FFmpeg Bitstream Filters Documentation

Table of Contents

1 Description# TOC

This document describes the bitstream filters provided by the libavcodec library.

A bitstream filter operates on the encoded stream data, and performs bitstream level modifications without performing decoding.

2 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.

2.1 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.

2.2 chomp# TOC

Remove zero padding at the end of a packet.

2.3 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
```

2.4 dca_core# TOC

Extract DCA core from DTS-HD streams.

2.5 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
```

2.6 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
```

2.7 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
```

2.8 mjpega_dump_header# TOC

2.9 movsub# TOC

2.10 mp3_header_decompress# TOC

2.11 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
```

2.12 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.

2.13 remove_extra# TOC

3 See Also# TOC

ffmpeg, ffplay, ffprobe, ffserver, libavcodec

4 Authors# TOC

The FFmpeg developers.

For details about the authorship, see the Git history of the project (git://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*.