Decisions

Contents

- 2026-09-28: What The 2.6.0 Audit Found In Trimming
- 2026-09-28: Rows Are Variants, Not One Timeline
- 2026-09-14: A Split Decides Each Joint Once
- 2026-09-14: Sound Files Are Cut On The Sample, And The Purist May Copy Packets
- 2026-08-26: A Real Camera File Found What Synthetic Ones Could Not
- 2026-08-26: What The Camera Matrix Still Does Not Cover
- 2026-08-26: MXF Rewrites PCM Instead Of Copying It
- 2026-08-26: A Channel The File Does Not Have Is Refused
- 2026-08-25: A/V Sync Is Measured, Not Assumed
- 2026-08-25: Trimming Is Per File
- 2026-08-25: The Trim Backend Answers To A Source Matrix, Not To One Clip
- 2026-08-25: A Container That Cannot Hold The Stream Is Refused Before The Run
- 2026-08-25: Cut Points Are Wall-Clock Time, Timecode Is Metadata
- 2026-08-26: A Stopped Run Leaves Nothing That Cannot Be Opened
- 2026-08-26: FFmpeg 9.0.1 Was Verified, Not Adopted Blindly
- 2026-08-26: The Rate Comes From The File, Unaltered
- 2026-08-25: The Head Snaps To A Keyframe, The Tail Is Counted In Frames
- 2026-08-25: A Refused Overwrite Is A Skip, Not A Success
- 2026-08-24: Trimming Targets Camera Footage, Not Film Muxing
- 2026-08-24: Trimming Cuts On Keyframes Only
- The Window

2026-09-28: What The 2.6.0 Audit Found In Trimming

Five defects over an audit and a re-audit, each reproduced on real FFmpeg before it was fixed, each now held by a test that fails on the old code:

- **Subtitles are rebuilt for a cut, not copied through it.** First an MP4 take with subtitles did not trim at all ("Error selecting an encoder": the stream was mapped without a codec). Naming the codec and copying fixed that, and the re-audit then measured the copy wrong at every boundary (TRIM-5): a frame count dropped the line across OUT, a seek kept a line from before IN at full length, and the `Cut it out` join made a 0.5-1.5 s line a microsecond long and moved the next a second early - while a test that looked only for a subtitle track passed. Now each line is read out as SRT on the file's own clock (`-copyts`, minus the picture's start), cut to the spans kept, moved onto the result's timeline, and muxed as a second input of the call that copies the picture. MP4 and MOV hold mov_text, Matroska SRT (measured: neither takes the other's form); picture subtitles cannot be cut without redrawing and stay behind, named. The tests now read the lines back - text, start and end - and fail on the copying version.
- **The run says what happened.** The service returned counts and no `ok`, and the executor reads only `ok` - so a failed write, refused points and a cancel all showed a green status and exit code 0. A result kept because Overwrite is off is a skip the operator asked for, not an error.
- **A cancel cleans only what it stopped.** It used to inspect the target of the next, unstarted piece and remove it when it would not open - an older file, with Overwrite off.
- **The player hands over only its own file's points.** The arrows change the file in the panel, not in mpv, and `Take A and B` wrote the old take's loop into the new one. The file is asked of mpv itself. A and B are cleared when another file is opened: they are player options, and measured, a loop of 1-3 s survived a plain `loadfile`.

2026-09-28: Rows Are Variants, Not One Timeline

The trim section grew a `+` that adds another row of IN and OUT. The rows are deliberately not pieces of one timeline, the way an editor's segment list is: each row of `Keep it` or `Cut it out` is a separate run on the same take and a file of its own, `_keep01` or `_cut01`, numbered as the row on screen. Nothing between rows has to be reconciled, so rows may overlap freely - one moment can go into two clips, and each file of a cut has its own piece taken out of the whole take. One filled row is exactly the old run, with no suffix, which is what keeps the wrappers and the old behaviour intact.

`Split` stays one timeline: N points give N + 1 parts, up to 30 points. Points from the player are taken only walking forward - a position at or behind an earlier point writes nothing - because a slider

dragged back by accident would otherwise scramble the parts. The `+` clears A and B in the player, not the take of the points: while a row is open its loop keeps playing, and a wrong point is heard, which is the reason the loop exists. Two points on one keyframe stop the run and are named; merging them silently was considered and not done, since pieces under a second are not what a split is used for.

