph. In row mode, the graph on the left side represents color component value 0 and the right side represents value = 255. In column mode, the top side represents color component value = 0 and bottom side represents value = 255.

Default value is `levels`.

`level_height`

Set height of level in `levels`. Default value is 200. Allowed range is [50, 2048].

`scale_height`

Set height of color scale in `levels`. Default value is 12. Allowed range is [0, 40].

`step`

Set step for waveform mode. Smaller values are useful to find out how many values of the same luminance are distributed across input rows/columns. Default value is 10. Allowed range is [1, 255].

`waveform_mode`

Set mode for waveform. Can be either `row`, or `column`. Default is `row`.

`waveform_mirror`

Set mirroring mode for waveform. 0 means unmirrored, 1 means mirrored. In mirrored mode, higher values will be represented on the left side for `row` mode and at the top for `column` mode. Default is 0 (unmirrored).

`display_mode`

Set display mode for waveform and `levels`. It accepts the following values:

`'parade'`

Display separate graph for the color components side by side in `row` waveform mode or one below the other in `column` waveform mode for waveform histogram mode. For `levels` histogram mode, per color component graphs are placed below each other.

Using this display mode in waveform histogram mode makes it easy to spot color casts in the highlights and shadows of an image, by comparing the contours of the top and the bottom graphs of each waveform. Since whites, grays, and blacks are characterized by exactly equal amounts of red, green, and blue, neutral areas of the picture should display three waveforms of roughly equal width/height. If not, the correction is easy to perform by making level adjustments the

three waveforms.

`'overlay'`

Presents information identical to that in the `parade`, except that the graphs representing color components are superimposed directly over one another.

This display mode in `waveform histogram` mode makes it easier to spot relative differences or similarities in overlapping areas of the color components that are supposed to be identical, such as neutral whites, grays, or blacks.

Default is `parade`.

`levels_mode`

Set mode for `levels`. Can be either `linear`, or `logarithmic`. Default is `linear`.

`components`

Set what color components to display for mode `levels`. Default is 7.

### 37.58.1 Examples# TOC

- Calculate and draw histogram:

```
ffplay -i input -vf histogram
```

### 37.59 hqdn3d# TOC

This is a high precision/quality 3d denoise filter. It aims to reduce image noise, producing smooth images and making still images really still. It should enhance compressibility.

It accepts the following optional parameters:

`luma_spatial`

A non-negative floating point number which specifies spatial luma strength. It defaults to 4.0.

`chroma_spatial`

A non-negative floating point number which specifies spatial chroma strength. It defaults to  $3.0 * \textit{luma\_spatial} / 4.0$ .

`luma_tmp`

A floating point number which specifies luma temporal strength. It defaults to  $6.0 * \textit{luma\_spatial} / 4.0$ .

`chroma_tmp`

A floating point number which specifies chroma temporal strength. It defaults to  $\text{luma\_tmp} * \text{chroma\_spatial} / \text{luma\_spatial}$ .

## 37.60 hqx# TOC

Apply a high-quality magnification filter designed for pixel art. This filter was originally created by Maxim Stepin.

It accepts the following option:

`n`

Set the scaling dimension: 2 for hq2x, 3 for hq3x and 4 for hq4x. Default is 3.

## 37.61 hstack# TOC

Stack input videos horizontally.

All streams must be of same pixel format and of same height.

Note that this filter is faster than using overlay and pad filter to create same output.

The filter accept the following option:

`nb_inputs`

Set number of input streams. Default is 2.

## 37.62 hue# TOC

Modify the hue and/or the saturation of the input.

It accepts the following parameters:

`h`

Specify the hue angle as a number of degrees. It accepts an expression, and defaults to "0".

`s`

Specify the saturation in the [-10,10] range. It accepts an expression and defaults to "1".

`H`

Specify the hue angle as a number of radians. It accepts an expression, and defaults to "0".

`b`



Specify the brightness in the [-10,10] range. It accepts an expression and defaults to "0".

`h` and `H` are mutually exclusive, and can't be specified at the same time.

The `b`, `h`, `H` and `s` option values are expressions containing the following constants:

`n`

frame count of the input frame starting from 0

`pts`

presentation timestamp of the input frame expressed in time base units

`r`

frame rate of the input video, NAN if the input frame rate is unknown

`t`

timestamp expressed in seconds, NAN if the input timestamp is unknown

`tb`

time base of the input video

### 37.62.1 Examples# TOC

- Set the hue to 90 degrees and the saturation to 1.0:

```
hue=h=90:s=1
```

- Same command but expressing the hue in radians:

```
hue=H=PI/2:s=1
```

- Rotate hue and make the saturation swing between 0 and 2 over a period of 1 second:

```
hue="H=2*PI*t: s=sin(2*PI*t)+1"
```

- Apply a 3 seconds saturation fade-in effect starting at 0:

```
hue="s=min(t/3\,1)"
```

The general fade-in expression can be written as:

```
hue="s=min(0\, max((t-START)/DURATION\, 1))"
```

- Apply a 3 seconds saturation fade-out effect starting at 5 seconds:

```
hue="s=max(0\, min(1\, (8-t)/3))"
```

The general fade-out expression can be written as:

```
hue="s=max(0\, min(1\, (START+DURATION-t)/DURATION))"
```

### 37.62.2 Commands# TOC

This filter supports the following commands:

b  
s  
h  
H

Modify the hue and/or the saturation and/or brightness of the input video. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.63 idet# TOC

Detect video interlacing type.

This filter tries to detect if the input frames are interlaced, progressive, top or bottom field first. It will also try and detect fields that are repeated between adjacent frames (a sign of telecine).

Single frame detection considers only immediately adjacent frames when classifying each frame. Multiple frame detection incorporates the classification history of previous frames.

The filter will log these metadata values:

`single.current_frame`

Detected type of current frame using single-frame detection. One of: “tff” (top field first), “bff” (bottom field first), “progressive”, or “undetermined”

`single.tff`

Cumulative number of frames detected as top field first using single-frame detection.

`multiple.tff`

Cumulative number of frames detected as top field first using multiple-frame detection.

`single.bff`

Cumulative number of frames detected as bottom field first using single-frame detection.

`multiple.current_frame`

Detected type of current frame using multiple-frame detection. One of: “tff” (top field first), “bff” (bottom field first), “progressive”, or “undetermined”

`multiple.bff`

Cumulative number of frames detected as bottom field first using multiple-frame detection.

`single.progressive`

Cumulative number of frames detected as progressive using single-frame detection.

`multiple.progressive`

Cumulative number of frames detected as progressive using multiple-frame detection.

`single.undetermined`

Cumulative number of frames that could not be classified using single-frame detection.

`multiple.undetermined`

Cumulative number of frames that could not be classified using multiple-frame detection.

`repeated.current_frame`

Which field in the current frame is repeated from the last. One of “neither”, “top”, or “bottom”.

`repeated.neither`

Cumulative number of frames with no repeated field.

`repeated.top`

Cumulative number of frames with the top field repeated from the previous frame’s top field.

`repeated.bottom`

Cumulative number of frames with the bottom field repeated from the previous frame’s bottom field.

The filter accepts the following options:

`intl_thres`

Set interlacing threshold.

`prog_thres`

Set progressive threshold.

`repeat_thres`

Threshold for repeated field detection.

`half_life`

Number of frames after which a given frame's contribution to the statistics is halved (i.e., it contributes only 0.5 to its classification). The default of 0 means that all frames seen are given full weight of 1.0 forever.

`analyze_interlaced_flag`

When this is not 0 then idet will use the specified number of frames to determine if the interlaced flag is accurate, it will not count undetermined frames. If the flag is found to be accurate it will be used without any further computations, if it is found to be inaccurate it will be cleared without any further computations. This allows inserting the idet filter as a low computational method to clean up the interlaced flag

## 37.64 il# TOC

Deinterleave or interleave fields.

This filter allows one to process interlaced images fields without deinterlacing them. Deinterleaving splits the input frame into 2 fields (so called half pictures). Odd lines are moved to the top half of the output image, even lines to the bottom half. You can process (filter) them independently and then re-interleave them.

The filter accepts the following options:

`luma_mode, l`

`chroma_mode, c`

`alpha_mode, a`

Available values for *luma\_mode*, *chroma\_mode* and *alpha\_mode* are:

'none'

Do nothing.

'deinterleave, d'

Deinterleave fields, placing one above the other.

`'interleave, i'`

Interleave fields. Reverse the effect of deinterleaving.

Default value is none.

`luma_swap, ls`  
`chroma_swap, cs`  
`alpha_swap, as`

Swap luma/chroma/alpha fields. Exchange even & odd lines. Default value is 0.

## 37.65 inflate# TOC

Apply inflate effect to the video.

This filter replaces the pixel by the local(3x3) average by taking into account only values higher than the pixel.

It accepts the following options:

`threshold0`  
`threshold1`  
`threshold2`  
`threshold3`

Limit the maximum change for each plane, default is 65535. If 0, plane will remain unchanged.

## 37.66 interlace# TOC

Simple interlacing filter from progressive contents. This interleaves upper (or lower) lines from odd frames with lower (or upper) lines from even frames, halving the frame rate and preserving image height.

Original Frame 'j'	Original Frame 'j+1'	New Frame (tff)
=====	=====	=====
Line 0 ----->		Frame 'j' Line 0
Line 1	Line 1 ---->	Frame 'j+1' Line 1
Line 2 ----->		Frame 'j' Line 2
Line 3	Line 3 ---->	Frame 'j+1' Line 3
...	...	...

New Frame + 1 will be generated by Frame 'j+2' and Frame 'j+3' and so on

It accepts the following optional parameters:

`scan`

This determines whether the interlaced frame is taken from the even (tff - default) or odd (bff) lines of the progressive frame.

lowpass

Enable (default) or disable the vertical lowpass filter to avoid twitter interlacing and reduce moire patterns.

## 37.67 kerndeint# TOC

Deinterlace input video by applying Donald Graft's adaptive kernel deinterling. Work on interlaced parts of a video to produce progressive frames.

The description of the accepted parameters follows.

thresh

Set the threshold which affects the filter's tolerance when determining if a pixel line must be processed. It must be an integer in the range [0,255] and defaults to 10. A value of 0 will result in applying the process on every pixels.

map

Paint pixels exceeding the threshold value to white if set to 1. Default is 0.

order

Set the fields order. Swap fields if set to 1, leave fields alone if 0. Default is 0.

sharp

Enable additional sharpening if set to 1. Default is 0.

twoway

Enable twoway sharpening if set to 1. Default is 0.

### 37.67.1 Examples# TOC

- Apply default values:

```
kerndeint=thresh=10:map=0:order=0:sharp=0:twoway=0
```

- Enable additional sharpening:

```
kerndeint=sharp=1
```

- Paint processed pixels in white:

```
kerndeint=map=1
```

## 37.68 lenscorrection# TOC

### Correct radial lens distortion

This filter can be used to correct for radial distortion as can result from the use of wide angle lenses, and thereby re-rectify the image. To find the right parameters one can use tools available for example as part of opencv or simply trial-and-error. To use opencv use the calibration sample (under samples/cpp) from the opencv sources and extract the k1 and k2 coefficients from the resulting matrix.

Note that effectively the same filter is available in the open-source tools Krita and Digikam from the KDE project.

In contrast to the vignette filter, which can also be used to compensate lens errors, this filter corrects the distortion of the image, whereas vignette corrects the brightness distribution, so you may want to use both filters together in certain cases, though you will have to take care of ordering, i.e. whether vignetting should be applied before or after lens correction.

### 37.68.1 Options# TOC

The filter accepts the following options:

**cx**

Relative x-coordinate of the focal point of the image, and thereby the center of the distortion. This value has a range [0,1] and is expressed as fractions of the image width.

**cy**

Relative y-coordinate of the focal point of the image, and thereby the center of the distortion. This value has a range [0,1] and is expressed as fractions of the image height.

**k1**

Coefficient of the quadratic correction term. 0.5 means no correction.

**k2**

Coefficient of the double quadratic correction term. 0.5 means no correction.

The formula that generates the correction is:

$$r\_src = r\_tgt * (1 + k1 * (r\_tgt / r\_0)^2 + k2 * (r\_tgt / r\_0)^4)$$

where  $r\_0$  is halve of the image diagonal and  $r\_src$  and  $r\_tgt$  are the distances from the focal point in the source and target images, respectively.

## 37.69 lut3d# TOC

Apply a 3D LUT to an input video.

The filter accepts the following options:

`file`

Set the 3D LUT file name.

Currently supported formats:

‘3dl’

AfterEffects

‘cube’

Iridas

‘dat’

DaVinci

‘m3d’

Pandora

`interp`

Select interpolation mode.

Available values are:

‘nearest’

Use values from the nearest defined point.

‘trilinear’

Interpolate values using the 8 points defining a cube.

‘tetrahedral’

Interpolate values using a tetrahedron.



## 37.70 lut, lutrgb, lutyuv# TOC

Compute a look-up table for binding each pixel component input value to an output value, and apply it to the input video.

*lutyuv* applies a lookup table to a YUV input video, *lutrgb* to an RGB input video.

These filters accept the following parameters:

*c0*

set first pixel component expression

*c1*

set second pixel component expression

*c2*

set third pixel component expression

*c3*

set fourth pixel component expression, corresponds to the alpha component

*r*

set red component expression

*g*

set green component expression

*b*

set blue component expression

*a*

alpha component expression

*y*

set Y/luminance component expression

*u*

set U/Cb component expression

v

set V/Cr component expression

Each of them specifies the expression to use for computing the lookup table for the corresponding pixel component values.

The exact component associated to each of the *c*\* options depends on the format in input.

The *lut* filter requires either YUV or RGB pixel formats in input, *lutrgb* requires RGB pixel formats in input, and *lutyuv* requires YUV.

The expressions can contain the following constants and functions:

w

h

The input width and height.

val

The input value for the pixel component.

clipval

The input value, clipped to the *minval-maxval* range.

maxval

The maximum value for the pixel component.

minval

The minimum value for the pixel component.

negval

The negated value for the pixel component value, clipped to the *minval-maxval* range; it corresponds to the expression "maxval-clipval+minval".

clip(val)

The computed value in *val*, clipped to the *minval-maxval* range.

gammaval(gamma)

The computed gamma correction value of the pixel component value, clipped to the *minval-maxval* range. It corresponds to the expression

"pow((clipval-minval)/(maxval-minval),gamma)\*(maxval-minval)+minval"

All expressions default to "val".

### 37.70.1 Examples# TOC

- Negate input video:

```
lutrgb="r=maxval+minval-val:g=maxval+minval-val:b=maxval+minval-val"  
lutyuv="y=maxval+minval-val:u=maxval+minval-val:v=maxval+minval-val"
```

The above is the same as:

```
lutrgb="r=negval:g=negval:b=negval"  
lutyuv="y=negval:u=negval:v=negval"
```

- Negate luminance:

```
lutyuv=y=negval
```

- Remove chroma components, turning the video into a graytone image:

```
lutyuv="u=128:v=128"
```

- Apply a luma burning effect:

```
lutyuv="y=2*val"
```

- Remove green and blue components:

```
lutrgb="g=0:b=0"
```

- Set a constant alpha channel value on input:

```
format=rgba,lutrgb=a="maxval-minval/2"
```

- Correct luminance gamma by a factor of 0.5:

```
lutyuv=y=gammaval(0.5)
```

- Discard least significant bits of luma:

```
lutyuv=y='bitand(val, 128+64+32)'
```

### 37.71 mergeplanes# TOC

Merge color channel components from several video streams.

The filter accepts up to 4 input streams, and merge selected input planes to the output video.

This filter accepts the following options:

mapping

Set input to output plane mapping. Default is 0.

The mappings is specified as a bitmap. It should be specified as a hexadecimal number in the form 0xAa[Bb[Cc[Dd]]]. 'Aa' describes the mapping for the first plane of the output stream. 'A' sets the number of the input stream to use (from 0 to 3), and 'a' the plane number of the corresponding input to use (from 0 to 3). The rest of the mappings is similar, 'Bb' describes the mapping for the output stream second plane, 'Cc' describes the mapping for the output stream third plane and 'Dd' describes the mapping for the output stream fourth plane.

format

Set output pixel format. Default is yuva444p.

### 37.71.1 Examples# TOC

- Merge three gray video streams of same width and height into single video stream:

```
[a0][a1][a2]mergeplanes=0x001020:yuv444p
```

- Merge 1st yuv444p stream and 2nd gray video stream into yuva444p video stream:

```
[a0][a1]mergeplanes=0x00010210:yuva444p
```

- Swap Y and A plane in yuva444p stream:

```
format=yuva444p,mergeplanes=0x03010200:yuva444p
```

- Swap U and V plane in yuv420p stream:

```
format=yuv420p,mergeplanes=0x000201:yuv420p
```

- Cast a rgb24 clip to yuv444p:

```
format=rgb24,mergeplanes=0x000102:yuv444p
```

### 37.72 mcdeint# TOC

Apply motion-compensation deinterlacing.

It needs one field per frame as input and must thus be used together with yadif=1/3 or equivalent.

This filter accepts the following options:

mode

Set the deinterlacing mode.

It accepts one of the following values:

`'fast'`  
`'medium'`  
`'slow'`

use iterative motion estimation

`'extra_slow'`

like `'slow'`, but use multiple reference frames.

Default value is `'fast'`.

parity

Set the picture field parity assumed for the input video. It must be one of the following values:

`'0, tff'`

assume top field first

`'1, bff'`

assume bottom field first

Default value is `'bff'`.

qp

Set per-block quantization parameter (QP) used by the internal encoder.

Higher values should result in a smoother motion vector field but less optimal individual vectors.

Default value is 1.

## 37.73 mpdecimate# TOC

Drop frames that do not differ greatly from the previous frame in order to reduce frame rate.

The main use of this filter is for very-low-bitrate encoding (e.g. streaming over dialup modem), but it could in theory be used for fixing movies that were inverse-telecined incorrectly.

A description of the accepted options follows.

max

Set the maximum number of consecutive frames which can be dropped (if positive), or the minimum interval between dropped frames (if negative). If the value is 0, the frame is dropped unregarding the number of previous sequentially dropped frames.

Default value is 0.

`hi`  
`lo`  
`frac`

Set the dropping threshold values.

Values for `hi` and `lo` are for 8x8 pixel blocks and represent actual pixel value differences, so a threshold of 64 corresponds to 1 unit of difference for each pixel, or the same spread out differently over the block.

A frame is a candidate for dropping if no 8x8 blocks differ by more than a threshold of `hi`, and if no more than `frac` blocks (1 meaning the whole image) differ by more than a threshold of `lo`.

Default value for `hi` is 64\*12, default value for `lo` is 64\*5, and default value for `frac` is 0.33.

## 37.74 negate# TOC

Negate input video.

It accepts an integer in input; if non-zero it negates the alpha component (if available). The default value in input is 0.

## 37.75 noformat# TOC

Force libavfilter not to use any of the specified pixel formats for the input to the next filter.

It accepts the following parameters:

`pix_fmts`

A `'|'`-separated list of pixel format names, such as `apix_fmts=yuv420p|monow|rgb24`.