The `▲▼` spinners do not break a timecode into one field per section: it could no longer be copied and pasted whole. The field stays one line, and an arrow steps the section the caret stands in. The smallest section steps by a frame of the file, not a millisecond, which changes nothing in a cut on 25p.

2026-09-14: A Split Decides Each Joint Once

A split used to be two independent cuts - drop the tail up to the point, drop the head from it - and only the head snapped to a keyframe, so the two parts met only when the point already was one. Measured on 750 frames with a keyframe every second: a point at 10.2 s put 5 frames into both parts, a point at 10.7 s left 7 frames in neither.

Now `plan_split` snaps each point to its nearest keyframe once, and that keyframe is the shared joint: one part ends just before it, the next starts on it. IN alone gives two parts, IN and OUT give three. Part lengths are counted with `round`, not `ceil`: a keyframe timestamp already sits on a frame boundary, and at 23.976 frame 240 reads 10.010010 s - 240.00024 frames, which `ceil` would put into both parts. Five cases were checked by hashing every decoded frame and comparing raw samples: the parts laid end to end are the source.

2026-09-14: Sound Files Are Cut On The Sample, And The Purist May Copy Packets

A file of sound alone has no keyframes, so nothing snaps: each point is the nearest sample, every piece is decoded and cut with `atrim` by sample number, and the result is written as it was or in a format from the Audio set. Decoding costs nothing here: the piece is bit for bit the same slice of the fully decoded source, measured on WAV, FLAC, MP3, AAC and Vorbis. For a compressed source the lossless choice is WAV or FLAC; "As is" encodes it once more and says so.

`Exact cut` can be turned off. Then compressed packets are copied untouched, each point moves to a packet boundary, and a joint costs what was measured for that codec - 34 ms on MP3, 21 ms on AAC and Vorbis, about half a second on Opus - which the log states. Two container defects were found and removed rather than written up: MP3 pieces carried the source's encoder-delay header and shifted the sound after every joint, and an AAC join replayed the priming the edit list had hidden. Codecs and containers that were not measured are not offered in this mode.

A Broadcast WAV's time reference shifts with the cut, exactly as a video timecode does. FFmpeg drops the `bext` chunk unless asked and hands four of its fields back under names its own writer does not read; they are passed back under the right ones. iXML cannot be written by FFmpeg at all, and the log says it stays behind.

2026-08-26: A Real Camera File Found What Synthetic Ones Could Not

A Canon `MVI_*.MP4` - HEVC Rext 4:2:2 10-bit UHD at 23.976, LPCM `pcm_s16be` (`twos`), timecode 15:58:57:17 - went through every mode. Frames were exact everywhere (242 / 240 / 166, and a split of 144 + 170 that accounts for all 314 frames of the source), the timecode shifted correctly, and the audio was found verbatim inside the source: the piece begins 48 samples - one millisecond, a fortieth of a frame - after the cut point, copied rather than re-encoded.

Two defects only a real file could show:

Camera metadata was being lost. The source carries `make`, `model` and the brand `mp42hvc1CAEP` in an mdta box; `-map_metadata 0` does not move those, so the muxer wrote generic `isomiso2mp41` and dropped the rest. The plan for this section says camera metadata is to be preserved, so `-movflags use_metadata_tags` was added: verified tag by tag against the source afterwards.

Picking a channel changed the sample format. `pcm_s16be` came out as `pcm_s16le` - the same samples byte-swapped, and a different tag in the file. Both containers accept big-endian PCM, so the source's endianness is kept.

Neither would have surfaced on generated clips: `lavfi` writes little-endian PCM and no camera tags at all.

2026-08-26: What The Camera Matrix Still Does Not Cover

Verified since: two audio tracks (camera plus recorder) both survive with the timecode shifted; HDR/HLG colour tags come through unchanged; an open-GOP source cuts to the exact frame count; 59.94 drop-frame timecode shifts correctly (`01:00:00;00` to `01:00:03;00`); and HEVC with LPCM in MP4 — what Sony and Panasonic actually write — keeps `hvc1`, keeps `ipcm`, and lands on the frame.