### 37.75.1 Examples# TOC

- Force libavfilter to use a format different from *yuv420p* for the input to the *vflip* filter:

```
noformat=pix_fmts=yuv420p,vflip
```

- Convert the input video to any of the formats not contained in the list:

```
noformat=yuv420p|yuv444p|yuv410p
```

## 37.76 noise# TOC

Add noise on video input frame.

The filter accepts the following options:

`all_seed`  
`c0_seed`  
`c1_seed`  
`c2_seed`  
`c3_seed`

Set noise seed for specific pixel component or all pixel components in case of *all\_seed*. Default value is 123457.

`all_strength, alls`  
`c0_strength, c0s`  
`c1_strength, c1s`  
`c2_strength, c2s`  
`c3_strength, c3s`

Set noise strength for specific pixel component or all pixel components in case *all\_strength*. Default value is 0. Allowed range is [0, 100].

`all_flags, allf`  
`c0_flags, c0f`  
`c1_flags, c1f`  
`c2_flags, c2f`  
`c3_flags, c3f`

Set pixel component flags or set flags for all components if *all\_flags*. Available values for component flags are:

‘a’

averaged temporal noise (smoother)

‘p’

mix random noise with a (semi)regular pattern

‘t’

temporal noise (noise pattern changes between frames)

‘u’

uniform noise (gaussian otherwise)

### 37.76.1 Examples# TOC

Add temporal and uniform noise to input video:

```
noise=all; s=20; allf=t+u
```

### 37.77 null# TOC

Pass the video source unchanged to the output.

### 37.78 ocv# TOC

Apply a video transform using libopencv.

To enable this filter, install the libopencv library and headers and configure FFmpeg with `--enable-libopencv`.

It accepts the following parameters:

`filter_name`

The name of the libopencv filter to apply.

`filter_params`

The parameters to pass to the libopencv filter. If not specified, the default values are assumed.

Refer to the official libopencv documentation for more precise information:

<http://docs.opencv.org/master/modules/imgproc/doc/filtering.html>

Several libopencv filters are supported; see the following subsections.

#### 37.78.1 dilate# TOC

Dilate an image by using a specific structuring element. It corresponds to the libopencv function `cvDilate`.

It accepts the parameters: *struct\_el* | *nb\_iterations*.

*struct\_el* represents a structuring element, and has the syntax: *cols* x *rows* + *anchor\_x* *anchor\_y* / *shape*

*cols* and *rows* represent the number of columns and rows of the structuring element, *anchor\_x* and *anchor\_y* the anchor point, and *shape* the shape for the structuring element. *shape* must be "rect", "cross", "ellipse", or "custom".



If the value for *shape* is "custom", it must be followed by a string of the form "*=filename*". The file with name *filename* is assumed to represent a binary image, with each printable character corresponding to a bright pixel. When a custom *shape* is used, *cols* and *rows* are ignored, the number of columns and rows of the read file are assumed instead.

The default value for *struct\_el* is "3x3+0x0/rect".

*nb\_iterations* specifies the number of times the transform is applied to the image, and defaults to 1.

Some examples:

```
# Use the default values
ocv=dilate

# Dilate using a structuring element with a 5x5 cross, iterating two times
ocv=filter_name=dilate:filter_params=5x5+2x2/cross|2

# Read the shape from the file diamond.shape, iterating two times.
# The file diamond.shape may contain a pattern of characters like this
#   *
#  ***
# *****
#  ***
#   *
# The specified columns and rows are ignored
# but the anchor point coordinates are not
ocv=dilate:0x0+2x2/custom=diamond.shape|2
```

### 37.78.2 erode# TOC

Erode an image by using a specific structuring element. It corresponds to the libopencv function `cvErode`.

It accepts the parameters: *struct\_el:nb\_iterations*, with the same syntax and semantics as the dilate filter.

### 37.78.3 smooth# TOC

Smooth the input video.

The filter takes the following parameters: *type|param1|param2|param3|param4*.

*type* is the type of smooth filter to apply, and must be one of the following values: "blur", "blur\_no\_scale", "median", "gaussian", or "bilateral". The default value is "gaussian".

The meaning of *param1*, *param2*, *param3*, and *param4* depend on the smooth type. *param1* and *param2* accept integer positive values or 0. *param3* and *param4* accept floating point values.

The default value for *param1* is 3. The default value for the other parameters is 0.

These parameters correspond to the parameters assigned to the libopencv function `cvSmooth`.

## 37.79 overlay# TOC

Overlay one video on top of another.

It takes two inputs and has one output. The first input is the "main" video on which the second input is overlaid.

It accepts the following parameters:

A description of the accepted options follows.

`x`

`y`

Set the expression for the x and y coordinates of the overlaid video on the main video. Default value is "0" for both expressions. In case the expression is invalid, it is set to a huge value (meaning that the overlay will not be displayed within the output visible area).

`eof_action`

The action to take when EOF is encountered on the secondary input; it accepts one of the following values:

`repeat`

Repeat the last frame (the default).

`endall`

End both streams.

`pass`

Pass the main input through.

`eval`

Set when the expressions for x, and y are evaluated.

It accepts the following values:

`'init'`

only evaluate expressions once during the filter initialization or when a command is processed

`'frame'`

evaluate expressions for each incoming frame

Default value is 'frame'.

`shortest`

If set to 1, force the output to terminate when the shortest input terminates. Default value is 0.

`format`

Set the format for the output video.

It accepts the following values:

'yuv420'

force YUV420 output

'yuv422'

force YUV422 output

'yuv444'

force YUV444 output

'rgb'

force RGB output

Default value is 'yuv420'.

`rgb` (*deprecated*)

If set to 1, force the filter to accept inputs in the RGB color space. Default value is 0. This option is deprecated, use `format` instead.

`repeatlast`

If set to 1, force the filter to draw the last overlay frame over the main input until the end of the stream. A value of 0 disables this behavior. Default value is 1.

The `x`, and `y` expressions can contain the following parameters.

`main_w`, `W`

`main_h`, `H`

The main input width and height.

overlay\_w, w  
overlay\_h, h

The overlay input width and height.

x  
y

The computed values for  $x$  and  $y$ . They are evaluated for each new frame.

hsub  
vsub

horizontal and vertical chroma subsample values of the output format. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

n

the number of input frame, starting from 0

pos

the position in the file of the input frame, NAN if unknown

t

The timestamp, expressed in seconds. It's NAN if the input timestamp is unknown.

Note that the  $n$ ,  $pos$ ,  $t$  variables are available only when evaluation is done *per frame*, and will evaluate to NAN when `eval` is set to 'init'.

Be aware that frames are taken from each input video in timestamp order, hence, if their initial timestamps differ, it is a good idea to pass the two inputs through a *setpts=PTS-STARTPTS* filter to have them begin in the same zero timestamp, as the example for the *movie* filter does.

You can chain together more overlays but you should test the efficiency of such approach.

### 37.79.1 Commands# TOC

This filter supports the following commands:

x  
y

Modify the  $x$  and  $y$  of the overlay input. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.79.2 Examples# TOC

- Draw the overlay at 10 pixels from the bottom right corner of the main video:

```
overlay=main_w-overlay_w-10:main_h-overlay_h-10
```

Using named options the example above becomes:

```
overlay=x=main_w-overlay_w-10:y=main_h-overlay_h-10
```

- Insert a transparent PNG logo in the bottom left corner of the input, using the `ffmpeg` tool with the `-filter_complex` option:

```
ffmpeg -i input -i logo -filter_complex 'overlay=10:main_h-overlay_h-10' output
```

- Insert 2 different transparent PNG logos (second logo on bottom right corner) using the `ffmpeg` tool:

```
ffmpeg -i input -i logo1 -i logo2 -filter_complex 'overlay=x=10:y=H-h-10,overlay=x=W-w-10:y=H-h-10' output
```

- Add a transparent color layer on top of the main video; `WxH` must specify the size of the main input to the overlay filter:

```
color=color=red@.3:size=WxH [over]; [in][over] overlay [out]
```

- Play an original video and a filtered version (here with the `deshake` filter) side by side using the `ffplay` tool:

```
ffplay input.avi -vf 'split[a][b]; [a]pad=iw*2:ih[src]; [b]deshake[filt]; [src][filt]overlay=w'
```

The above command is the same as:

```
ffplay input.avi -vf 'split[b], pad=iw*2[src], [b]deshake, [src]overlay=w'
```

- Make a sliding overlay appearing from the left to the right top part of the screen starting since time 2:

```
overlay=x='if(gte(t,2), -w+(t-2)*20, NAN)':y=0
```

- Compose output by putting two input videos side to side:

```
ffmpeg -i left.avi -i right.avi -filter_complex "  
nullsrc=size=200x100 [background];  
[0:v] setpts=PTS-STARTPTS, scale=100x100 [left];  
[1:v] setpts=PTS-STARTPTS, scale=100x100 [right];  
[background][left] overlay=shortest=1 [background+left];  
[background+left][right] overlay=shortest=1:x=100 [left+right]  
"
```

- Mask 10-20 seconds of a video by applying the `delogo` filter to a section

```
ffmpeg -i test.avi -codec:v:0 h264 -ar 11025 -b:v 9000k
-vf '[in]split[split_main][split_delogo];[split_delogo]trim=start=360:end=371,delogo=0:0:640:480[delogoed];[split_main][delogoed]overlay=eof_action=pass[out]'
masked.avi
```

- Chain several overlays in cascade:

```
nullsrc=s=200x200 [bg];
testsrc=s=100x100, split=4 [in0][in1][in2][in3];
[in0] lutrgb=r=0, [bg] overlay=0:0 [mid0];
[in1] lutrgb=g=0, [mid0] overlay=100:0 [mid1];
[in2] lutrgb=b=0, [mid1] overlay=0:100 [mid2];
[in3] null, [mid2] overlay=100:100 [out0]
```

## 37.80 owdenoise# TOC

Apply Overcomplete Wavelet denoiser.

The filter accepts the following options:

**depth**

Set depth.

Larger depth values will denoise lower frequency components more, but slow down filtering.

Must be an int in the range 8-16, default is 8.

**luma\_strength, ls**

Set luma strength.

Must be a double value in the range 0-1000, default is 1 . 0.

**chroma\_strength, cs**

Set chroma strength.

Must be a double value in the range 0-1000, default is 1 . 0.

## 37.81 pad# TOC

Add paddings to the input image, and place the original input at the provided *x*, *y* coordinates.

It accepts the following parameters:

**width, w**

**height, h**

Specify an expression for the size of the output image with the paddings added. If the value for *width* or *height* is 0, the corresponding input size is used for the output.

The *width* expression can reference the value set by the *height* expression, and vice versa.

The default value of *width* and *height* is 0.

*x*

*y*

Specify the offsets to place the input image at within the padded area, with respect to the top/left border of the output image.

The *x* expression can reference the value set by the *y* expression, and vice versa.

The default value of *x* and *y* is 0.

*color*

Specify the color of the padded area. For the syntax of this option, check the "Color" section in the ffmpeg-utils manual.

The default value of *color* is "black".

The value for the *width*, *height*, *x*, and *y* options are expressions containing the following constants:

*in\_w*

*in\_h*

The input video width and height.

*iw*

*ih*

These are the same as *in\_w* and *in\_h*.

*out\_w*

*out\_h*

The output width and height (the size of the padded area), as specified by the *width* and *height* expressions.

*ow*

*oh*

These are the same as *out\_w* and *out\_h*.

*x*

*y*

The x and y offsets as specified by the *x* and *y* expressions, or NAN if not yet specified.

a

same as *iw / ih*

sar

input sample aspect ratio

dar

input display aspect ratio, it is the same as  $(iw / ih) * sar$

hsub

vsub

The horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

### 37.81.1 Examples# TOC

- Add paddings with the color "violet" to the input video. The output video size is 640x480, and the top-left corner of the input video is placed at column 0, row 40

```
pad=640:480:0:40:violet
```

The example above is equivalent to the following command:

```
pad=width=640:height=480:x=0:y=40:color=violet
```

- Pad the input to get an output with dimensions increased by 3/2, and put the input video at the center of the padded area:

```
pad="3/2*iw:3/2*ih:(ow-iw)/2:(oh-ih)/2"
```

- Pad the input to get a squared output with size equal to the maximum value between the input width and height, and put the input video at the center of the padded area:

```
pad="max(iw\,ih):ow:(ow-iw)/2:(oh-ih)/2"
```

- Pad the input to get a final w/h ratio of 16:9:

```
pad="ih*16/9:ih:(ow-iw)/2:(oh-ih)/2"
```

- In case of anamorphic video, in order to set the output display aspect correctly, it is necessary to use *sar* in the expression, according to the relation:

```
(ih * X / ih) * sar = output_dar  
X = output_dar / sar
```



Thus the previous example needs to be modified to:

```
pad="ih*16/9/sar:ih:(ow-iw)/2:(oh-ih)/2"
```

- Double the output size and put the input video in the bottom-right corner of the output padded area:

```
pad="2*iw:2*ih:ow-iw:oh-ih"
```

## 37.82 palettegen# TOC

Generate one palette for a whole video stream.

It accepts the following options:

`max_colors`

Set the maximum number of colors to quantize in the palette. Note: the palette will still contain 256 colors; the unused palette entries will be black.

`reserve_transparent`

Create a palette of 255 colors maximum and reserve the last one for transparency. Reserving the transparency color is useful for GIF optimization. If not set, the maximum of colors in the palette will be 256. You probably want to disable this option for a standalone image. Set by default.

`stats_mode`

Set statistics mode.

It accepts the following values:

`'full'`

Compute full frame histograms.

`'diff'`

Compute histograms only for the part that differs from previous frame. This might be relevant to give more importance to the moving part of your input if the background is static.

Default value is *full*.

The filter also exports the frame metadata `lavfi.color_quant_ratio(nb_color_in / nb_color_out)` which you can use to evaluate the degree of color quantization of the palette. This information is also visible at *info* logging level.

## 37.82.1 Examples# TOC

- Generate a representative palette of a given video using `ffmpeg`:

```
ffmpeg -i input.mkv -vf palettegen palette.png
```

## 37.83 paletteuse# TOC

Use a palette to downsample an input video stream.

The filter takes two inputs: one video stream and a palette. The palette must be a 256 pixels image.

It accepts the following options:

`dither`

Select dithering mode. Available algorithms are:

`'bayer'`

Ordered 8x8 bayer dithering (deterministic)

`'heckbert'`

Dithering as defined by Paul Heckbert in 1982 (simple error diffusion). Note: this dithering is sometimes considered "wrong" and is included as a reference.

`'floyd_steinberg'`

Floyd and Steingberg dithering (error diffusion)

`'sierra2'`

Frankie Sierra dithering v2 (error diffusion)

`'sierra2_4a'`

Frankie Sierra dithering v2 "Lite" (error diffusion)

Default is *sierra2\_4a*.

`bayer_scale`

When *bayer* dithering is selected, this option defines the scale of the pattern (how much the crosshatch pattern is visible). A low value means more visible pattern for less banding, and higher value means less visible pattern at the cost of more banding.

The option must be an integer value in the range [0,5]. Default is 2.

`diff_mode`

If set, define the zone to process

`'rectangle'`

Only the changing rectangle will be reprocessed. This is similar to GIF cropping/offsetting compression mechanism. This option can be useful for speed if only a part of the image is changing, and has use cases such as limiting the scope of the error diffusal dither to the rectangle that bounds the moving scene (it leads to more deterministic output if the scene doesn't change much, and as a result less moving noise and better GIF compression).

Default is *none*.

### 37.83.1 Examples# TOC

- Use a palette (generated for example with `palettegen`) to encode a GIF using `ffmpeg`:

```
ffmpeg -i input.mkv -i palette.png -lavfi paletteuse output.gif
```

### 37.84 perspective# TOC

Correct perspective of video not recorded perpendicular to the screen.

A description of the accepted parameters follows.

`x0`  
`y0`  
`x1`  
`y1`  
`x2`  
`y2`  
`x3`  
`y3`

Set coordinates expression for top left, top right, bottom left and bottom right corners. Default values are `0:0:W:0:0:H:W:H` with which perspective will remain unchanged. If the `sense` option is set to `source`, then the specified points will be sent to the corners of the destination. If the `sense` option is set to `destination`, then the corners of the source will be sent to the specified coordinates.

The expressions can use the following variables:

`W`  
`H`

the width and height of video frame.

interpolation

Set interpolation for perspective correction.

It accepts the following values:

‘linear’

‘cubic’

Default value is ‘linear’.

sense

Set interpretation of coordinate options.

It accepts the following values:

‘0, source’

Send point in the source specified by the given coordinates to the corners of the destination.

‘1, destination’

Send the corners of the source to the point in the destination specified by the given coordinates.

Default value is ‘source’.

## **37.85 phase# TOC**

Delay interlaced video by one field time so that the field order changes.

The intended use is to fix PAL movies that have been captured with the opposite field order to the film-to-video transfer.

A description of the accepted parameters follows.

mode

Set phase mode.

It accepts the following values:

‘t’

Capture field order top-first, transfer bottom-first. Filter will delay the bottom field.

‘b’

Capture field order bottom-first, transfer top-first. Filter will delay the top field.

‘p’

Capture and transfer with the same field order. This mode only exists for the documentation of the other options to refer to, but if you actually select it, the filter will faithfully do nothing.

‘a’

Capture field order determined automatically by field flags, transfer opposite. Filter selects among ‘t’ and ‘b’ modes on a frame by frame basis using field flags. If no field information is available, then this works just like ‘u’.

‘u’

Capture unknown or varying, transfer opposite. Filter selects among ‘t’ and ‘b’ on a frame by frame basis by analyzing the images and selecting the alternative that produces best match between the fields.

‘T’

Capture top-first, transfer unknown or varying. Filter selects among ‘t’ and ‘p’ using image analysis.

‘B’

Capture bottom-first, transfer unknown or varying. Filter selects among ‘b’ and ‘p’ using image analysis.

‘A’

Capture determined by field flags, transfer unknown or varying. Filter selects among ‘t’, ‘b’ and ‘p’ using field flags and image analysis. If no field information is available, then this works just like ‘U’. This is the default mode.

‘U’

Both capture and transfer unknown or varying. Filter selects among ‘t’, ‘b’ and ‘p’ using image analysis only.

## 37.86 pixdesctest# TOC

Pixel format descriptor test filter, mainly useful for internal testing. The output video should be equal to the input video.

For example:

```
format=monow, pixdesctest
```

can be used to test the monowhite pixel format descriptor definition.

## 37.87 pp# TOC

Enable the specified chain of postprocessing subfilters using libpostproc. This library should be automatically selected with a GPL build (`--enable-gpl`). Subfilters must be separated by '/' and can be disabled by prepending a '-'. Each subfilter and some options have a short and a long name that can be used interchangeably, i.e. `dr/dering` are the same.

The filters accept the following options:

```
subfilters
```

Set postprocessing subfilters string.

All subfilters share common options to determine their scope:

```
a/autoq
```

Honor the quality commands for this subfilter.

```
c/chrom
```

Do chrominance filtering, too (default).

```
y/nochrom
```

Do luminance filtering only (no chrominance).

```
n/noluma
```

Do chrominance filtering only (no luminance).

These options can be appended after the subfilter name, separated by a '|'.

Available subfilters are:

```
hb/hdeblock[|difference[|flatness]]
```

Horizontal deblocking filter

```
difference
```

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

vb/vdeblock[ |difference[ |flatness]]

Vertical deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

ha/hadeblock[ |difference[ |flatness]]

Accurate horizontal deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

va/vadeblock[ |difference[ |flatness]]

Accurate vertical deblocking filter

difference

Difference factor where higher values mean more deblocking (default: 32).

flatness

Flatness threshold where lower values mean more deblocking (default: 39).

The horizontal and vertical deblocking filters share the difference and flatness values so you cannot set different horizontal and vertical thresholds.

h1/x1hdeblock

Experimental horizontal deblocking filter

v1/x1vdeblock

## Experimental vertical deblocking filter

dr/dering

### Deringing filter

tn/tmpnoise[|threshold1[|threshold2[|threshold3]]], temporal noise reducer

threshold1

larger -> stronger filtering

threshold2

larger -> stronger filtering

threshold3

larger -> stronger filtering

al/autolevels[:f/fullyrange], automatic brightness / contrast correction

f/fullyrange

Stretch luminance to 0-255.

lb/linblenddeint

Linear blend deinterlacing filter that deinterlaces the given block by filtering all lines with a (1 2 1) filter.

li/linipoldeint

Linear interpolating deinterlacing filter that deinterlaces the given block by linearly interpolating every second line.

ci/cubicipoldeint

Cubic interpolating deinterlacing filter deinterlaces the given block by cubically interpolating every second line.

md/mediandeint

Median deinterlacing filter that deinterlaces the given block by applying a median filter to every second line.

fd/ffmpegdeint



FFmpeg deinterlacing filter that deinterlaces the given block by filtering every second line with a  $(-1 \ 4 \ 2 \ 4 \ -1)$  filter.

l5/lowpass5

Vertically applied FIR lowpass deinterlacing filter that deinterlaces the given block by filtering all lines with a  $(-1 \ 2 \ 6 \ 2 \ -1)$  filter.

fq/forceQuant[|quantizer]

Overrides the quantizer table from the input with the constant quantizer you specify.

quantizer

Quantizer to use

de/default

Default pp filter combination (hb|a,vb|a,dr|a)

fa/fast

Fast pp filter combination (h1|a,v1|a,dr|a)

ac

High quality pp filter combination (ha|a|128|7,va|a,dr|a)

### 37.87.1 Examples# TOC

- Apply horizontal and vertical deblocking, deringing and automatic brightness/contrast:

pp=hb/vb/dr/al

- Apply default filters without brightness/contrast correction:

pp=de/-al

- Apply default filters and temporal denoiser:

pp=default/tmpnoise|1|2|3

- Apply deblocking on luminance only, and switch vertical deblocking on or off automatically depending on available CPU time:

pp=hb|y/vb|a

## 37.88 pp7# TOC

Apply Postprocessing filter 7. It is variant of the spp filter, similar to spp = 6 with 7 point DCT, where only the center sample is used after IDCT.

The filter accepts the following options:

qp

Force a constant quantization parameter. It accepts an integer in range 0 to 63. If not set, the filter will use the QP from the video stream (if available).

mode

Set thresholding mode. Available modes are:

'hard'

Set hard thresholding.

'soft'

Set soft thresholding (better de-ringing effect, but likely blurrier).

'medium'

Set medium thresholding (good results, default).

## 37.89 psnr# TOC

Obtain the average, maximum and minimum PSNR (Peak Signal to Noise Ratio) between two input videos.

This filter takes in input two input videos, the first input is considered the "main" source and is passed unchanged to the output. The second input is used as a "reference" video for computing the PSNR.

Both video inputs must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The obtained average PSNR is printed through the logging system.

The filter stores the accumulated MSE (mean squared error) of each frame, and at the end of the processing it is averaged across all frames equally, and the following formula is applied to obtain the PSNR:

$$\text{PSNR} = 10 \cdot \log_{10}(\text{MAX}^2 / \text{MSE})$$

Where MAX is the average of the maximum values of each component of the image.

The description of the accepted parameters follows.

`stats_file, f`

If specified the filter will use the named file to save the PSNR of each individual frame.

The file printed if *stats\_file* is selected, contains a sequence of key/value pairs of the form *key:value* for each compared couple of frames.

A description of each shown parameter follows:

`n`

sequential number of the input frame, starting from 1

`mse_avg`

Mean Square Error pixel-by-pixel average difference of the compared frames, averaged over all the image components.

`mse_y, mse_u, mse_v, mse_r, mse_g, mse_g, mse_a`

Mean Square Error pixel-by-pixel average difference of the compared frames for the component specified by the suffix.

`psnr_y, psnr_u, psnr_v, psnr_r, psnr_g, psnr_b, psnr_a`

Peak Signal to Noise ratio of the compared frames for the component specified by the suffix.

For example:

```
movie=ref_movie.mpg, setpts=PTS-STARTPTS [main];  
[main][ref] psnr="stats_file=stats.log" [out]
```

On this example the input file being processed is compared with the reference file `ref_movie.mpg`. The PSNR of each individual frame is stored in `stats.log`.

## 37.90 pullup# TOC

Pulldown reversal (inverse telecine) filter, capable of handling mixed hard-telecine, 24000/1001 fps progressive, and 30000/1001 fps progressive content.

The pullup filter is designed to take advantage of future context in making its decisions. This filter is stateless in the sense that it does not lock onto a pattern to follow, but it instead looks forward to the following fields in order to identify matches and rebuild progressive frames.

To produce content with an even framerate, insert the fps filter after pullup, use  $\text{fps}=24000/1001$  if the input frame rate is 29.97fps,  $\text{fps}=24$  for 30fps and the (rare) telecined 25fps input.

The filter accepts the following options:

j1  
jr  
jt  
jb

These options set the amount of "junk" to ignore at the left, right, top, and bottom of the image, respectively. Left and right are in units of 8 pixels, while top and bottom are in units of 2 lines. The default is 8 pixels on each side.

sb

Set the strict breaks. Setting this option to 1 will reduce the chances of filter generating an occasional mismatched frame, but it may also cause an excessive number of frames to be dropped during high motion sequences. Conversely, setting it to -1 will make filter match fields more easily. This may help processing of video where there is slight blurring between the fields, but may also cause there to be interlaced frames in the output. Default value is 0.

mp

Set the metric plane to use. It accepts the following values:

'l'

Use luma plane.

'u'

Use chroma blue plane.

'v'

Use chroma red plane.

This option may be set to use chroma plane instead of the default luma plane for doing filter's computations. This may improve accuracy on very clean source material, but more likely will decrease accuracy, especially if there is chroma noise (rainbow effect) or any grayscale video. The main purpose of setting mp to a chroma plane is to reduce CPU load and make pullup usable in realtime on slow machines.

For best results (without duplicated frames in the output file) it is necessary to change the output frame rate. For example, to inverse telecine NTSC input:

```
ffmpeg -i input -vf pullup -r 24000/1001 ...
```

## 37.91 qp# TOC

Change video quantization parameters (QP).

The filter accepts the following option:

**qp**

Set expression for quantization parameter.

The expression is evaluated through the eval API and can contain, among others, the following constants:

*known*

1 if index is not 129, 0 otherwise.

*qp*

Sequential index starting from -129 to 128.

### 37.91.1 Examples# TOC

- Some equation like:

```
qp=2+2*sin(PI*qp)
```

## 37.92 random# TOC

Flush video frames from internal cache of frames into a random order. No frame is discarded. Inspired by frei0r nervous filter.

**frames**

Set size in number of frames of internal cache, in range from 2 to 512. Default is 30.

**seed**

Set seed for random number generator, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to less than 0, the filter will try to use a good random seed on a best effort basis.

## 37.93 removegrain# TOC

The removegrain filter is a spatial denoiser for progressive video.

m0

Set mode for the first plane.

m1

Set mode for the second plane.

m2

Set mode for the third plane.

m3

Set mode for the fourth plane.

Range of mode is from 0 to 24. Description of each mode follows:

0

Leave input plane unchanged. Default.

1

Clips the pixel with the minimum and maximum of the 8 neighbour pixels.

2

Clips the pixel with the second minimum and maximum of the 8 neighbour pixels.

3

Clips the pixel with the third minimum and maximum of the 8 neighbour pixels.

4

Clips the pixel with the fourth minimum and maximum of the 8 neighbour pixels. This is equivalent to a median filter.

5

Line-sensitive clipping giving the minimal change.

6

Line-sensitive clipping, intermediate.

7

Line-sensitive clipping, intermediate.

8

Line-sensitive clipping, intermediate.

9

Line-sensitive clipping on a line where the neighbours pixels are the closest.

10

Replaces the target pixel with the closest neighbour.

11

[1 2 1] horizontal and vertical kernel blur.

12

Same as mode 11.

13

Bob mode, interpolates top field from the line where the neighbours pixels are the closest.

14

Bob mode, interpolates bottom field from the line where the neighbours pixels are the closest.

15

Bob mode, interpolates top field. Same as 13 but with a more complicated interpolation formula.

16

Bob mode, interpolates bottom field. Same as 14 but with a more complicated interpolation formula.

17

Clips the pixel with the minimum and maximum of respectively the maximum and minimum of each pair of opposite neighbour pixels.

18

Line-sensitive clipping using opposite neighbours whose greatest distance from the current pixel is minimal.

19

Replaces the pixel with the average of its 8 neighbours.

20

Averages the 9 pixels ([1 1 1] horizontal and vertical blur).

21

Clips pixels using the averages of opposite neighbour.

22

Same as mode 21 but simpler and faster.

23

Small edge and halo removal, but reputed useless.

24

Similar as 23.

## 37.94 removelogo# TOC

Suppress a TV station logo, using an image file to determine which pixels comprise the logo. It works by filling in the pixels that comprise the logo with neighboring pixels.

The filter accepts the following options:

`filename, f`

Set the filter bitmap file, which can be any image format supported by libavformat. The width and height of the image file must match those of the video stream being processed.

Pixels in the provided bitmap image with a value of zero are not considered part of the logo, non-zero pixels are considered part of the logo. If you use white (255) for the logo and black (0) for the rest, you will be safe. For making the filter bitmap, it is recommended to take a screen capture of a black frame with the logo visible, and then using a threshold filter followed by the erode filter once or twice.

If needed, little splotches can be fixed manually. Remember that if logo pixels are not covered, the filter quality will be much reduced. Marking too many pixels as part of the logo does not hurt as much, but it will increase the amount of blurring needed to cover over the image and will destroy more information than necessary, and extra pixels will slow things down on a large logo.



## 37.95 repeatfields# TOC

This filter uses the repeat\_field flag from the Video ES headers and hard repeats fields based on its value.

## 37.96 reverse, areverse# TOC

Reverse a clip.

Warning: This filter requires memory to buffer the entire clip, so trimming is suggested.

### 37.96.1 Examples# TOC

- Take the first 5 seconds of a clip, and reverse it.

```
trim=end=5,reverse
```

## 37.97 rotate# TOC

Rotate video by an arbitrary angle expressed in radians.

The filter accepts the following options:

A description of the optional parameters follows.

angle, a

Set an expression for the angle by which to rotate the input video clockwise, expressed as a number of radians. A negative value will result in a counter-clockwise rotation. By default it is set to "0".

This expression is evaluated for each frame.

out\_w, ow

Set the output width expression, default value is "iw". This expression is evaluated just once during configuration.

out\_h, oh

Set the output height expression, default value is "ih". This expression is evaluated just once during configuration.

bilinear

Enable bilinear interpolation if set to 1, a value of 0 disables it. Default value is 1.

fillcolor, c

Set the color used to fill the output area not covered by the rotated image. For the general syntax of this option, check the "Color" section in the ffmpeg-utils manual. If the special value "none" is selected then no background is printed (useful for example if the background is never shown).

Default value is "black".

The expressions for the angle and the output size can contain the following constants and functions:

`n`

sequential number of the input frame, starting from 0. It is always NAN before the first frame is filtered.

`t`

time in seconds of the input frame, it is set to 0 when the filter is configured. It is always NAN before the first frame is filtered.

`hsub`

`vsub`

horizontal and vertical chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

`in_w, iw`

`in_h, ih`

the input video width and height

`out_w, ow`

`out_h, oh`

the output width and height, that is the size of the padded area as specified by the *width* and *height* expressions

`rotw(a)`

`roth(a)`

the minimal width/height required for completely containing the input video rotated by *a* radians.

These are only available when computing the `out_w` and `out_h` expressions.

### 37.97.1 Examples# TOC

- Rotate the input by  $\text{PI}/6$  radians clockwise:

```
rotate=PI/6
```

- Rotate the input by  $\text{PI}/6$  radians counter-clockwise:

```
rotate=-PI/6
```

- Rotate the input by 45 degrees clockwise:

```
rotate=45*PI/180
```

- Apply a constant rotation with period T, starting from an angle of  $\pi/3$ :

```
rotate=PI/3+2*PI*t/T
```

- Make the input video rotation oscillating with a period of T seconds and an amplitude of A radians:

```
rotate=A*sin(2*PI/T*t)
```

- Rotate the video, output size is chosen so that the whole rotating input video is always completely contained in the output:

```
rotate=2*PI*t:ow=hypot(iw,ih):oh=ow'
```

- Rotate the video, reduce the output size so that no background is ever shown:

```
rotate=2*PI*t:ow='min(iw,ih)/sqrt(2)':oh=ow:c=none
```

### 37.97.2 Commands# TOC

The filter supports the following commands:

a, angle

Set the angle expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.98 sab# TOC

Apply Shape Adaptive Blur.

The filter accepts the following options:

luma\_radius, lr

Set luma blur filter strength, must be a value in range 0.1-4.0, default value is 1.0. A greater value will result in a more blurred image, and in slower processing.

luma\_pre\_filter\_radius, lpfr

Set luma pre-filter radius, must be a value in the 0.1-2.0 range, default value is 1.0.

luma\_strength, ls

Set luma maximum difference between pixels to still be considered, must be a value in the 0.1-100.0 range, default value is 1.0.

`chroma_radius, cr`

Set chroma blur filter strength, must be a value in range 0.1-4.0. A greater value will result in a more blurred image, and in slower processing.

`chroma_pre_filter_radius, cpfr`

Set chroma pre-filter radius, must be a value in the 0.1-2.0 range.

`chroma_strength, cs`

Set chroma maximum difference between pixels to still be considered, must be a value in the 0.1-100.0 range.

Each chroma option value, if not explicitly specified, is set to the corresponding luma option value.

## **37.99 scale# TOC**

Scale (resize) the input video, using the libswscale library.

The scale filter forces the output display aspect ratio to be the same of the input, by changing the output sample aspect ratio.

If the input image format is different from the format requested by the next filter, the scale filter will convert the input to the requested format.

### **37.99.1 Options# TOC**

The filter accepts the following options, or any of the options supported by the libswscale scaler.

See (ffmpeg-scaler)the ffmpeg-scaler manual for the complete list of scaler options.

`width, w`

`height, h`

Set the output video dimension expression. Default value is the input dimension.

If the value is 0, the input width is used for the output.

If one of the values is -1, the scale filter will use a value that maintains the aspect ratio of the input image, calculated from the other specified dimension. If both of them are -1, the input size is used

If one of the values is -n with  $n > 1$ , the scale filter will also use a value that maintains the aspect ratio of the input image, calculated from the other specified dimension. After that it will, however, make sure that the calculated dimension is divisible by n and adjust the value if necessary.

See below for the list of accepted constants for use in the dimension expression.

`interl`

Set the interlacing mode. It accepts the following values:

‘1’

Force interlaced aware scaling.

‘0’

Do not apply interlaced scaling.

‘-1’

Select interlaced aware scaling depending on whether the source frames are flagged as interlaced or not.

Default value is ‘0’.

`flags`

Set libswscale scaling flags. See (ffmpeg-scaler)the ffmpeg-scaler manual for the complete list of values. If not explicitly specified the filter applies the default flags.

`size, s`

Set the video size. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

`in_color_matrix`

`out_color_matrix`

Set in/output YCbCr color space type.

This allows the autodetected value to be overridden as well as allows forcing a specific value used for the output and encoder.

If not specified, the color space type depends on the pixel format.

Possible values:

‘auto’

Choose automatically.

‘bt709’

Format conforming to International Telecommunication Union (ITU) Recommendation BT.709.

`'fcc'`

Set color space conforming to the United States Federal Communications Commission (FCC) Code of Federal Regulations (CFR) Title 47 (2003) 73.682 (a).

`'bt601'`

Set color space conforming to:

- ITU Radiocommunication Sector (ITU-R) Recommendation BT.601
- ITU-R Rec. BT.470-6 (1998) Systems B, B1, and G
- Society of Motion Picture and Television Engineers (SMPTE) ST 170:2004

`'smpte240m'`

Set color space conforming to SMPTE ST 240:1999.

`in_range`

`out_range`

Set in/output YCbCr sample range.

This allows the autodetected value to be overridden as well as allows forcing a specific value used for the output and encoder. If not specified, the range depends on the pixel format. Possible values:

`'auto'`

Choose automatically.

`'jpeg/full/pc'`

Set full range (0-255 in case of 8-bit luma).

`'mpeg/tv'`

Set "MPEG" range (16-235 in case of 8-bit luma).

`force_original_aspect_ratio`

Enable decreasing or increasing output video width or height if necessary to keep the original aspect ratio. Possible values:

`'disable'`

Scale the video as specified and disable this feature.

`'decrease'`

The output video dimensions will automatically be decreased if needed.

‘increase’

The output video dimensions will automatically be increased if needed.

One useful instance of this option is that when you know a specific device’s maximum allowed resolution, you can use this to limit the output video to that, while retaining the aspect ratio. For example, device A allows 1280x720 playback, and your video is 1920x800. Using this option (set it to decrease) and specifying 1280x720 to the command line makes the output 1280x533.

Please note that this is a different thing than specifying -1 for w or h, you still need to specify the output resolution for this option to work.

The values of the w and h options are expressions containing the following constants:

*in\_w*  
*in\_h*

The input width and height

*iw*  
*ih*

These are the same as *in\_w* and *in\_h*.

*out\_w*  
*out\_h*

The output (scaled) width and height

*ow*  
*oh*

These are the same as *out\_w* and *out\_h*

*a*

The same as *iw* / *ih*

*sar*

input sample aspect ratio

*dar*

The input display aspect ratio. Calculated from  $(iw / ih) * sar$ .

*hsub*  
*vsub*

horizontal and vertical input chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

*ohsub*  
*ovsub*

horizontal and vertical output chroma subsample values. For example for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

## 37.99.2 Examples# TOC

- Scale the input video to a size of 200x100

```
scale=w=200:h=100
```

This is equivalent to:

```
scale=200:100
```

or:

```
scale=200x100
```

- Specify a size abbreviation for the output size:

```
scale=qcif
```

which can also be written as:

```
scale=size=qcif
```

- Scale the input to 2x:

```
scale=w=2*iw:h=2*ih
```

- The above is the same as:

```
scale=2*in_w:2*in_h
```

- Scale the input to 2x with forced interlaced scaling:

```
scale=2*iw:2*ih:interl=1
```

- Scale the input to half size:

```
scale=w=iw/2:h=ih/2
```

- Increase the width, and set the height to the same size:



```
scale=3/2*iw:ow
```

- Seek Greek harmony:

```
scale=iw:1/PHI*iw  
scale=ih*PHI:ih
```

- Increase the height, and set the width to 3/2 of the height:

```
scale=w=3/2*oh:h=3/5*ih
```

- Increase the size, making the size a multiple of the chroma subsample values:

```
scale="trunc(3/2*iw/hsub)*hsub:trunc(3/2*ih/vsub)*vsub"
```

- Increase the width to a maximum of 500 pixels, keeping the same aspect ratio as the input:

```
scale=w='min(500\, iw*3/2):h=-1'
```

### 37.99.3 Commands# TOC

This filter supports the following commands:

```
width, w  
height, h
```

Set the output video dimension expression. The command accepts the same syntax of the corresponding option.