The PCM-in-MP4 note now fires only when the container is being changed. A camera that shoots HEVC with LPCM already writes exactly that MP4, and warning about it on every such file is noise

rather than information.

Still untested, and worth saying plainly rather than implying the matrix is complete: real camera files (this is all synthetic), files over an hour, a full disk, a source on a network path or held open by another program, cancelling a run mid-write, several operations at once, 96 kHz or 32-bit float audio, and whether an open-GOP cut shows artefacts in the first frames rather than merely counting them correctly.

2026-08-26: MXF Rewrites PCM Instead Of Copying It

MXF is Premiere's working format, so the 9-18 ms it was shifting was not an acceptable rounding error. Measured sample by sample, the cause was not the muxer's edit units at all: copying PCM into MXF carries the 384 samples between the chunk boundary and the cut - 8 ms of sound that belongs before the cut.

Rewriting the same PCM at the same depth removes it. The encoder trims on the sample it was given, and the result matches a reference cut byte for byte (`exact_match=True`), so nothing is lost and nothing shifts: MXF now measures the same as MOV. PCM to PCM is not a generation, and the log says the rewrite happened rather than leaving it to be discovered.

Compressed audio never reaches that path: the MXF muxer refuses aac, ac3, mp3, alac and flac outright ("not supported by the bitstream filter"). Such a file is skipped before the run with the reason and the alternative, instead of failing with exit code 4294967274.

2026-08-26: A Channel The File Does Not Have Is Refused

`pan=mono|c0=c1` on a mono source is not an error to FFmpeg: it maps a channel that is not there and writes silence, with exit code 0. So the channel count is checked against the request before the run, in the contract, and both front-ends skip the file with the reason rather than producing a silent track.

Found the same way: the CLI trim path asked `audio_stream_info` for `codec_name` while that probe returns `codec`. It therefore concluded that every file had no audio and wrote `-an` - silent output on every command-line trim, reported as success. The probe now returns both spellings and the sample format it was already being asked for.

2026-08-25: A/V Sync Is Measured, Not Assumed

A trim that moves one stream and not the other is the failure that shows up last - in the edit, not in the log - so it is verified with a marker that exists in both streams at the same instant: a white frame at the top of every second and a 20 ms click at the same moment.

Reading those markers correctly took two attempts. `blackdetect` and `silencedetect` work in whole frames and in windows, which puts a 40 ms floor under the measurement, and comparing "first detected flash" with "first detected click" compares different events when one stream opens on a marker and the other does not. The honest method reads the video through `signalstats` with `fps_mode passthrough` - so no frame is duplicated to fit a filter's clock - and pairs each flash with its own click on the file's real timestamps.

Measured that way, against the source's own offset:

Container	Change vs source
MP4	0 … 2 ms
MOV	0 … 5 ms
MKV	1 … 3 ms
MXF	9 … 18 ms

Every mode is covered - drop the head, drop the tail, keep the middle, split in two - on AAC, on 24-bit PCM and on ProRes, including the channel-picking path that rebuilds the audio. Only MXF shifts anything: its muxer aligns audio to its own edit units, which is under half a frame and is now stated as a warning.

An earlier warning claimed MKV ran the audio ahead by the length of an audio packet. The precise measurement disproved it - MKV shifts the start of both streams together, which is not a sync error - and the warning was removed rather than left in as a plausible-sounding caution.

2026-08-25: Trimming Is Per File

A folder of takes does not share cut points: the keyframes differ, the slate differs, the walk-away differs. So the window trims one file at a time, and the file is chosen by stepping — a badge with the current name and an arrow on each side — rather than from a dropdown.

The CLI wrappers trim one file too: the one `AUDION_TRIM_FILE` points at, or the first in `input`, and the wrapper says which. Cutting a folder to one pair of numbers was never the useful behaviour, only the easy one.