If the specified expression is not valid, it is kept at its current value.

### 37.100 scale2ref# TOC

Scale (resize) the input video, based on a reference video.

See the scale filter for available options, scale2ref supports the same but uses the reference video instead of the main input as basis.

#### 37.100.1 Examples# TOC

- Scale a subtitle stream to match the main video in size before overlaying

```
'scale2ref[b][a];[a][b]overlay'
```

### 37.101 separatefields# TOC

The `separatefields` takes a frame-based video input and splits each frame into its components fields, producing a new half height clip with twice the frame rate and twice the frame count.

This filter use field-dominance information in frame to decide which of each pair of fields to place first in the output. If it gets it wrong use `setfield` filter before `separatefields` filter.

## 37.102 `setdar`, `setsar`# TOC

The `setdar` filter sets the Display Aspect Ratio for the filter output video.

This is done by changing the specified Sample (aka Pixel) Aspect Ratio, according to the following equation:

$$DAR = HORIZONTAL\_RESOLUTION / VERTICAL\_RESOLUTION * SAR$$

Keep in mind that the `setdar` filter does not modify the pixel dimensions of the video frame. Also, the display aspect ratio set by this filter may be changed by later filters in the filterchain, e.g. in case of scaling or if another "setdar" or a "setsar" filter is applied.

The `setsar` filter sets the Sample (aka Pixel) Aspect Ratio for the filter output video.

Note that as a consequence of the application of this filter, the output display aspect ratio will change according to the equation above.

Keep in mind that the sample aspect ratio set by the `setsar` filter may be changed by later filters in the filterchain, e.g. if another "setsar" or a "setdar" filter is applied.

It accepts the following parameters:

`r`, `ratio`, `dar` (`setdar` only), `sar` (`setsar` only)

Set the aspect ratio used by the filter.

The parameter can be a floating point number string, an expression, or a string of the form *num:den*, where *num* and *den* are the numerator and denominator of the aspect ratio. If the parameter is not specified, it is assumed the value "0". In case the form "*num:den*" is used, the `:` character should be escaped.

`max`

Set the maximum integer value to use for expressing numerator and denominator when reducing the expressed aspect ratio to a rational. Default value is 100.

The parameter *sar* is an expression containing the following constants:

`E`, `PI`, `PHI`

These are approximated values for the mathematical constants *e* (Euler's number), *pi* (Greek pi), and *phi* (the golden ratio).

`w, h`

The input width and height.

`a`

These are the same as  $w / h$ .

`sar`

The input sample aspect ratio.

`dar`

The input display aspect ratio. It is the same as  $(w / h) * sar$ .

`hsub, vsub`

Horizontal and vertical chroma subsample values. For example, for the pixel format "yuv422p" *hsub* is 2 and *vsub* is 1.

### 37.102.1 Examples# TOC

- To change the display aspect ratio to 16:9, specify one of the following:

```
setdar=dar=1.77777
setdar=dar=16/9
setdar=dar=1.77777
```

- To change the sample aspect ratio to 10:11, specify:

```
setsar=sar=10/11
```

- To set a display aspect ratio of 16:9, and specify a maximum integer value of 1000 in the aspect ratio reduction, use the command:

```
setdar=ratio=16/9:max=1000
```

### 37.103 setfield# TOC

Force field for the output video frame.

The `setfield` filter marks the interlace type field for the output frames. It does not change the input frame, but only sets the corresponding property, which affects how the frame is treated by following filters (e.g. `fieldorder` or `yadif`).

The filter accepts the following options:

mode

Available values are:

‘auto’

Keep the same field property.

‘bff’

Mark the frame as bottom-field-first.

‘tff’

Mark the frame as top-field-first.

‘prog’

Mark the frame as progressive.

## 37.104 showinfo# TOC

Show a line containing various information for each input video frame. The input video is not modified.

The shown line contains a sequence of key/value pairs of the form *key:value*.

The following values are shown in the output:

n

The (sequential) number of the input frame, starting from 0.

pts

The Presentation TimeStamp of the input frame, expressed as a number of time base units. The time base unit depends on the filter input pad.

pts\_time

The Presentation TimeStamp of the input frame, expressed as a number of seconds.

pos

The position of the frame in the input stream, or -1 if this information is unavailable and/or meaningless (for example in case of synthetic video).

fmt

The pixel format name.

`sar`

The sample aspect ratio of the input frame, expressed in the form *num/den*.

`s`

The size of the input frame. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

`i`

The type of interlaced mode ("P" for "progressive", "T" for top field first, "B" for bottom field first).

`iskey`

This is 1 if the frame is a key frame, 0 otherwise.

`type`

The picture type of the input frame ("I" for an I-frame, "P" for a P-frame, "B" for a B-frame, or "?" for an unknown type). Also refer to the documentation of the `AVPictureType` enum and of the `av_get_picture_type_char` function defined in `libavutil/avutil.h`.

`checksum`

The Adler-32 checksum (printed in hexadecimal) of all the planes of the input frame.

`plane_checksum`

The Adler-32 checksum (printed in hexadecimal) of each plane of the input frame, expressed in the form "[*c0 c1 c2 c3*]".

## 37.105 showpalette# TOC

Displays the 256 colors palette of each frame. This filter is only relevant for *pal8* pixel format frames.

It accepts the following option:

`s`

Set the size of the box used to represent one palette color entry. Default is 30 (for a 30×30 pixel box).

## 37.106 shuffleplanes# TOC

Reorder and/or duplicate video planes.

It accepts the following parameters:

map0

The index of the input plane to be used as the first output plane.

map1

The index of the input plane to be used as the second output plane.

map2

The index of the input plane to be used as the third output plane.

map3

The index of the input plane to be used as the fourth output plane.

The first plane has the index 0. The default is to keep the input unchanged.

Swap the second and third planes of the input:

```
ffmpeg -i INPUT -vf shuffleplanes=0:2:1:3 OUTPUT
```

## 37.107 signalstats# TOC

Evaluate various visual metrics that assist in determining issues associated with the digitization of analog video media.

By default the filter will log these metadata values:

YMIN

Display the minimal Y value contained within the input frame. Expressed in range of [0-255].

YLOW

Display the Y value at the 10% percentile within the input frame. Expressed in range of [0-255].

YAVG

Display the average Y value within the input frame. Expressed in range of [0-255].

YHIGH

Display the Y value at the 90% percentile within the input frame. Expressed in range of [0-255].

YMAX

Display the maximum Y value contained within the input frame. Expressed in range of [0-255].

UMIN

Display the minimal U value contained within the input frame. Expressed in range of [0-255].

ULOW

Display the U value at the 10% percentile within the input frame. Expressed in range of [0-255].

UAVG

Display the average U value within the input frame. Expressed in range of [0-255].

UHIGH

Display the U value at the 90% percentile within the input frame. Expressed in range of [0-255].

UMAX

Display the maximum U value contained within the input frame. Expressed in range of [0-255].

VMIN

Display the minimal V value contained within the input frame. Expressed in range of [0-255].

VLOW

Display the V value at the 10% percentile within the input frame. Expressed in range of [0-255].

VAVG

Display the average V value within the input frame. Expressed in range of [0-255].

VHIGH

Display the V value at the 90% percentile within the input frame. Expressed in range of [0-255].

VMAX

Display the maximum V value contained within the input frame. Expressed in range of [0-255].

SATMIN

Display the minimal saturation value contained within the input frame. Expressed in range of [0~181.02].

SATLOW

Display the saturation value at the 10% percentile within the input frame. Expressed in range of [0~181.02].

SATAVG

Display the average saturation value within the input frame. Expressed in range of [0~181.02].

SATHIGH

Display the saturation value at the 90% percentile within the input frame. Expressed in range of [0~181.02].

SATMAX

Display the maximum saturation value contained within the input frame. Expressed in range of [0~181.02].

HUEMED

Display the median value for hue within the input frame. Expressed in range of [0-360].

HUEAVG

Display the average value for hue within the input frame. Expressed in range of [0-360].

YDIF

Display the average of sample value difference between all values of the Y plane in the current frame and corresponding values of the previous input frame. Expressed in range of [0-255].

UDIF

Display the average of sample value difference between all values of the U plane in the current frame and corresponding values of the previous input frame. Expressed in range of [0-255].

VDIF

Display the average of sample value difference between all values of the V plane in the current frame and corresponding values of the previous input frame. Expressed in range of [0-255].

The filter accepts the following options:

stat



out

stat specify an additional form of image analysis. out output video with the specified type of pixel highlighted.

Both options accept the following values:

‘tout’

Identify *temporal outliers* pixels. A *temporal outlier* is a pixel unlike the neighboring pixels of the same field. Examples of temporal outliers include the results of video dropouts, head clogs, or tape tracking issues.

‘vrep’

Identify *vertical line repetition*. Vertical line repetition includes similar rows of pixels within a frame. In born-digital video vertical line repetition is common, but this pattern is uncommon in video digitized from an analog source. When it occurs in video that results from the digitization of an analog source it can indicate concealment from a dropout compensator.

‘brng’

Identify pixels that fall outside of legal broadcast range.

color, c

Set the highlight color for the out option. The default color is yellow.

### 37.107.1 Examples# TOC

- Output data of various video metrics:

```
ffprobe -f lavfi movie=example.mov,signalstats="stat=tout+vrep+brng" -show_frames
```

- Output specific data about the minimum and maximum values of the Y plane per frame:

```
ffprobe -f lavfi movie=example.mov,signalstats -show_entries frame_tags=lavfi.signalstats.YMAX,lavfi.signalstats.YMIN
```

- Playback video while highlighting pixels that are outside of broadcast range in red.

```
ffplay example.mov -vf signalstats="out=brng:color=red"
```

- Playback video with signalstats metadata drawn over the frame.

```
ffplay example.mov -vf signalstats=stat=brng+vrep+tout,drawtext=fontfile=FreeSerif.ttf:textfile=signalstat_drawtext.txt
```

The contents of signalstat\_drawtext.txt used in the command are:

```
time %{pts:hms}  
Y (%{metadata:lavfi.signalstats.YMIN}-{metadata:lavfi.signalstats.YMAX})  
U (%{metadata:lavfi.signalstats.UMIN}-{metadata:lavfi.signalstats.UMAX})  
V (%{metadata:lavfi.signalstats.VMIN}-{metadata:lavfi.signalstats.VMAX})  
saturation maximum: %{metadata:lavfi.signalstats.SATMAX}
```

## 37.108 smartblur# TOC

Blur the input video without impacting the outlines.

It accepts the following options:

`luma_radius, lr`

Set the luma radius. The option value must be a float number in the range [0.1,5.0] that specifies the variance of the gaussian filter used to blur the image (slower if larger). Default value is 1.0.

`luma_strength, ls`

Set the luma strength. The option value must be a float number in the range [-1.0,1.0] that configures the blurring. A value included in [0.0,1.0] will blur the image whereas a value included in [-1.0,0.0] will sharpen the image. Default value is 1.0.

`luma_threshold, lt`

Set the luma threshold used as a coefficient to determine whether a pixel should be blurred or not. The option value must be an integer in the range [-30,30]. A value of 0 will filter all the image, a value included in [0,30] will filter flat areas and a value included in [-30,0] will filter edges. Default value is 0.

`chroma_radius, cr`

Set the chroma radius. The option value must be a float number in the range [0.1,5.0] that specifies the variance of the gaussian filter used to blur the image (slower if larger). Default value is 1.0.

`chroma_strength, cs`

Set the chroma strength. The option value must be a float number in the range [-1.0,1.0] that configures the blurring. A value included in [0.0,1.0] will blur the image whereas a value included in [-1.0,0.0] will sharpen the image. Default value is 1.0.

`chroma_threshold, ct`

Set the chroma threshold used as a coefficient to determine whether a pixel should be blurred or not. The option value must be an integer in the range [-30,30]. A value of 0 will filter all the image, a value included in [0,30] will filter flat areas and a value included in [-30,0] will filter edges. Default value is 0.

If a chroma option is not explicitly set, the corresponding luma value is set.

## 37.109 ssim# TOC

Obtain the SSIM (Structural SIMilarity Metric) between two input videos.

This filter takes in input two input videos, the first input is considered the "main" source and is passed unchanged to the output. The second input is used as a "reference" video for computing the SSIM.

Both video inputs must have the same resolution and pixel format for this filter to work correctly. Also it assumes that both inputs have the same number of frames, which are compared one by one.

The filter stores the calculated SSIM of each frame.

The description of the accepted parameters follows.

`stats_file, f`

If specified the filter will use the named file to save the SSIM of each individual frame.

The file printed if *stats\_file* is selected, contains a sequence of key/value pairs of the form *key:value* for each compared couple of frames.

A description of each shown parameter follows:

`n`

sequential number of the input frame, starting from 1

`Y, U, V, R, G, B`

SSIM of the compared frames for the component specified by the suffix.

`All`

SSIM of the compared frames for the whole frame.

`dB`

Same as above but in dB representation.

For example:

```
movie=ref_movie.mpg, setpts=PTS-STARTPTS [main];  
[main][ref] ssim="stats_file=stats.log" [out]
```

On this example the input file being processed is compared with the reference file `ref_movie.mpg`. The SSIM of each individual frame is stored in `stats.log`.

Another example with both psnr and ssim at same time:

```
ffmpeg -i main.mpg -i ref.mpg -lavfi "ssim:[0:v][1:v]psnr" -f null -
```

## 37.110 stereo3d# TOC

Convert between different stereoscopic image formats.

The filters accept the following options:

in

Set stereoscopic image format of input.

Available values for input image formats are:

‘sbsl’

side by side parallel (left eye left, right eye right)

‘sbsr’

side by side crosseye (right eye left, left eye right)

‘sbs2l’

side by side parallel with half width resolution (left eye left, right eye right)

‘sbs2r’

side by side crosseye with half width resolution (right eye left, left eye right)

‘abl’

above-below (left eye above, right eye below)

‘abr’

above-below (right eye above, left eye below)

‘ab2l’

above-below with half height resolution (left eye above, right eye below)

‘ab2r’

above-below with half height resolution (right eye above, left eye below)

‘al’

alternating frames (left eye first, right eye second)

‘ar’

alternating frames (right eye first, left eye second)

Default value is ‘sbsl’.

out

Set stereoscopic image format of output.

Available values for output image formats are all the input formats as well as:

‘arbg’

anaglyph red/blue gray (red filter on left eye, blue filter on right eye)

‘argg’

anaglyph red/green gray (red filter on left eye, green filter on right eye)

‘arcg’

anaglyph red/cyan gray (red filter on left eye, cyan filter on right eye)

‘arch’

anaglyph red/cyan half colored (red filter on left eye, cyan filter on right eye)

‘arcc’

anaglyph red/cyan color (red filter on left eye, cyan filter on right eye)

‘arcd’

anaglyph red/cyan color optimized with the least squares projection of dubois (red filter on left eye, cyan filter on right eye)

‘agmg’

anaglyph green/magenta gray (green filter on left eye, magenta filter on right eye)

‘agmh’

anaglyph green/magenta half colored (green filter on left eye, magenta filter on right eye)

‘agmc’

anaglyph green/magenta colored (green filter on left eye, magenta filter on right eye)

‘agmd’

anaglyph green/magenta color optimized with the least squares projection of dubois (green filter on left eye, magenta filter on right eye)

‘aybg’

anaglyph yellow/blue gray (yellow filter on left eye, blue filter on right eye)

‘aybh’

anaglyph yellow/blue half colored (yellow filter on left eye, blue filter on right eye)

‘aybc’

anaglyph yellow/blue colored (yellow filter on left eye, blue filter on right eye)

‘aybd’

anaglyph yellow/blue color optimized with the least squares projection of dubois (yellow filter on left eye, blue filter on right eye)

‘irl’

interleaved rows (left eye has top row, right eye starts on next row)

‘irr’

interleaved rows (right eye has top row, left eye starts on next row)

‘ml’

mono output (left eye only)

‘mr’

mono output (right eye only)

Default value is ‘arcd’.

### 37.110.1 Examples# TOC

- Convert input video from side by side parallel to anaglyph yellow/blue dubois:

```
stereo3d=sbsl:aybd
```

- Convert input video from above below (left eye above, right eye below) to side by side crosseye.

```
stereo3d=abl:sbsr
```

## 37.111 spp# TOC

Apply a simple postprocessing filter that compresses and decompresses the image at several (or - in the case of `quality` level 6 - all) shifts and average the results.

The filter accepts the following options:

`quality`

Set quality. This option defines the number of levels for averaging. It accepts an integer in the range 0-6. If set to 0, the filter will have no effect. A value of 6 means the higher quality. For each increment of that value the speed drops by a factor of approximately 2. Default value is 3.

`qp`

Force a constant quantization parameter. If not set, the filter will use the QP from the video stream (if available).

`mode`

Set thresholding mode. Available modes are:

`'hard'`

Set hard thresholding (default).

`'soft'`

Set soft thresholding (better de-ringing effect, but likely blurrier).

`use_bframe_qp`

Enable the use of the QP from the B-Frames if set to 1. Using this option may cause flicker since the B-Frames have often larger QP. Default is 0 (not enabled).

## 37.112 subtitles# TOC

Draw subtitles on top of input video using the libass library.

To enable compilation of this filter you need to configure FFmpeg with `--enable-libass`. This filter also requires a build with libavcodec and libavformat to convert the passed subtitles file to ASS (Advanced Substation Alpha) subtitles format.

The filter accepts the following options:

`filename, f`

Set the filename of the subtitle file to read. It must be specified.

`original_size`

Specify the size of the original video, the video for which the ASS file was composed. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Due to a misdesign in ASS aspect ratio arithmetic, this is necessary to correctly scale the fonts if the aspect ratio has been changed.

`fontsdir`

Set a directory path containing fonts that can be used by the filter. These fonts will be used in addition to whatever the font provider uses.

`charenc`

Set subtitles input character encoding. `subtitles` filter only. Only useful if not UTF-8.

`stream_index, si`

Set subtitles stream index. `subtitles` filter only.

`force_style`

Override default style or script info parameters of the subtitles. It accepts a string containing ASS style format `KEY=VALUE` couples separated by `,`.

If the first key is not specified, it is assumed that the first value specifies the `filename`.

For example, to render the file `sub.srt` on top of the input video, use the command:

```
subtitles=sub.srt
```

which is equivalent to:

```
subtitles=filename=sub.srt
```

To render the default subtitles stream from file `video.mkv`, use:

```
subtitles=video.mkv
```

To render the second subtitles stream from that file, use:

```
subtitles=video.mkv:si=1
```



To make the subtitles stream from `sub.srt` appear in transparent green DeJaVu Serif, use:

```
subtitles=sub.srt:force_style='FontName=DeJaVu Serif,PrimaryColour=&HAA00FF00'
```

### 37.113 super2xsai# TOC

Scale the input by 2x and smooth using the Super2xSaI (Scale and Interpolate) pixel art scaling algorithm.

Useful for enlarging pixel art images without reducing sharpness.

### 37.114 swapuv# TOC

Swap U & V plane.

### 37.115 telecine# TOC

Apply telecine process to the video.

This filter accepts the following options:

```
first_field
    'top, t'
```

top field first

```
    'bottom, b'
```

bottom field first The default value is top.

```
pattern
```

A string of numbers representing the pulldown pattern you wish to apply. The default value is 23.

Some typical patterns:

NTSC output (30i):

27.5p: 32222

24p: 23 (classic)

24p: 2332 (preferred)

20p: 33

18p: 334

16p: 3444

PAL output (25i):

27.5p: 12222

24p: 222222222223 ("Euro pulldown")

16.67p: 33

16p: 33333334

## 37.116 thumbnail# TOC

Select the most representative frame in a given sequence of consecutive frames.

The filter accepts the following options:

`n`

Set the frames batch size to analyze; in a set of  $n$  frames, the filter will pick one of them, and then handle the next batch of  $n$  frames until the end. Default is 100.

Since the filter keeps track of the whole frames sequence, a bigger  $n$  value will result in a higher memory usage, so a high value is not recommended.

### 37.116.1 Examples# TOC

- Extract one picture each 50 frames:

```
thumbnail=50
```

- Complete example of a thumbnail creation with `ffmpeg`:

```
ffmpeg -i in.avi -vf thumbnail,scale=300:200 -frames:v 1 out.png
```

## 37.117 tile# TOC

Tile several successive frames together.

The filter accepts the following options:

`layout`

Set the grid size (i.e. the number of lines and columns). For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

`nb_frames`

Set the maximum number of frames to render in the given area. It must be less than or equal to  $w \times h$ . The default value is 0, meaning all the area will be used.

`margin`

Set the outer border margin in pixels.

`padding`

Set the inner border thickness (i.e. the number of pixels between frames). For more advanced padding options (such as having different values for the edges), refer to the `pad` video filter.

color

Specify the color of the unused area. For the syntax of this option, check the "Color" section in the `ffmpeg-utils` manual. The default value of `color` is "black".

### 37.117.1 Examples# TOC

- Produce 8x8 PNG tiles of all keyframes (`-skip_frame nokey`) in a movie:

```
ffmpeg -skip_frame nokey -i file.avi -vf 'scale=128:72,tile=8x8' -an -vsync 0 keyframes%03d.png
```

The `-vsync 0` is necessary to prevent `ffmpeg` from duplicating each output frame to accommodate the originally detected frame rate.

- Display 5 pictures in an area of 3x2 frames, with 7 pixels between them, and 2 pixels of initial margin, using mixed flat and named options:

```
tile=3x2:nb_frames=5:padding=7:margin=2
```

### 37.118 tinterlace# TOC

Perform various types of temporal field interlacing.

Frames are counted starting from 1, so the first input frame is considered odd.

The filter accepts the following options:

mode

Specify the mode of the interlacing. This option can also be specified as a value alone. See below for a list of values for this option.

Available values are:

'merge, 0'

Move odd frames into the upper field, even into the lower field, generating a double height frame at half frame rate.

```
-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444

Output:
11111        33333
22222        44444
11111        33333
```

22222	44444
11111	33333
22222	44444
11111	33333
22222	44444

‘drop\_odd, 1’

Only output even frames, odd frames are dropped, generating a frame with unchanged height at half frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444

Output:
                22222                44444
                22222                44444
                22222                44444
                22222                44444

```

‘drop\_even, 2’

Only output odd frames, even frames are dropped, generating a frame with unchanged height at half frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444
11111        22222        33333        44444

Output:
11111                33333
11111                33333
11111                33333
11111                33333

```

‘pad, 3’

Expand each frame to full height, but pad alternate lines with black, generating a frame with double height at the same input frame rate.

```

-----> time
Input:
Frame 1      Frame 2      Frame 3      Frame 4

```

11111	22222	33333	44444
11111	22222	33333	44444
11111	22222	33333	44444
11111	22222	33333	44444

Output:

11111	.....	33333	.....
.....	22222	.....	44444
11111	.....	33333	.....
.....	22222	.....	44444
11111	.....	33333	.....
.....	22222	.....	44444
11111	.....	33333	.....
.....	22222	.....	44444

‘interleave\_top, 4’

Interleave the upper field from odd frames with the lower field from even frames, generating a frame with unchanged height at half frame rate.

-----> time

Input:

Frame 1	Frame 2	Frame 3	Frame 4
11111<-	22222	33333<-	44444
11111	22222<-	33333	44444<-
11111<-	22222	33333<-	44444
11111	22222<-	33333	44444<-

Output:

11111	33333
22222	44444
11111	33333
22222	44444

‘interleave\_bottom, 5’

Interleave the lower field from odd frames with the upper field from even frames, generating a frame with unchanged height at half frame rate.

-----> time

Input:

Frame 1	Frame 2	Frame 3	Frame 4
11111	22222<-	33333	44444<-
11111<-	22222	33333<-	44444
11111	22222<-	33333	44444<-
11111<-	22222	33333<-	44444

Output:

22222	44444
11111	33333
22222	44444
11111	33333

`'interlacex2, 6'`

Double frame rate with unchanged height. Frames are inserted each containing the second temporal field from the previous input frame and the first temporal field from the next input frame. This mode relies on the `top_field_first` flag. Useful for interlaced video displays with no field synchronisation.

```
-----> time
Input:
Frame 1          Frame 2          Frame 3          Frame 4

11111           22222           33333           44444
 11111           22222           33333           44444
11111           22222           33333           44444
 11111           22222           33333           44444

Output:
11111  22222  22222  33333  33333  44444  44444
 11111  11111  22222  22222  33333  33333  44444
11111  22222  22222  33333  33333  44444  44444
 11111  11111  22222  22222  33333  33333  44444
```

Numeric values are deprecated but are accepted for backward compatibility reasons.

Default mode is merge.

flags

Specify flags influencing the filter process.

Available value for *flags* is:

`low_pass_filter, vlfp`

Enable vertical low-pass filtering in the filter. Vertical low-pass filtering is required when creating an interlaced destination from a progressive source which contains high-frequency vertical detail. Filtering will reduce interlace 'twitter' and Moire patterning.

Vertical low-pass filtering can only be enabled for mode *interleave\_top* and *interleave\_bottom*.

## 37.119 transpose# TOC

Transpose rows with columns in the input video and optionally flip it.

It accepts the following parameters:

`dir`

Specify the transposition direction.

Can assume the following values:

`'0, 4, cclock_flip'`

Rotate by 90 degrees counterclockwise and vertically flip (default), that is:

L.R		L.l
. .	->	. .
l.r		R.r

`'1, 5, clock'`

Rotate by 90 degrees clockwise, that is:

L.R		l.L
. .	->	. .
l.r		r.R

`'2, 6, cclock'`

Rotate by 90 degrees counterclockwise, that is:

L.R		R.r
. .	->	. .
l.r		L.l

`'3, 7, clock_flip'`

Rotate by 90 degrees clockwise and vertically flip, that is:

L.R		r.R
. .	->	. .
l.r		l.L

For values between 4-7, the transposition is only done if the input video geometry is portrait and not landscape. These values are deprecated, the `passthrough` option should be used instead.

Numerical values are deprecated, and should be dropped in favor of symbolic constants.

`passthrough`

Do not apply the transposition if the input geometry matches the one specified by the specified value. It accepts the following values:

`'none'`

Always apply transposition.

`'portrait'`

Preserve portrait geometry (when *height*  $\geq$  *width*).

‘landscape’

Preserve landscape geometry (when *width*  $\geq$  *height*).

Default value is none.

For example to rotate by 90 degrees clockwise and preserve portrait layout:

```
transpose=dir=1:passthrough=portrait
```

The command above can also be specified as:

```
transpose=1:portrait
```

## 37.120 trim# TOC

Trim the input so that the output contains one continuous subpart of the input.

It accepts the following parameters:

*start*

Specify the time of the start of the kept section, i.e. the frame with the timestamp *start* will be the first frame in the output.

*end*

Specify the time of the first frame that will be dropped, i.e. the frame immediately preceding the one with the timestamp *end* will be the last frame in the output.

*start\_pts*

This is the same as *start*, except this option sets the start timestamp in timebase units instead of seconds.

*end\_pts*

This is the same as *end*, except this option sets the end timestamp in timebase units instead of seconds.

*duration*

The maximum duration of the output in seconds.

*start\_frame*



The number of the first frame that should be passed to the output.

`end_frame`

The number of the first frame that should be dropped.

`start`, `end`, and `duration` are expressed as time duration specifications; see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax.

Note that the first two sets of the `start/end` options and the `duration` option look at the frame timestamp, while the `_frame` variants simply count the frames that pass through the filter. Also note that this filter does not modify the timestamps. If you wish for the output timestamps to start at zero, insert a `setpts` filter after the trim filter.

If multiple `start` or `end` options are set, this filter tries to be greedy and keep all the frames that match at least one of the specified constraints. To keep only the part that matches all the constraints at once, chain multiple trim filters.

The defaults are such that all the input is kept. So it is possible to set e.g. just the end values to keep everything before the specified time.

Examples:

- Drop everything except the second minute of input:

```
ffmpeg -i INPUT -vf trim=60:120
```

- Keep only the first second:

```
ffmpeg -i INPUT -vf trim=duration=1
```

## 37.121 unsharp# TOC

Sharpen or blur the input video.

It accepts the following parameters:

`luma_msize_x`, `lx`

Set the luma matrix horizontal size. It must be an odd integer between 3 and 63. The default value is 5.

`luma_msize_y`, `ly`

Set the luma matrix vertical size. It must be an odd integer between 3 and 63. The default value is 5.

`luma_amount`, `la`

Set the luma effect strength. It must be a floating point number, reasonable values lay between -1.5 and 1.5.

Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

Default value is 1.0.

`chroma_msize_x, cx`

Set the chroma matrix horizontal size. It must be an odd integer between 3 and 63. The default value is 5.

`chroma_msize_y, cy`

Set the chroma matrix vertical size. It must be an odd integer between 3 and 63. The default value is 5.

`chroma_amount, ca`

Set the chroma effect strength. It must be a floating point number, reasonable values lay between -1.5 and 1.5.

Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

Default value is 0.0.

`opencl`

If set to 1, specify using OpenCL capabilities, only available if FFmpeg was configured with `--enable-opencl`. Default value is 0.

All parameters are optional and default to the equivalent of the string '5:5:1.0:5:5:0.0'.

### 37.121.1 Examples# TOC

- Apply strong luma sharpen effect:

```
unsharp=luma_msize_x=7:luma_msize_y=7:luma_amount=2.5
```

- Apply a strong blur of both luma and chroma parameters:

```
unsharp=7:7:-2:7:7:-2
```

## 37.122 uspp# TOC

Apply ultra slow/simple postprocessing filter that compresses and decompresses the image at several (or - in the case of `quality` level 8 - all) shifts and average the results.

The way this differs from the behavior of `spp` is that `uspp` actually encodes & decodes each case with libavcodec Snow, whereas `spp` uses a simplified intra only 8x8 DCT similar to MJPEG.

The filter accepts the following options:

`quality`

Set quality. This option defines the number of levels for averaging. It accepts an integer in the range 0-8. If set to 0, the filter will have no effect. A value of 8 means the higher quality. For each increment of that value the speed drops by a factor of approximately 2. Default value is 3.

`qp`

Force a constant quantization parameter. If not set, the filter will use the QP from the video stream (if available).

## 37.123 vectorscope# TOC

Display 2 color component values in the two dimensional graph (which is called a vectorscope).

This filter accepts the following options:

`mode , m`

Set vectorscope mode.

It accepts the following values:

`'gray'`

Gray values are displayed on graph, higher brightness means more pixels have same component color value on location in graph. This is the default mode.

`'color'`

Gray values are displayed on graph. Surrounding pixels values which are not present in video frame are drawn in gradient of 2 color components which are set by option `x` and `y`.

`'color2'`

Actual color components values present in video frame are displayed on graph.

`'color3'`

Similar as color2 but higher frequency of same values x and y on graph increases value of another color component, which is luminance by default values of x and y.

`'color4'`

Actual colors present in video frame are displayed on graph. If two different colors map to same position on graph then color with higher value of component not present in graph is picked.

x

Set which color component will be represented on X-axis. Default is 1.

y

Set which color component will be represented on Y-axis. Default is 2.

intensity, i

Set intensity, used by modes: gray, color and color3 for increasing brightness of color component which represents frequency of (X, Y) location in graph.

envelope, e

`'none'`

No envelope, this is default.

`'instant'`

Instant envelope, even darkest single pixel will be clearly highlighted.

`'peak'`

Hold maximum and minimum values presented in graph over time. This way you can still spot out of range values without constantly looking at vectorscope.

`'peak+instant'`

Peak and instant envelope combined together.

## 37.124 vidstabdetect# TOC

Analyze video stabilization/deshaking. Perform pass 1 of 2, see vidstabtransform for pass 2.

This filter generates a file with relative translation and rotation transform information about subsequent frames, which is then used by the vidstabtransform filter.

To enable compilation of this filter you need to configure FFmpeg with `--enable-libvidstab`.

This filter accepts the following options:

`result`

Set the path to the file used to write the transforms information. Default value is `transforms.trf`.

`shakiness`

Set how shaky the video is and how quick the camera is. It accepts an integer in the range 1-10, a value of 1 means little shakiness, a value of 10 means strong shakiness. Default value is 5.

`accuracy`

Set the accuracy of the detection process. It must be a value in the range 1-15. A value of 1 means low accuracy, a value of 15 means high accuracy. Default value is 15.

`stepsize`

Set stepsize of the search process. The region around minimum is scanned with 1 pixel resolution. Default value is 6.

`mincontrast`

Set minimum contrast. Below this value a local measurement field is discarded. Must be a floating point value in the range 0-1. Default value is 0.3.

`tripod`

Set reference frame number for tripod mode.

If enabled, the motion of the frames is compared to a reference frame in the filtered stream, identified by the specified number. The idea is to compensate all movements in a more-or-less static scene and keep the camera view absolutely still.

If set to 0, it is disabled. The frames are counted starting from 1.

`show`

Show fields and transforms in the resulting frames. It accepts an integer in the range 0-2. Default value is 0, which disables any visualization.

### 37.124.1 Examples# TOC

- Use default values:

`vidstabdetect`

- Analyze strongly shaky movie and put the results in file `mytransforms.trf`:

```
vidstabdetect=shakiness=10:accuracy=15:result="mytransforms.trf"
```

- Visualize the result of internal transformations in the resulting video:

```
vidstabdetect=show=1
```

- Analyze a video with medium shakiness using `ffmpeg`:

```
ffmpeg -i input -vf vidstabdetect=shakiness=5:show=1 dummy.avi
```

## 37.125 vidstabtransform# TOC

Video stabilization/deshaking: pass 2 of 2, see `vidstabdetect` for pass 1.

Read a file with transform information for each frame and apply/compensate them. Together with the `vidstabdetect` filter this can be used to deshake videos. See also <http://public.hronopik.de/vid.stab>. It is important to also use the `unsharp` filter, see below.

To enable compilation of this filter you need to configure FFmpeg with `--enable-libvidstab`.

### 37.125.1 Options# TOC

`input`

Set path to the file used to read the transforms. Default value is `transforms.trf`.

`smoothing`

Set the number of frames ( $\text{value} \times 2 + 1$ ) used for lowpass filtering the camera movements. Default value is 10.

For example a number of 10 means that 21 frames are used (10 in the past and 10 in the future) to smoothen the motion in the video. A larger value leads to a smoother video, but limits the acceleration of the camera (pan/tilt movements). 0 is a special case where a static camera is simulated.

`optalgo`

Set the camera path optimization algorithm.

Accepted values are:

`'gauss'`

gaussian kernel low-pass filter on camera motion (default)

‘avg’

averaging on transformations

maxshift

Set maximal number of pixels to translate frames. Default value is -1, meaning no limit.

maxangle

Set maximal angle in radians ( $\text{degree} \cdot \pi / 180$ ) to rotate frames. Default value is -1, meaning no limit.

crop

Specify how to deal with borders that may be visible due to movement compensation.

Available values are:

‘keep’

keep image information from previous frame (default)

‘black’

fill the border black

invert

Invert transforms if set to 1. Default value is 0.

relative

Consider transforms as relative to previous frame if set to 1, absolute if set to 0. Default value is 0.

zoom

Set percentage to zoom. A positive value will result in a zoom-in effect, a negative value in a zoom-out effect. Default value is 0 (no zoom).

optzoom

Set optimal zooming to avoid borders.

Accepted values are:

‘0’

disabled

‘1’

optimal static zoom value is determined (only very strong movements will lead to visible borders) (default)

‘2’

optimal adaptive zoom value is determined (no borders will be visible), see zoomspeed

Note that the value given at zoom is added to the one calculated here.

zoomspeed

Set percent to zoom maximally each frame (enabled when optzoom is set to 2). Range is from 0 to 5, default value is 0.25.

interpol

Specify type of interpolation.

Available values are:

‘no’

no interpolation

‘linear’

linear only horizontal

‘bilinear’

linear in both directions (default)

‘bicubic’

cubic in both directions (slow)

tripod

Enable virtual tripod mode if set to 1, which is equivalent to relative=0:smoothing=0. Default value is 0.

Use also tripod option of vidstabdetect.

debug



Increase log verbosity if set to 1. Also the detected global motions are written to the temporary file `global_motions.trf`. Default value is 0.

### 37.125.2 Examples# TOC

- Use `ffmpeg` for a typical stabilization with default values:

```
ffmpeg -i inp.mpeg -vf vidstabtransform,unsharp=5:5:0.8:3:3:0.4 inp_stabilized.mpeg
```

Note the use of the `unsharp` filter which is always recommended.

- Zoom in a bit more and load transform data from a given file:

```
vidstabtransform=zoom=5:input="mytransforms.trf"
```

- Smoothen the video even more:

```
vidstabtransform=smoothing=30
```

### 37.126 vflip# TOC

Flip the input video vertically.

For example, to vertically flip a video with `ffmpeg`:

```
ffmpeg -i in.avi -vf "vflip" out.avi
```

### 37.127 vignette# TOC

Make or reverse a natural vignetting effect.

The filter accepts the following options:

`angle`, `a`

Set lens angle expression as a number of radians.

The value is clipped in the  $[0, \pi/2]$  range.

Default value: `" $\pi/5$ "`

`x0`

`y0`

Set center coordinates expressions. Respectively `"w/2"` and `"h/2"` by default.

`mode`

Set forward/backward mode.

Available modes are:

‘forward’

The larger the distance from the central point, the darker the image becomes.

‘backward’

The larger the distance from the central point, the brighter the image becomes. This can be used to reverse a vignette effect, though there is no automatic detection to extract the lens angle and other settings (yet). It can also be used to create a burning effect.

Default value is ‘forward’.

eval

Set evaluation mode for the expressions (angle, x0, y0).

It accepts the following values:

‘init’

Evaluate expressions only once during the filter initialization.

‘frame’

Evaluate expressions for each incoming frame. This is way slower than the ‘init’ mode since it requires all the scalars to be re-computed, but it allows advanced dynamic expressions.