2026-08-25: The Trim Backend Answers To A Source Matrix, Not To One Clip

Trimming was verified against a spread of camera-shaped sources rather than the one file it was written on: ProRes/PCM in MOV, long-GOP HEVC, all-intra H.264, 29.97 drop-frame, variable frame rate, four audio channels, MPEG-TS, and a name with spaces and Cyrillic. Five defects only that spread could show:

A frame just inside the end point belongs in the piece. The end index rounds up, not to nearest: at 29.97 a thirty-second point is frame 899.1, and frame 899 stands at 29.9966 — inside. Rounding to nearest silently dropped it.

`-frames:v` is only exact when `-ss` lands on a real keyframe. Given any other position, FFmpeg counts frames from the keyframe before it and then discards the ones ahead of the request, so the piece comes out short by exactly that gap. The service always seeks to the keyframe it computed.

HEVC with a B-pyramid writes two to four frames fewer than asked. The shortfall is stable per stream but does not follow `has_b_frames`, so it is measured rather than guessed: the frames are counted, and if any are missing the cut is redone once with the count raised by the shortfall. Asking for `N + d` yields exactly `N`.

Variable frame rate forbids frame arithmetic. `avg_frame_rate` is an average, so a frame count taken from it covers the wrong span — a twelve-second request came out sixteen seconds long. Such a source is detected by the disagreement between `r_frame_rate` and `avg_frame_rate`, cut by time, and labelled as such in the log.

Timestamps do not always start at zero. MPEG-TS commonly starts at 1.44 s. Positions typed by the operator count from the first picture; FFmpeg counts in the stream's own timeline. The origin is subtracted from probed keyframes and added back at seek time, in the service, in the CLI runner and in the preview.

2026-08-25: A Container That Cannot Hold The Stream Is Refused Before The Run

MP4 cannot hold ProRes, and FFmpeg's way of saying so is "Could not find tag for codec" plus a zero-byte file. Known refusals are checked against the source codec first and reported as a skip with the reason and the containers that would work.

MPEG-TS carries no index, so its seek is interpolated from byte positions: exact in the middle of a file, and capable of overshooting past the end near it. That is stated as a warning, and a result far shorter than planned — or one that cannot be read back at all — is counted as a failure rather than reported as a success with an odd duration.

2026-08-25: Cut Points Are Wall-Clock Time, Timecode Is Metadata

A cut point is entered and displayed as real time (`00:01:28,4`), never as a timecode. On NTSC rates the two disagree by design: non-drop `00:04:00:00` at 29.97 is frame 7200, which is 240.24 s of real time. A tool that treats one as the other drifts by `1001/1000` — the Shutter Encoder defect measured on 20.2 and present since 19.1, which started this work (`E:\TOOLS\fps-test`), four seconds an hour.

Timecode still lives in the operation, but only as metadata to shift. Every conversion between seconds and frames goes through the exact `Fraction`, in `system_core/core/trim_contract.py`.

2026-08-26: A Stopped Run Leaves Nothing That Cannot Be Opened

Cancelling a trim half an hour into an hour-long file killed FFmpeg correctly and reported failure correctly - and left 6 GB in the output folder under the name the finished piece would have had. The muxer never wrote the index, so nothing opens it, but a glance at the folder says the cut is there.

The previous rule removed only empty files, on the reasoning that a partial result may still be worth keeping. That holds while the partial file opens. The test is now readability: what opens stays, and the log says how far it got; what does not open is removed, with the reason and the size.

2026-08-26: FFmpeg 9.0.1 Was Verified, Not Adopted Blindly

The bundled build moved to 9.0.1 and every table was re-measured on it: the container/codec matrix matched the 8.0.1 one cell for cell, the fractional-rate cuts landed on the same frames, and all 75 wrappers ran unchanged. Of the eight options that disappeared between the versions, none is used here.

Which build actually ships is still the installer's decision, taken from the NVIDIA driver version - on most machines that is the 8.x branch. Both are therefore kept working, and the docs name the build the numbers came from rather than a shipped version.

2026-08-26: The Rate Comes From The File, Unaltered

`ffprobe` reports the exact rational rate, so nothing is inferred from it. A file shot at 23.976 states `24000/1001`; one shot at a true 24.000 states `24/1`. Both are read as written.