Default value is ‘init’.

dither

Set dithering to reduce the circular banding effects. Default is 1 (enabled).

aspect

Set vignette aspect. This setting allows one to adjust the shape of the vignette. Setting this value to the SAR of the input will make a rectangular vignetting following the dimensions of the video.

Default is 1/1.

### **37.127.1 Expressions# TOC**

The alpha, x0 and y0 expressions can contain the following parameters.

w  
h

input width and height

n

the number of input frame, starting from 0

pts

the PTS (Presentation TimeStamp) time of the filtered video frame, expressed in *TB* units, NAN if undefined

r

frame rate of the input video, NAN if the input frame rate is unknown

t

the PTS (Presentation TimeStamp) of the filtered video frame, expressed in seconds, NAN if undefined

tb

time base of the input video

### 37.127.2 Examples# TOC

- Apply simple strong vignetting effect:

```
vignette=PI/4
```

- Make a flickering vignetting:

```
vignette='PI/4+random(1)*PI/50':eval=frame
```

### 37.128 vstack# TOC

Stack input videos vertically.

All streams must be of same pixel format and of same width.

Note that this filter is faster than using overlay and pad filter to create same output.

The filter accept the following option:

nb\_inputs

Set number of input streams. Default is 2.

## 37.129 w3fdif# TOC

Deinterlace the input video ("w3fdif" stands for "Weston 3 Field Deinterlacing Filter").

Based on the process described by Martin Weston for BBC R&D, and implemented based on the de-interlace algorithm written by Jim Easterbrook for BBC R&D, the Weston 3 field deinterlacing filter uses filter coefficients calculated by BBC R&D.

There are two sets of filter coefficients, so called "simple": and "complex". Which set of filter coefficients is used can be set by passing an optional parameter:

`filter`

Set the interlacing filter coefficients. Accepts one of the following values:

`'simple'`

Simple filter coefficient set.

`'complex'`

More-complex filter coefficient set.

Default value is `'complex'`.

`deint`

Specify which frames to deinterlace. Accept one of the following values:

`'all'`

Deinterlace all frames,

`'interlaced'`

Only deinterlace frames marked as interlaced.

Default value is `'all'`.

## 37.130 waveform# TOC

Video waveform monitor.

The waveform monitor plots color component intensity. By default luminance only. Each column of the waveform corresponds to a column of pixels in the source video.

It accepts the following options:

`mode, m`

Can be either `row`, or `column`. Default is `column`. In row mode, the graph on the left side represents color component value 0 and the right side represents value = 255. In column mode, the top side represents color component value = 0 and bottom side represents value = 255.

`intensity, i`

Set intensity. Smaller values are useful to find out how many values of the same luminance are distributed across input rows/columns. Default value is 0.04. Allowed range is [0, 1].

`mirror, r`

Set mirroring mode. 0 means unmirrored, 1 means mirrored. In mirrored mode, higher values will be represented on the left side for `row` mode and at the top for `column` mode. Default is 1 (mirrored).

`display, d`

Set display mode. It accepts the following values:

`'overlay'`

Presents information identical to that in the `parade`, except that the graphs representing color components are superimposed directly over one another.

This display mode makes it easier to spot relative differences or similarities in overlapping areas of the color components that are supposed to be identical, such as neutral whites, grays, or blacks.

`'parade'`

Display separate graph for the color components side by side in `row` mode or one below the other in `column` mode.

Using this display mode makes it easy to spot color casts in the highlights and shadows of an image, by comparing the contours of the top and the bottom graphs of each waveform. Since whites, grays, and blacks are characterized by exactly equal amounts of red, green, and blue, neutral areas of the picture should display three waveforms of roughly equal width/height. If not, the correction is easy to perform by making level adjustments the three waveforms.

Default is `parade`.

`components, c`

Set which color components to display. Default is 1, which means only luminance or red color component if input is in RGB colorspace. If is set for example to 7 it will display all 3 (if) available color components.

envelope, e  
    'none'

No envelope, this is default.

    'instant'

Instant envelope, minimum and maximum values presented in graph will be easily visible even with small step value.

    'peak'

Hold minimum and maximum values presented in graph across time. This way you can still spot out of range values without constantly looking at waveforms.

    'peak+instant'

Peak and instant envelope combined together.

filter, f  
    'lowpass'

No filtering, this is default.

    'flat'

Luma and chroma combined together.

    'aflat'

Similar as above, but shows difference between blue and red chroma.

    'chroma'

Displays only chroma.

    'achroma'

Similar as above, but shows difference between blue and red chroma.

    'color'

Displays actual color value on waveform.

## 37.131 xbr# TOC

Apply the xBR high-quality magnification filter which is designed for pixel art. It follows a set of edge-detection rules, see <http://www.libretro.com/forums/viewtopic.php?f=6&t=134>.

It accepts the following option:

`n`

Set the scaling dimension: 2 for 2xBR, 3 for 3xBR and 4 for 4xBR. Default is 3.

### **37.132 yadif# TOC**

Deinterlace the input video ("yadif" means "yet another deinterlacing filter").

It accepts the following parameters:

`mode`

The interlacing mode to adopt. It accepts one of the following values:

`0, send_frame`

Output one frame for each frame.

`1, send_field`

Output one frame for each field.

`2, send_frame_nospatial`

Like `send_frame`, but it skips the spatial interlacing check.

`3, send_field_nospatial`

Like `send_field`, but it skips the spatial interlacing check.

The default value is `send_frame`.

`parity`

The picture field parity assumed for the input interlaced video. It accepts one of the following values:

`0, tff`

Assume the top field is first.

`1, bff`

Assume the bottom field is first.

`-1, auto`

Enable automatic detection of field parity.

The default value is `auto`. If the interlacing is unknown or the decoder does not export this information, top field first will be assumed.

`deint`

Specify which frames to deinterlace. Accept one of the following values:

`0, all`

Deinterlace all frames.

`1, interlaced`

Only deinterlace frames marked as interlaced.

The default value is `all`.

### 37.133 zoompan# TOC

Apply Zoom & Pan effect.

This filter accepts the following options:

`zoom, z`

Set the zoom expression. Default is 1.

`x`

`y`

Set the x and y expression. Default is 0.

`d`

Set the duration expression in number of frames. This sets for how many number of frames effect will last for single input image.

`s`

Set the output image size, default is 'hd720'.

Each expression can contain the following constants:

`in_w, iw`



Input width.

in\_h, ih

Input height.

out\_w, ow

Output width.

out\_h, oh

Output height.

in

Input frame count.

on

Output frame count.

x

y

Last calculated 'x' and 'y' position from 'x' and 'y' expression for current input frame.

px

py

'x' and 'y' of last output frame of previous input frame or 0 when there was not yet such frame (first input frame).

zoom

Last calculated zoom from 'z' expression for current input frame.

pzoom

Last calculated zoom of last output frame of previous input frame.

duration

Number of output frames for current input frame. Calculated from 'd' expression for each input frame.

pduration

number of output frames created for previous input frame

a

Rational number: input width / input height

sar

sample aspect ratio

dar

display aspect ratio

### 37.133.1 Examples# TOC

- Zoom-in up to 1.5 and pan at same time to some spot near center of picture:

```
zoompan=z='min(zoom+0.0015,1.5)':d=700:x='if(gte(zoom,1.5),x,x+1/a)':y='if(gte(zoom,1.5),y,y+1)':s=640x360
```

- Zoom-in up to 1.5 and pan always at center of picture:

```
zoompan=z='min(zoom+0.0015,1.5)':d=700:x='iw/2-(iw/zoom/2)':y='ih/2-(ih/zoom/2)'
```

## 38 Video Sources# TOC

Below is a description of the currently available video sources.

### 38.1 buffer# TOC

Buffer video frames, and make them available to the filter chain.

This source is mainly intended for a programmatic use, in particular through the interface defined in `libavfilter/vsrc_buffer.h`.

It accepts the following parameters:

video\_size

Specify the size (width and height) of the buffered video frames. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

width

The input video width.

height

The input video height.

`pix_fmt`

A string representing the pixel format of the buffered video frames. It may be a number corresponding to a pixel format, or a pixel format name.

`time_base`

Specify the timebase assumed by the timestamps of the buffered frames.

`frame_rate`

Specify the frame rate expected for the video stream.

`pixel_aspect, sar`

The sample (pixel) aspect ratio of the input video.

`sws_param`

Specify the optional parameters to be used for the scale filter which is automatically inserted when an input change is detected in the input size or format.

For example:

```
buffer=width=320:height=240:pix_fmt=yuv410p:time_base=1/24:sar=1
```

will instruct the source to accept video frames with size 320x240 and with format "yuv410p", assuming 1/24 as the timestamps timebase and square pixels (1:1 sample aspect ratio). Since the pixel format with name "yuv410p" corresponds to the number 6 (check the enum `AVPixelFormat` definition in `libavutil/pixfmt.h`), this example corresponds to:

```
buffer=size=320x240:pixfmt=6:time_base=1/24:pixel_aspect=1/1
```

Alternatively, the options can be specified as a flat string, but this syntax is deprecated:

```
width:height:pix_fmt:time_base.num:time_base.den:pixel_aspect.num:pixel_aspect.den[:sws_param]
```

## 38.2 cellauto# TOC

Create a pattern generated by an elementary cellular automaton.

The initial state of the cellular automaton can be defined through the `filename`, and `pattern` options. If such options are not specified an initial state is created randomly.

At each new frame a new row in the video is filled with the result of the cellular automaton next generation. The behavior when the whole frame is filled is defined by the `scroll` option.

This source accepts the following options:

`filename, f`

Read the initial cellular automaton state, i.e. the starting row, from the specified file. In the file, each non-whitespace character is considered an alive cell, a newline will terminate the row, and further characters in the file will be ignored.

`pattern, p`

Read the initial cellular automaton state, i.e. the starting row, from the specified string.

Each non-whitespace character in the string is considered an alive cell, a newline will terminate the row, and further characters in the string will be ignored.

`rate, r`

Set the video rate, that is the number of frames generated per second. Default is 25.

`random_fill_ratio, ratio`

Set the random fill ratio for the initial cellular automaton row. It is a floating point number value ranging from 0 to 1, defaults to 1/PHI.

This option is ignored when a file or a pattern is specified.

`random_seed, seed`

Set the seed for filling randomly the initial row, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to -1, the filter will try to use a good random seed on a best effort basis.

`rule`

Set the cellular automaton rule, it is a number ranging from 0 to 255. Default value is 110.

`size, s`

Set the size of the output video. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

If `filename` or `pattern` is specified, the size is set by default to the width of the specified initial state row, and the height is set to *width* \* PHI.

If `size` is set, it must contain the width of the specified pattern string, and the specified pattern will be centered in the larger row.

If a filename or a pattern string is not specified, the size value defaults to "320x518" (used for a randomly generated initial state).

`scroll`

If set to 1, scroll the output upward when all the rows in the output have been already filled. If set to 0, the new generated row will be written over the top row just after the bottom row is filled. Defaults to 1.

`start_full, full`

If set to 1, completely fill the output with generated rows before outputting the first frame. This is the default behavior, for disabling set the value to 0.

`stitch`

If set to 1, stitch the left and right row edges together. This is the default behavior, for disabling set the value to 0.

### 38.2.1 Examples# TOC

- Read the initial state from `pattern`, and specify an output of size 200x400.

```
cellauto=f=pattern:s=200x400
```

- Generate a random initial row with a width of 200 cells, with a fill ratio of 2/3:

```
cellauto=ratio=2/3:s=200x200
```

- Create a pattern generated by rule 18 starting by a single alive cell centered on an initial row with width 100:

```
cellauto=p=@:s=100x400:full=0:rule=18
```

- Specify a more elaborated initial pattern:

```
cellauto=p='@@ @ @@':s=100x400:full=0:rule=18
```

### 38.3 mandelbrot# TOC

Generate a Mandelbrot set fractal, and progressively zoom towards the point specified with *start\_x* and *start\_y*.

This source accepts the following options:

`end_pts`

Set the terminal pts value. Default value is 400.

`end_scale`

Set the terminal scale value. Must be a floating point value. Default value is 0.3.

`inner`

Set the inner coloring mode, that is the algorithm used to draw the Mandelbrot fractal internal region.

It shall assume one of the following values:

`black`

Set black mode.

`convergence`

Show time until convergence.

`mincol`

Set color based on point closest to the origin of the iterations.

`period`

Set period mode.

Default value is *mincol*.

`bailout`

Set the bailout value. Default value is 10.0.

`maxiter`

Set the maximum of iterations performed by the rendering algorithm. Default value is 7189.

`outer`

Set outer coloring mode. It shall assume one of following values:

`iteration_count`

Set iteration count mode.

`normalized_iteration_count`

set normalized iteration count mode.

Default value is *normalized\_iteration\_count*.

`rate, r`

Set frame rate, expressed as number of frames per second. Default value is "25".

`size, s`

Set frame size. For the syntax of this option, check the "Video size" section in the ffmpeg-utils manual. Default value is "640x480".

`start_scale`

Set the initial scale value. Default value is 3.0.

`start_x`

Set the initial x position. Must be a floating point value between -100 and 100. Default value is -0.743643887037158704752191506114774.

`start_y`

Set the initial y position. Must be a floating point value between -100 and 100. Default value is -0.131825904205311970493132056385139.

## 38.4 mptestsrc# TOC

Generate various test patterns, as generated by the MPlayer test filter.

The size of the generated video is fixed, and is 256x256. This source is useful in particular for testing encoding features.

This source accepts the following options:

`rate, r`

Specify the frame rate of the sourced video, as the number of frames generated per second. It has to be a string in the format *frame\_rate\_num/frame\_rate\_den*, an integer number, a floating point number or a valid video frame rate abbreviation. The default value is "25".

`duration, d`

Set the duration of the sourced video. See (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual for the accepted syntax.

If not specified, or the expressed duration is negative, the video is supposed to be generated forever.

`test, t`

Set the number or the name of the test to perform. Supported tests are:

dc\_luma  
dc\_chroma  
freq\_luma  
freq\_chroma  
amp\_luma  
amp\_chroma  
cbp  
mv  
ring1  
ring2  
all

Default value is "all", which will cycle through the list of all tests.

Some examples:

```
mptestsrc=t=dc_luma
```

will generate a "dc\_luma" test pattern.

## 38.5 frei0r\_src# TOC

Provide a frei0r source.

To enable compilation of this filter you need to install the frei0r header and configure FFmpeg with `--enable-frei0r`.

This source accepts the following parameters:

`size`

The size of the video to generate. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

`framerate`

The framerate of the generated video. It may be a string of the form *num/den* or a frame rate abbreviation.

`filter_name`

The name to the frei0r source to load. For more information regarding frei0r and how to set the parameters, read the frei0r section in the video filters documentation.

`filter_params`



A `'`-separated list of parameters to pass to the `frei0r` source.

For example, to generate a `frei0r` `partik01` source with size 200x200 and frame rate 10 which is overlaid on the overlay filter main input:

```
frei0r_src=size=200x200:framerate=10:filter_name=partik01:filter_params=1234 [overlay]; [in][overlay] overlay
```

## 38.6 life# TOC

Generate a life pattern.

This source is based on a generalization of John Conway's life game.

The sourced input represents a life grid, each pixel represents a cell which can be in one of two possible states, alive or dead. Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent.

At each interaction the grid evolves according to the adopted rule, which specifies the number of neighbor alive cells which will make a cell stay alive or born. The `rule` option allows one to specify the rule to adopt.

This source accepts the following options:

`filename, f`

Set the file from which to read the initial grid state. In the file, each non-whitespace character is considered an alive cell, and newline is used to delimit the end of each row.

If this option is not specified, the initial grid is generated randomly.

`rate, r`

Set the video rate, that is the number of frames generated per second. Default is 25.

`random_fill_ratio, ratio`

Set the random fill ratio for the initial random grid. It is a floating point number value ranging from 0 to 1, defaults to 1/PHI. It is ignored when a file is specified.

`random_seed, seed`

Set the seed for filling the initial random grid, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to -1, the filter will try to use a good random seed on a best effort basis.

`rule`

Set the life rule.

A rule can be specified with a code of the kind "SNS/BNB", where *NS* and *NB* are sequences of numbers in the range 0-8, *NS* specifies the number of alive neighbor cells which make a live cell stay alive, and *NB* the number of alive neighbor cells which make a dead cell to become alive (i.e. to "born"). "s" and "b" can be used in place of "S" and "B", respectively.

Alternatively a rule can be specified by an 18-bits integer. The 9 high order bits are used to encode the next cell state if it is alive for each number of neighbor alive cells, the low order bits specify the rule for "born" new cells. Higher order bits encode for an higher number of neighbor cells. For example the number 6153 = ( 12<<9 ) +9 specifies a stay alive rule of 12 and a born rule of 9, which corresponds to "S23/B03".

Default value is "S23/B3", which is the original Conway's game of life rule, and will keep a cell alive if it has 2 or 3 neighbor alive cells, and will born a new cell if there are three alive cells around a dead cell.

size, s

Set the size of the output video. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual.

If filename is specified, the size is set by default to the same size of the input file. If size is set, it must contain the size specified in the input file, and the initial grid defined in that file is centered in the larger resulting area.

If a filename is not specified, the size value defaults to "320x240" (used for a randomly generated initial grid).

stitch

If set to 1, stitch the left and right grid edges together, and the top and bottom edges also. Defaults to 1.

mold

Set cell mold speed. If set, a dead cell will go from death\_color to mold\_color with a step of mold. mold can have a value from 0 to 255.

life\_color

Set the color of living (or new born) cells.

death\_color

Set the color of dead cells. If mold is set, this is the first color used to represent a dead cell.

mold\_color

Set mold color, for definitely dead and moldy cells.

For the syntax of these 3 color options, check the "Color" section in the ffmpeg-utils manual.

### 38.6.1 Examples# TOC

- Read a grid from `pattern`, and center it on a grid of size 300x300 pixels:

```
life=f=pattern:s=300x300
```

- Generate a random grid of size 200x200, with a fill ratio of 2/3:

```
life=ratio=2/3:s=200x200
```

- Specify a custom rule for evolving a randomly generated grid:

```
life=rule=S14/B34
```

- Full example with slow death effect (mold) using `ffplay`:

```
ffplay -f lavfi life=s=300x200:mold=10:r=60:ratio=0.1:death_color=#C83232:life_color=#00ff00,scale=1200:800:flags=16
```

## 38.7 allrgb, allyuv, color, haldclutsrc, nullsrc, rgbtestsrc, smptebars, smptehtbars, testsrc# TOC

The `allrgb` source returns frames of size 4096x4096 of all rgb colors.

The `allyuv` source returns frames of size 4096x4096 of all yuv colors.

The `color` source provides an uniformly colored input.

The `haldclutsrc` source provides an identity Hald CLUT. See also `haldclut` filter.

The `nullsrc` source returns unprocessed video frames. It is mainly useful to be employed in analysis / debugging tools, or as the source for filters which ignore the input data.

The `rgbtestsrc` source generates an RGB test pattern useful for detecting RGB vs BGR issues. You should see a red, green and blue stripe from top to bottom.

The `smptebars` source generates a color bars pattern, based on the SMPTE Engineering Guideline EG 1-1990.

The `smptehtbars` source generates a color bars pattern, based on the SMPTE RP 219-2002.

The `testsrc` source generates a test video pattern, showing a color pattern, a scrolling gradient and a timestamp. This is mainly intended for testing purposes.

The sources accept the following parameters:

`color, c`

Specify the color of the source, only available in the `color` source. For the syntax of this option, check the "Color" section in the `ffmpeg-utils` manual.

`level`

Specify the level of the Hald CLUT, only available in the `haldclutsrc` source. A level of  $N$  generates a picture of  $N*N*N$  by  $N*N*N$  pixels to be used as identity matrix for 3D lookup tables. Each component is coded on a  $1/(N*N)$  scale.

`size, s`

Specify the size of the sourced video. For the syntax of this option, check the (`ffmpeg-utils`)"Video size" section in the `ffmpeg-utils` manual. The default value is  $320 \times 240$ .

This option is not available with the `haldclutsrc` filter.

`rate, r`

Specify the frame rate of the sourced video, as the number of frames generated per second. It has to be a string in the format *frame\_rate\_num/frame\_rate\_den*, an integer number, a floating point number or a valid video frame rate abbreviation. The default value is "25".

`sar`

Set the sample aspect ratio of the sourced video.

`duration, d`

Set the duration of the sourced video. See (`ffmpeg-utils`)the Time duration section in the `ffmpeg-utils(1)` manual for the accepted syntax.

If not specified, or the expressed duration is negative, the video is supposed to be generated forever.

`decimals, n`

Set the number of decimals to show in the timestamp, only available in the `testsrc` source.

The displayed timestamp value will correspond to the original timestamp value multiplied by the power of 10 of the specified value. Default value is 0.

For example the following:

```
testsrc=duration=5.3:size=qcif:rate=10
```

will generate a video with a duration of 5.3 seconds, with size  $176 \times 144$  and a frame rate of 10 frames per second.

The following graph description will generate a red source with an opacity of 0.2, with size "qcif" and a frame rate of 10 frames per second.

```
color=c=red@0.2:s=qcif:r=10
```

If the input content is to be ignored, `nullsrc` can be used. The following command generates noise in the luminance plane by employing the `geq` filter:

```
nullsrc=s=256x256, geq=random(1)*255:128:128
```

### 38.7.1 Commands# TOC

The `color` source supports the following commands:

`c, color`

Set the color of the created image. Accepts the same syntax of the corresponding `color` option.

## 39 Video Sinks# TOC

Below is a description of the currently available video sinks.

### 39.1 buffersink# TOC

Buffer video frames, and make them available to the end of the filter graph.

This sink is mainly intended for programmatic use, in particular through the interface defined in `libavfilter/buffersink.h` or the options system.

It accepts a pointer to an `AVBufferSinkContext` structure, which defines the incoming buffers' formats, to be passed as the opaque parameter to `avfilter_init_filter` for initialization.

### 39.2 nullsink# TOC

Null video sink: do absolutely nothing with the input video. It is mainly useful as a template and for use in analysis / debugging tools.

## 40 Multimedia Filters# TOC

Below is a description of the currently available multimedia filters.

### 40.1 aphasemeter# TOC

Convert input audio to a video output, displaying the audio phase.

The filter accepts the following options:

`rate, r`

Set the output frame rate. Default value is 25.

`size, s`

Set the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 800×400.

`rc`

`gc`

`bc`

Specify the red, green, blue contrast. Default values are 2, 7 and 1. Allowed range is [ 0 , 255 ].

`mpc`

Set color which will be used for drawing median phase. If color is none which is default, no median phase value will be drawn.

The filter also exports the frame metadata `lavfi.aphasemeter.phase` which represents mean phase of current audio frame. Value is in range [ -1 , 1 ]. The -1 means left and right channels are completely out of phase and 1 means channels are in phase.

## 40.2 avectorscope# TOC

Convert input audio to a video output, representing the audio vector scope.

The filter is used to measure the difference between channels of stereo audio stream. A monoaural signal, consisting of identical left and right signal, results in straight vertical line. Any stereo separation is visible as a deviation from this line, creating a Lissajous figure. If the straight (or deviation from it) but horizontal line appears this indicates that the left and right channels are out of phase.

The filter accepts the following options:

`mode, m`

Set the vectorscope mode.

Available values are:

`'lissajous'`

Lissajous rotated by 45 degrees.

`'lissajous_xy'`

Same as above but not rotated.

`'polar'`

Shape resembling half of circle.

Default value is `'lissajous'`.

`size, s`

Set the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 400×400.

`rate, r`

Set the output frame rate. Default value is 25.

`rc`

`gc`

`bc`

`ac`

Specify the red, green, blue and alpha contrast. Default values are 40, 160, 80 and 255. Allowed range is [0, 255].

`rf`

`gf`

`bf`

`af`

Specify the red, green, blue and alpha fade. Default values are 15, 10, 5 and 5. Allowed range is [0, 255].

`zoom`

Set the zoom factor. Default value is 1. Allowed range is [1, 10].

### 40.2.1 Examples# TOC

- Complete example using `ffplay`:

```
ffplay -f lavfi 'amovie=input.mp3, asplit [a][out1];  
[a] avectorscope=zoom=1.3:rc=2:gc=200:bc=10:rf=1:gf=8:bf=7 [out0]'
```

## 40.3 concat# TOC

Concatenate audio and video streams, joining them together one after the other.

The filter works on segments of synchronized video and audio streams. All segments must have the same number of streams of each type, and that will also be the number of streams at output.

The filter accepts the following options:

`n`

Set the number of segments. Default is 2.

`v`

Set the number of output video streams, that is also the number of video streams in each segment. Default is 1.

`a`

Set the number of output audio streams, that is also the number of audio streams in each segment. Default is 0.

`unsafe`

Activate unsafe mode: do not fail if segments have a different format.

The filter has  $v+a$  outputs: first  $v$  video outputs, then  $a$  audio outputs.

There are  $nx(v+a)$  inputs: first the inputs for the first segment, in the same order as the outputs, then the inputs for the second segment, etc.

Related streams do not always have exactly the same duration, for various reasons including codec frame size or sloppy authoring. For that reason, related synchronized streams (e.g. a video and its audio track) should be concatenated at once. The concat filter will use the duration of the longest stream in each segment (except the last one), and if necessary pad shorter audio streams with silence.

For this filter to work correctly, all segments must start at timestamp 0.

All corresponding streams must have the same parameters in all segments; the filtering system will automatically select a common pixel format for video streams, and a common sample format, sample rate and channel layout for audio streams, but other settings, such as resolution, must be converted explicitly by the user.

Different frame rates are acceptable but will result in variable frame rate at output; be sure to configure the output file to handle it.



### 40.3.1 Examples# TOC

- Concatenate an opening, an episode and an ending, all in bilingual version (video in stream 0, audio in streams 1 and 2):

```
ffmpeg -i opening.mkv -i episode.mkv -i ending.mkv -filter_complex \
'[0:0] [0:1] [0:2] [1:0] [1:1] [1:2] [2:0] [2:1] [2:2]
concat=n=3:v=1:a=2 [v] [a1] [a2]' \
-map '[v]' -map '[a1]' -map '[a2]' output.mkv
```

- Concatenate two parts, handling audio and video separately, using the (a)movie sources, and adjusting the resolution:

```
movie=part1.mp4, scale=512:288 [v1] ; amovie=part1.mp4 [a1] ;
movie=part2.mp4, scale=512:288 [v2] ; amovie=part2.mp4 [a2] ;
[v1] [v2] concat [outv] ; [a1] [a2] concat=v=0:a=1 [outa]
```

Note that a desync will happen at the stitch if the audio and video streams do not have exactly the same duration in the first file.

## 40.4 ebur128# TOC

EBU R128 scanner filter. This filter takes an audio stream as input and outputs it unchanged. By default, it logs a message at a frequency of 10Hz with the Momentary loudness (identified by M), Short-term loudness (S), Integrated loudness (I) and Loudness Range (LRA).

The filter also has a video output (see the *video* option) with a real time graph to observe the loudness evolution. The graphic contains the logged message mentioned above, so it is not printed anymore when this option is set, unless the verbose logging is set. The main graphing area contains the short-term loudness (3 seconds of analysis), and the gauge on the right is for the momentary loudness (400 milliseconds).

More information about the Loudness Recommendation EBU R128 on <http://tech.ebu.ch/loudness>.

The filter accepts the following options:

**video**

Activate the video output. The audio stream is passed unchanged whether this option is set or no. The video stream will be the first output stream if activated. Default is 0.

**size**

Set the video size. This option is for video only. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default and minimum resolution is 640x480.

**meter**

Set the EBU scale meter. Default is 9. Common values are 9 and 18, respectively for EBU scale meter +9 and EBU scale meter +18. Any other integer value between this range is allowed.

## metadata

Set metadata injection. If set to 1, the audio input will be segmented into 100ms output frames, each of them containing various loudness information in metadata. All the metadata keys are prefixed with `lavfi.r128..`

Default is 0.

## framelog

Force the frame logging level.

Available values are:

‘info’

information logging level

‘verbose’

verbose logging level

By default, the logging level is set to *info*. If the `video` or the `metadata` options are set, it switches to *verbose*.

## peak

Set peak mode(s).

Available modes can be cumulated (the option is a `flag` type). Possible values are:

‘none’

Disable any peak mode (default).

‘sample’

Enable sample-peak mode.

Simple peak mode looking for the higher sample value. It logs a message for sample-peak (identified by SPK).

‘true’

Enable true-peak mode.

If enabled, the peak lookup is done on an over-sampled version of the input stream for better peak accuracy. It logs a message for true-peak. (identified by TPK) and true-peak per frame (identified by FTPK). This mode requires a build with libswresample.

#### 40.4.1 Examples# TOC

- Real-time graph using `ffplay`, with a EBU scale meter +18:

```
ffplay -f lavfi -i "amovie=input.mp3,eburl28=video=1:meter=18 [out0][out1]"
```

- Run an analysis with `ffmpeg`:

```
ffmpeg -nostats -i input.mp3 -filter_complex eburl28 -f null -
```

### 40.5 interleave, ainterleave# TOC

Temporally interleave frames from several inputs.

`interleave` works with video inputs, `ainterleave` with audio.

These filters read frames from several inputs and send the oldest queued frame to the output.

Input streams must have a well defined, monotonically increasing frame timestamp values.

In order to submit one frame to output, these filters need to enqueue at least one frame for each input, so they cannot work in case one input is not yet terminated and will not receive incoming frames.

For example consider the case when one input is a `select` filter which always drop input frames. The `interleave` filter will keep reading from that input, but it will never be able to send new frames to output until the input will send an end-of-stream signal.

Also, depending on inputs synchronization, the filters will drop frames in case one input receives more frames than the other ones, and the queue is already filled.

These filters accept the following options:

`nb_inputs, n`

Set the number of different inputs, it is 2 by default.

#### 40.5.1 Examples# TOC

- Interleave frames belonging to different streams using `ffmpeg`:

```
ffmpeg -i bambi.avi -i pr0n.mkv -filter_complex "[0:v][1:v] interleave" out.avi
```

- Add flickering blur effect:

```
select='if(gt(random(0), 0.2), 1, 2)':n=2 [tmp], boxblur=2:2, [tmp] interleave
```

## 40.6 perms, aperms# TOC

Set read/write permissions for the output frames.

These filters are mainly aimed at developers to test direct path in the following filter in the filtergraph.

The filters accept the following options:

mode

Select the permissions mode.

It accepts the following values:

‘none’

Do nothing. This is the default.

‘ro’

Set all the output frames read-only.

‘rw’

Set all the output frames directly writable.

‘toggle’

Make the frame read-only if writable, and writable if read-only.

‘random’

Set each output frame read-only or writable randomly.

seed

Set the seed for the *random* mode, must be an integer included between 0 and `UINT32_MAX`. If not specified, or if explicitly set to `-1`, the filter will try to use a good random seed on a best effort basis.

Note: in case of auto-inserted filter between the permission filter and the following one, the permission might not be received as expected in that following filter. Inserting a `format` or `aformat` filter before the `perms/aperms` filter can avoid this problem.

## 40.7 select, aselect# TOC

Select frames to pass in output.

This filter accepts the following options:

`expr, e`

Set expression, which is evaluated for each input frame.

If the expression is evaluated to zero, the frame is discarded.

If the evaluation result is negative or NaN, the frame is sent to the first output; otherwise it is sent to the output with index `ceil(val)-1`, assuming that the input index starts from 0.

For example a value of `1.2` corresponds to the output with index `ceil(1.2)-1 = 2-1 = 1`, that is the second output.

`outputs, n`

Set the number of outputs. The output to which to send the selected frame is based on the result of the evaluation. Default value is 1.

The expression can contain the following constants:

`n`

The (sequential) number of the filtered frame, starting from 0.

`selected_n`

The (sequential) number of the selected frame, starting from 0.

`prev_selected_n`

The sequential number of the last selected frame. It's NAN if undefined.

`TB`

The timebase of the input timestamps.

`pts`

The PTS (Presentation TimeStamp) of the filtered video frame, expressed in *TB* units. It's NAN if undefined.

`t`

The PTS of the filtered video frame, expressed in seconds. It's NAN if undefined.

`prev_pts`

The PTS of the previously filtered video frame. It's NAN if undefined.

`prev_selected_pts`

The PTS of the last previously filtered video frame. It's NAN if undefined.

`prev_selected_t`

The PTS of the last previously selected video frame. It's NAN if undefined.

`start_pts`

The PTS of the first video frame in the video. It's NAN if undefined.

`start_t`

The time of the first video frame in the video. It's NAN if undefined.

`pict_type (video only)`

The type of the filtered frame. It can assume one of the following values:

I

P

B

S

SI

SP

BI

`interlace_type (video only)`

The frame interlace type. It can assume one of the following values:

PROGRESSIVE

The frame is progressive (not interlaced).

TOPFIRST

The frame is top-field-first.

BOTTOMFIRST

The frame is bottom-field-first.

`consumed_sample_n (audio only)`

the number of selected samples before the current frame

`samples_n (audio only)`

the number of samples in the current frame

`sample_rate (audio only)`

the input sample rate

`key`

This is 1 if the filtered frame is a key-frame, 0 otherwise.

`pos`

the position in the file of the filtered frame, -1 if the information is not available (e.g. for synthetic video)

`scene (video only)`

value between 0 and 1 to indicate a new scene; a low value reflects a low probability for the current frame to introduce a new scene, while a higher value means the current frame is more likely to be one (see the example below)

The default value of the select expression is "1".

### 40.7.1 Examples# TOC

- Select all frames in input:

```
select
```

The example above is the same as:

```
select=1
```

- Skip all frames:

```
select=0
```

- Select only I-frames:

```
select='eq(pict_type\,I)'
```

- Select one frame every 100:

```
select='not(mod(n\,100))'
```

- Select only frames contained in the 10-20 time interval:

```
select=between(t\,10\,20)
```

- Select only I frames contained in the 10-20 time interval:

```
select=between(t\,10\,20)*eq(pict_type\,I)
```

- Select frames with a minimum distance of 10 seconds:

```
select='isnan(prev_selected_t)+gte(t-prev_selected_t\,10)'
```

- Use aselect to select only audio frames with samples number > 100:

```
aselect='gt(samples_n\,100)'
```

- Create a mosaic of the first scenes:

```
ffmpeg -i video.avi -vf select='gt(scene\,0.4)',scale=160:120,tile -frames:v 1 preview.png
```

Comparing *scene* against a value between 0.3 and 0.5 is generally a sane choice.

- Send even and odd frames to separate outputs, and compose them:

```
select=n=2:e='mod(n, 2)+1' [odd][even]; [odd] pad=h=2*ih [tmp]; [tmp][even] overlay=y=h
```

## 40.8 sendcmd, asendcmd# TOC

Send commands to filters in the filtergraph.

These filters read commands to be sent to other filters in the filtergraph.

sendcmd must be inserted between two video filters, asendcmd must be inserted between two audio filters, but apart from that they act the same way.

The specification of commands can be provided in the filter arguments with the *commands* option, or in a file specified by the *filename* option.

These filters accept the following options:

`commands, c`

Set the commands to be read and sent to the other filters.

`filename, f`

Set the filename of the commands to be read and sent to the other filters.



## 40.8.1 Commands syntax# TOC

A commands description consists of a sequence of interval specifications, comprising a list of commands to be executed when a particular event related to that interval occurs. The occurring event is typically the current frame time entering or leaving a given time interval.

An interval is specified by the following syntax:

```
START [ -END ] COMMANDS ;
```

The time interval is specified by the *START* and *END* times. *END* is optional and defaults to the maximum time.

The current frame time is considered within the specified interval if it is included in the interval [*START*, *END*), that is when the time is greater or equal to *START* and is lesser than *END*.

*COMMANDS* consists of a sequence of one or more command specifications, separated by ",", relating to that interval. The syntax of a command specification is given by:

```
[FLAGS] TARGET COMMAND ARG
```

*FLAGS* is optional and specifies the type of events relating to the time interval which enable sending the specified command, and must be a non-null sequence of identifier flags separated by "+" or "|" and enclosed between "[" and "]".

The following flags are recognized:

`enter`

The command is sent when the current frame timestamp enters the specified interval. In other words, the command is sent when the previous frame timestamp was not in the given interval, and the current is.

`leave`

The command is sent when the current frame timestamp leaves the specified interval. In other words, the command is sent when the previous frame timestamp was in the given interval, and the current is not.

If *FLAGS* is not specified, a default value of [`enter`] is assumed.

*TARGET* specifies the target of the command, usually the name of the filter class or a specific filter instance name.

*COMMAND* specifies the name of the command for the target filter.

*ARG* is optional and specifies the optional list of argument for the given *COMMAND*.

Between one interval specification and another, whitespaces, or sequences of characters starting with # until the end of line, are ignored and can be used to annotate comments.

A simplified BNF description of the commands specification syntax follows:

```
COMMAND_FLAG ::= "enter" | "leave"
COMMAND_FLAGS ::= COMMAND_FLAG [(+|"")COMMAND_FLAG]
COMMAND ::= [" " COMMAND_FLAGS "]" TARGET COMMAND [ARG]
COMMANDS ::= COMMAND [,COMMANDS]
INTERVAL ::= START[-END] COMMANDS
INTERVALS ::= INTERVAL[;INTERVALS]
```

## 40.8.2 Examples# TOC

- Specify audio tempo change at second 4:

```
asendcmd=c='4.0 atempo tempo 1.5',atempo
```

- Specify a list of drawtext and hue commands in a file.

```
# show text in the interval 5-10
5.0-10.0 [enter] drawtext reinit 'fontfile=FreeSerif.ttf:text=hello world',
        [leave] drawtext reinit 'fontfile=FreeSerif.ttf:text=';

# desaturate the image in the interval 15-20
15.0-20.0 [enter] hue s 0,
        [enter] drawtext reinit 'fontfile=FreeSerif.ttf:text=nocolor',
        [leave] hue s 1,
        [leave] drawtext reinit 'fontfile=FreeSerif.ttf:text=color';

# apply an exponential saturation fade-out effect, starting from time 25
25 [enter] hue s exp(25-t)
```

A filtergraph allowing to read and process the above command list stored in a file `test.cmd`, can be specified with:

```
sendcmd=f=test.cmd,drawtext=fontfile=FreeSerif.ttf:text='',hue
```

## 40.9 setpts, asetpts# TOC

Change the PTS (presentation timestamp) of the input frames.