This replaces the decision of 2026-08-25, which read every integer NTSC rate as its `1000/1001` twin on the belief that containers round. They do not. An ARRI Alexa Mini shooting a true 24.000 (260 frames across 10.8333 s, measured) states `24/1`, and the multiplier turned that correct rate into a wrong one - five frames adrift over four minutes, the same `1001/1000` error the option existed to prevent, pointed the other way.

The checkbox is gone from the panel and from both front-ends. A rate that cannot be read at all is refused rather than replaced with a guess.

2026-08-25: The Head Snaps To A Keyframe, The Tail Is Counted In Frames

The start of a copied piece can only be a keyframe, so it snaps and the real point is printed with its distance from the request. The end has no such constraint: it travels as `-frames:v N`, where N comes from the exact rate.

`-t` was measured and rejected: on a stream with B-frames it overshoots by two or three frames. `-avoid_negative_ts make_zero` was also rejected — combined with a seek before `-i` it starts the video 0.1 s after the audio.

2026-08-25: A Refused Overwrite Is A Skip, Not A Success

FFmpeg answers `-n` on an existing target with "already exists. Exiting." and an exit code of **0**. A run that wrote nothing therefore reads as a success, and a verification step then measures the previous file. The trim path checks the target itself and reports `[SKIP]`.

Encoding, sound, repackaging, subtitles, frame rate and colour check their target the same way: a file already in the Target is passed by before FFmpeg is started, with a `[SKIP]` line in the journal, and counted as skipped, not written.

2026-08-24: Trimming Targets Camera Footage, Not Film Muxing

Trimming is built for camera originals: MOV and MP4, H.264/H.265/ProRes, PCM or AAC audio, one or two tracks. Not films with five language tracks, subtitles and AC-3 — that is muxing, and other tools do it well.

Consequence for the parameter set: channels instead of tracks (a camera writes the shotgun and the lav as two mono channels), camera metadata preserved rather than stripped, and no loudness normalisation — it destroys headroom and the relationship between takes. Normalisation belongs to a separate "meeting recording" profile.

And the one thing that must not be lost: **timecode**. Editors sync cameras to each other and to the field recorder by it. A trim has to shift it correctly, and `tmcd` is known to disappear silently when remuxing MOV to MP4 — so if it cannot be carried over, say so before the run, not after.

2026-08-24: Trimming Cuts On Keyframes Only

When trimming is added, video is copied (`-c:v copy`) and the cut lands on the nearest keyframe. Frame-exact trimming — which requires re-encoding the whole file — is deliberately out of scope.

Rationale: the tool's promise is "fast and lossless". A button that sometimes takes two seconds and sometimes twenty minutes, depending on a setting nobody reads, breaks that promise. Frame-exact cutting is served by a dozen other tools; being fast and honest about it is the more useful niche.

Consequence: the interface must show the *actual* cut point, not the requested one — nearest keyframe from `ffprobe`, with the offset spelled out ("00:01:28,4, −1,6 s"). A silent correction is worse than a slow cut.

The trim in detail: `Docs/tools/TRIM_RU.md`.

The Window

The window is the face of the console command. A button of the window and `media_cli.cmd run` are one and the same: the window starts an operation through the command, and the first line of a run in the journal is a ready command that can be repeated. The form of every operation is drawn from its description in the manifest, so the window, the console and the documents say one thing.

The terminal plays the main part. FFmpeg's output is part of the work: the lines of an operation come in the colours of their services, FFmpeg's own line stands under the progress bar. The outcome of a run stays on screen, not in a notification that is easy to miss.

The forms are separate. Every section and mode has a form of its own, remembered between launches; only the encoder and the decoder are shared by the whole window: they are a choice of the machine, not of the task.

An action names its object. A button stands at its field, and its tooltip names it: "Add LUT files", "Open the LUTs folder" - not "Add" and "Open": one screen may hold several fields with files.

External input is staged locally. The add buttons copy what was chosen into the input folder: safer than long operations on network paths and removable drives.

CMD encoding is a build gate. Every `.cmd` of the project is UTF-8 without a byte order mark and with strict CRLF; the check is built into the build of the environment and stands on its own as `install\Check-CmdEncoding.cmd`.