`setpts` works on video frames, `asetpts` on audio frames.

This filter accepts the following options:

`expr`

The expression which is evaluated for each frame to construct its timestamp.

The expression is evaluated through the eval API and can contain the following constants:

FRAME\_RATE

frame rate, only defined for constant frame-rate video

PTS

The presentation timestamp in input

N

The count of the input frame for video or the number of consumed samples, not including the current frame for audio, starting from 0.

NB\_CONSUMED\_SAMPLES

The number of consumed samples, not including the current frame (only audio)

NB\_SAMPLES, S

The number of samples in the current frame (only audio)

SAMPLE\_RATE, SR

The audio sample rate.

STARTPTS

The PTS of the first frame.

STARTT

the time in seconds of the first frame

INTERLACED

State whether the current frame is interlaced.

T

the time in seconds of the current frame

POS

original position in the file of the frame, or undefined if undefined for the current frame

PREV\_INPTS

The previous input PTS.

PREV\_INT

previous input time in seconds

PREV\_OUTPTS

The previous output PTS.

PREV\_OUTT

previous output time in seconds

RTCTIME

The wallclock (RTC) time in microseconds. This is deprecated, use time(0) instead.

RTCSTART

The wallclock (RTC) time at the start of the movie in microseconds.

TB

The timebase of the input timestamps.

#### 40.9.1 Examples# TOC

- Start counting PTS from zero

```
setpts=PTS-STARTPTS
```

- Apply fast motion effect:

```
setpts=0.5*PTS
```

- Apply slow motion effect:

```
setpts=2.0*PTS
```

- Set fixed rate of 25 frames per second:

```
setpts=N/(25*Tb)
```

- Set fixed rate 25 fps with some jitter:

```
setpts='1/(25*Tb) * (N + 0.05 * sin(N*2*PI/25))'
```

- Apply an offset of 10 seconds to the input PTS:

```
setpts=PTS+10/TB
```

- Generate timestamps from a "live source" and rebase onto the current timebase:

```
setpts='(RTCTIME - RTCSTART) / (TB * 1000000)'
```

- Generate timestamps by counting samples:

```
asetpts=N/SR/TB
```

## 40.10 settb, asetb# TOC

Set the timebase to use for the output frames timestamps. It is mainly useful for testing timebase configuration.

It accepts the following parameters:

`expr`, `tb`

The expression which is evaluated into the output timebase.

The value for `tb` is an arithmetic expression representing a rational. The expression can contain the constants "AVTB" (the default timebase), "intb" (the input timebase) and "sr" (the sample rate, audio only). Default value is "intb".

### 40.10.1 Examples# TOC

- Set the timebase to 1/25:

```
settb=expr=1/25
```

- Set the timebase to 1/10:

```
settb=expr=0.1
```

- Set the timebase to 1001/1000:

```
settb=1+0.001
```

- Set the timebase to 2\*intb:

```
settb=2*intb
```

- Set the default timebase value:

```
settb=AVTB
```

## 40.11 showcqt# TOC

Convert input audio to a video output representing frequency spectrum logarithmically (using constant Q transform with Brown-Puckette algorithm), with musical tone scale, from E0 to D#10 (10 octaves).

The filter accepts the following options:

volume

Specify transform volume (multiplier) expression. The expression can contain variables:

frequency, freq, f

the frequency where transform is evaluated

timeclamp, tc

value of timeclamp option

and functions:

a\_weighting(f)

A-weighting of equal loudness

b\_weighting(f)

B-weighting of equal loudness

c\_weighting(f)

C-weighting of equal loudness

Default value is 16.

tlength

Specify transform length expression. The expression can contain variables:

frequency, freq, f

the frequency where transform is evaluated

timeclamp, tc

value of timeclamp option

Default value is  $384/f*tc/(384/f+tc)$ .

timeclamp

Specify the transform timeclamp. At low frequency, there is trade-off between accuracy in time domain and frequency domain. If timeclamp is lower, event in time domain is represented more accurately (such as fast bass drum), otherwise event in frequency domain is represented more accurately (such as bass guitar). Acceptable value is [0.1, 1.0]. Default value is 0.17.

coeffclamp

Specify the transform coeffclamp. If coeffclamp is lower, transform is more accurate, otherwise transform is faster. Acceptable value is [0.1, 10.0]. Default value is 1.0.

gamma

Specify gamma. Lower gamma makes the spectrum more contrast, higher gamma makes the spectrum having more range. Acceptable value is [1.0, 7.0]. Default value is 3.0.

gamma2

Specify gamma of bargraph. Acceptable value is [1.0, 7.0]. Default value is 1.0.

fontfile

Specify font file for use with freetype. If not specified, use embedded font.

fontcolor

Specify font color expression. This is arithmetic expression that should return integer value 0xRRGGBB. The expression can contain variables:

frequency, freq, f

the frequency where transform is evaluated

timeclamp, tc

value of timeclamp option

and functions:

midi(f)

midi number of frequency f, some midi numbers: E0(16), C1(24), C2(36), A4(69)

r(x), g(x), b(x)

red, green, and blue value of intensity x

Default value is `st(0, (midi(f)-59.5)/12); st(1, if(between(ld(0),0,1), 0.5-0.5*cos(2*PI*ld(0)), 0)); r(1-lld(1)) + b(lld(1))`

`fullhd`

If set to 1 (the default), the video size is 1920x1080 (full HD), if set to 0, the video size is 960x540. Use this option to make CPU usage lower.

`fps`

Specify video fps. Default value is 25.

`count`

Specify number of transform per frame, so there are `fps*count` transforms per second. Note that audio data rate must be divisible by `fps*count`. Default value is 6.

### 40.11.1 Examples# TOC

- Playing audio while showing the spectrum:

```
ffplay -f lavfi 'amovie=a.mp3, asplit [a][out1]; [a] showcqt [out0]'
```

- Same as above, but with frame rate 30 fps:

```
ffplay -f lavfi 'amovie=a.mp3, asplit [a][out1]; [a] showcqt=fps=30:count=5 [out0]'
```

- Playing at 960x540 and lower CPU usage:

```
ffplay -f lavfi 'amovie=a.mp3, asplit [a][out1]; [a] showcqt=fullhd=0:count=3 [out0]'
```

- A1 and its harmonics: A1, A2, (near)E3, A3:

```
ffplay -f lavfi 'aevalsrc=0.1*sin(2*PI*55*t)+0.1*sin(4*PI*55*t)+0.1*sin(6*PI*55*t)+0.1*sin(8*PI*55*t), asplit[a][out1]; [a] showcqt [out0]'
```

- Same as above, but with more accuracy in frequency domain (and slower):

```
ffplay -f lavfi 'aevalsrc=0.1*sin(2*PI*55*t)+0.1*sin(4*PI*55*t)+0.1*sin(6*PI*55*t)+0.1*sin(8*PI*55*t), asplit[a][out1]; [a] showcqt=timeclamp=0.5 [out0]'
```

- B-weighting of equal loudness

```
volume=16*b_weighting(f)
```

- Lower Q factor

```
tlength=100/f*tc/(100/f+tc)
```

- Custom fontcolor, C-note is colored green, others are colored blue



```
fontcolor='if(mod(floor(midi(f)+0.5),12), 0x0000FF, g(1))'
```

- Custom gamma, now spectrum is linear to the amplitude.

```
gamma=2:gamma2=2
```

## 40.12 showfreqs# TOC

Convert input audio to video output representing the audio power spectrum. Audio amplitude is on Y-axis while frequency is on X-axis.

The filter accepts the following options:

`size, s`

Specify size of video. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default is 1024x512.

`mode`

Set display mode. This set how each frequency bin will be represented.

It accepts the following values:

```
'line'  
'bar'  
'dot'
```

Default is bar.

`ascale`

Set amplitude scale.

It accepts the following values:

```
'lin'
```

Linear scale.

```
'sqrt'
```

Square root scale.

```
'cbrt'
```

Cubic root scale.

`'log'`

Logarithmic scale.

Default is `log`.

`fscale`

Set frequency scale.

It accepts the following values:

`'lin'`

Linear scale.

`'log'`

Logarithmic scale.

`'rlog'`

Reverse logarithmic scale.

Default is `lin`.

`win_size`

Set window size.

It accepts the following values:

`'w16'`

`'w32'`

`'w64'`

`'w128'`

`'w256'`

`'w512'`

`'w1024'`

`'w2048'`

`'w4096'`

`'w8192'`

`'w16384'`

`'w32768'`

`'w65536'`

Default is `w2048`

`win_func`

Set windowing function.

It accepts the following values:

`'rect'`  
`'bartlett'`  
`'hanning'`  
`'hamming'`  
`'blackman'`  
`'welch'`  
`'flattop'`  
`'bharris'`  
`'bnuttall'`  
`'bhann'`  
`'sine'`  
`'nutall'`

Default is `hanning`.

`overlap`

Set window overlap. In range `[ 0 , 1 ]`. Default is 1, which means optimal overlap for selected window function will be picked.

`averaging`

Set time averaging. Setting this to 0 will display current maximal peaks. Default is 1, which means time averaging is disabled.

`color`

Specify list of colors separated by space or by `'|'` which will be used to draw channel frequencies. Unrecognized or missing colors will be replaced by white color.

## **40.13 showspectrum# TOC**

Convert input audio to a video output, representing the audio frequency spectrum.

The filter accepts the following options:

`size, s`

Specify the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 640x512.

slide

Specify how the spectrum should slide along the window.

It accepts the following values:

‘replace’

the samples start again on the left when they reach the right

‘scroll’

the samples scroll from right to left

‘fullframe’

frames are only produced when the samples reach the right

Default value is `replace`.

mode

Specify display mode.

It accepts the following values:

‘combined’

all channels are displayed in the same row

‘separate’

all channels are displayed in separate rows

Default value is ‘combined’.

color

Specify display color mode.

It accepts the following values:

‘channel’

each channel is displayed in a separate color

‘intensity’

each channel is displayed using the same color scheme

Default value is 'channel'.

scale

Specify scale used for calculating intensity color values.

It accepts the following values:

'lin'

linear

'sqrt'

square root, default

'cbrt'

cubic root

'log'

logarithmic

Default value is 'sqrt'.

saturation

Set saturation modifier for displayed colors. Negative values provide alternative color scheme. 0 is no saturation at all. Saturation must be in [-10.0, 10.0] range. Default value is 1.

win\_func

Set window function.

It accepts the following values:

'none'

No samples pre-processing (do not expect this to be faster)

'hann'

Hann window

'hamming'

Hamming window

`'blackman'`

Blackman window

Default value is hann.

The usage is very similar to the showwaves filter; see the examples in that section.

### 40.13.1 Examples# TOC

- Large window with logarithmic color scaling:

```
showspectrum=s=1280x480:scale=log
```

- Complete example for a colored and sliding spectrum per channel using ffmpeg:

```
ffmpeg -f lavfi 'amovie=input.mp3, asplit [a][out1];  
[a] showspectrum=mode=separate:color=intensity:slide=1:scale=cbrt [out0]'
```

### 40.14 showvolume# TOC

Convert input audio volume to a video output.

The filter accepts the following options:

rate, r

Set video rate.

b

Set border width, allowed range is [0, 5]. Default is 1.

w

Set channel width, allowed range is [40, 1080]. Default is 400.

h

Set channel height, allowed range is [1, 100]. Default is 20.

f

Set fade, allowed range is [1, 255]. Default is 20.

c

Set volume color expression.

The expression can use the following variables:

VOLUME

Current max volume of channel in dB.

CHANNEL

Current channel number, starting from 0.

t

If set, displays channel names. Default is enabled.

## 40.15 showwaves# TOC

Convert input audio to a video output, representing the samples waves.

The filter accepts the following options:

size, s

Specify the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 600x240.

mode

Set display mode.

Available values are:

‘point’

Draw a point for each sample.

‘line’

Draw a vertical line for each sample.

‘p2p’

Draw a point for each sample and a line between them.

‘cline’

Draw a centered vertical line for each sample.

Default value is `point`.

`n`

Set the number of samples which are printed on the same column. A larger value will decrease the frame rate. Must be a positive integer. This option can be set only if the value for *rate* is not explicitly specified.

`rate, r`

Set the (approximate) output frame rate. This is done by setting the option *n*. Default value is "25".

`split_channels`

Set if channels should be drawn separately or overlap. Default value is 0.

### 40.15.1 Examples# TOC

- Output the input file audio and the corresponding video representation at the same time:

```
amovie=a.mp3,asplit[out0],showwaves[out1]
```

- Create a synthetic signal and show it with showwaves, forcing a frame rate of 30 frames per second:

```
aevalsrc=sin(1*2*PI*t)*sin(880*2*PI*t):cos(2*PI*200*t),asplit[out0],showwaves=r=30[out1]
```

### 40.16 showwavespic# TOC

Convert input audio to a single video frame, representing the samples waves.

The filter accepts the following options:

`size, s`

Specify the video size for the output. For the syntax of this option, check the (ffmpeg-utils)"Video size" section in the ffmpeg-utils manual. Default value is 600x240.

`split_channels`

Set if channels should be drawn separately or overlap. Default value is 0.

### 40.16.1 Examples# TOC

- Extract a channel split representation of the wave form of a whole audio track in a 1024x800 picture using ffmpeg:

```
ffmpeg -i audio.flac -lavfi showwavespic=split_channels=1:s=1024x800 waveform.png
```



## 40.17 split, asplit# TOC

Split input into several identical outputs.

`asplit` works with audio input, `split` with video.

The filter accepts a single parameter which specifies the number of outputs. If unspecified, it defaults to 2.

### 40.17.1 Examples# TOC

- Create two separate outputs from the same input:

```
[in] split [out0][out1]
```

- To create 3 or more outputs, you need to specify the number of outputs, like in:

```
[in] asplit=3 [out0][out1][out2]
```

- Create two separate outputs from the same input, one cropped and one padded:

```
[in] split [splitout1][splitout2];  
[splitout1] crop=100:100:0:0 [cropout];  
[splitout2] pad=200:200:100:100 [padout];
```

- Create 5 copies of the input audio with `ffmpeg`:

```
ffmpeg -i INPUT -filter_complex asplit=5 OUTPUT
```

## 40.18 zmq, azmq# TOC

Receive commands sent through a libzmq client, and forward them to filters in the filtergraph.

`zmq` and `azmq` work as a pass-through filters. `zmq` must be inserted between two video filters, `azmq` between two audio filters.

To enable these filters you need to install the libzmq library and headers and configure FFmpeg with `--enable-libzmq`.

For more information about libzmq see: <http://www.zeromq.org/>

The `zmq` and `azmq` filters work as a libzmq server, which receives messages sent through a network interface defined by the `bind_address` option.

The received message must be in the form:

```
TARGET COMMAND [ARG]
```

*TARGET* specifies the target of the command, usually the name of the filter class or a specific filter instance name.

*COMMAND* specifies the name of the command for the target filter.

*ARG* is optional and specifies the optional argument list for the given *COMMAND*.

Upon reception, the message is processed and the corresponding command is injected into the filtergraph. Depending on the result, the filter will send a reply to the client, adopting the format:

```
ERROR_CODE ERROR_REASON  
MESSAGE
```

*MESSAGE* is optional.

## 40.18.1 Examples# TOC

Look at `tools/zmqsend` for an example of a zmq client which can be used to send commands processed by these filters.

Consider the following filtergraph generated by `ffplay`

```
ffplay -dumpgraph 1 -f lavfi "  
color=s=100x100:c=red [l];  
color=s=100x100:c=blue [r];  
nullsrc=s=200x100, zmq [bg];  
[bg][l] overlay [bg+l];  
[bg+l][r] overlay=x=100 "
```

To change the color of the left side of the video, the following command can be used:

```
echo Parsed_color_0 c yellow | tools/zmqsend
```

To change the right side:

```
echo Parsed_color_1 c pink | tools/zmqsend
```

## 41 Multimedia Sources# TOC

Below is a description of the currently available multimedia sources.

### 41.1 amovie# TOC

This is the same as movie source, except it selects an audio stream by default.

### 41.2 movie# TOC

Read audio and/or video stream(s) from a movie container.

It accepts the following parameters:

filename

The name of the resource to read (not necessarily a file; it can also be a device or a stream accessed through some protocol).

format\_name, f

Specifies the format assumed for the movie to read, and can be either the name of a container or an input device. If not specified, the format is guessed from *movie\_name* or by probing.

seek\_point, sp

Specifies the seek point in seconds. The frames will be output starting from this seek point. The parameter is evaluated with `av_strtod`, so the numerical value may be suffixed by an IS postfix. The default value is "0".

streams, s

Specifies the streams to read. Several streams can be specified, separated by "+". The source will then have as many outputs, in the same order. The syntax is explained in the "Stream specifiers" section in the ffmpeg manual. Two special names, "dv" and "da" specify respectively the default (best suited) video and audio stream. Default is "dv", or "da" if the filter is called as "amovie".

stream\_index, si

Specifies the index of the video stream to read. If the value is -1, the most suitable video stream will be automatically selected. The default value is "-1". Deprecated. If the filter is called "amovie", it will select audio instead of video.

loop

Specifies how many times to read the stream in sequence. If the value is less than 1, the stream will be read again and again. Default value is "1".

Note that when the movie is looped the source timestamps are not changed, so it will generate non monotonically increasing timestamps.

It allows overlaying a second video on top of the main input of a filtergraph, as shown in this graph:

```
input -----> deltapts0 --> overlay --> output
                        ^
                        |
movie --> scale--> deltapts1 -----+
```

### 41.2.1 Examples# TOC

- Skip 3.2 seconds from the start of the AVI file in.avi, and overlay it on top of the input labelled "in":

```
movie=in.avi:seek_point=3.2, scale=180:-1, setpts=PTS-STARTPTS [over];  
[in] setpts=PTS-STARTPTS [main];  
[main][over] overlay=16:16 [out]
```

- Read from a video4linux2 device, and overlay it on top of the input labelled "in":

```
movie=/dev/video0:f=video4linux2, scale=180:-1, setpts=PTS-STARTPTS [over];  
[in] setpts=PTS-STARTPTS [main];  
[main][over] overlay=16:16 [out]
```

- Read the first video stream and the audio stream with id 0x81 from dvd.vob; the video is connected to the pad named "video" and the audio is connected to the pad named "audio":

```
movie=dvd.vob:s=v:0+#0x81 [video] [audio]
```

## 42 See Also# TOC

ffmpeg, ffplay, ffprobe, ffserver, ffmpeg-utils, ffmpeg-scaler, ffmpeg-resampler, ffmpeg-codecs, ffmpeg-bitstream-filters, ffmpeg-formats, ffmpeg-devices, ffmpeg-protocols, ffmpeg-filters

## 43 Authors# TOC

The FFmpeg developers.

For details about the authorship, see the Git history of the project ([git://source.ffmpeg.org/ffmpeg](http://source.ffmpeg.org/ffmpeg)), e.g. by typing the command `git log` in the FFmpeg source directory, or browsing the online repository at <http://source.ffmpeg.org>.

Maintainers for the specific components are listed in the file MAINTAINERS in the source code tree.

This document was generated using *makeinfo*.