

Decisions

Contents

- [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: The Form Shows What Survives, Not What Is Removed](#)
- [2026-08-25: Trimming Is Per File, And The Marks Are Remembered](#)
- [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 Scrub Plays One Frame, Not One Second](#)
- [2026-08-26: Sound Answers Before Picture](#)
- [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](#)
- [2026-05-04: GUI Shell Over CLI](#)
- [2026-05-04: Separate Server And Window](#)
- [2026-05-04: Dark Terminal-First Layout](#)
- [2026-05-04: Compact Ghost Buttons](#)
- [2026-05-04: Operation Buttons Left-Align Their Labels](#)
- [2026-05-04: Laptop Compression Is A Baseline Requirement](#)

- [2026-05-07: 1600x900 Is The Roomy Default Window](#)
- [2026-05-07: Form Order Follows User Decisions](#)
- [2026-05-07: One Visible Action Per User Outcome](#)
- [2026-05-07: Actions Must Name Their Object](#)
- [2026-05-07: Advanced Fields Collapse By Default](#)
- [2026-05-07: Model List Is Not Model Access](#)
- [2026-05-07: Visual Smoke Screenshots Are Part Of Porting](#)
- [2026-05-04: Stage External Input Locally](#)
- [2026-05-04: Language And Theme Are Conservative](#)
- [2026-05-06: Completion State Is Persistent](#)
- [2026-05-06: Hide Windows CLI Helpers At Process Creation](#)
- [2026-05-06: CMD Encoding Is A Build Gate](#)

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: The Form Shows What Survives, Not What Is Removed

The trim form answers one question — what will be left — and it answers it visually. The kept span is lit green on both waveform strips and outlined; the discarded parts are covered by a dark veil. Modes are named by their outcome ("Drop the head", "Keep the middle") rather than by the act of cutting.

Points are IN and OUT, the words an editor already uses. The earlier "head" and "tail" collided with the words in the mode names, so nothing on screen told the two apart.

Controls under the faders are icons with tooltips, grouped and centred the way a viewer's transport bar is, with a frame-of-total and time readout on the right. The only text in the row is the time field itself, which states its format — `hh:mm:ss,ms`, neither timecode nor samples.

Two operations joined the section because they answer questions asked while the cut point is being chosen: a still of the current frame (PNG or JPG, full size, into `Transcoded/Stills`) and a split, which cuts the file in two at CUT and throws nothing away — `_part1` and `_part2`, with the second part's timecode shifted.

2026-08-25: Trimming Is Per File, And The Marks Are Remembered

A folder of takes does not share cut points: the keyframes differ, the slate differs, the walk-away differs. So the GUI 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.

Points are remembered per file for as long as the window is open, keyed by a cheap content hash (size plus the first and last megabyte) so that renaming a take does not lose its marks. Running trims every marked file at its own points in one pass, and the badge says how many are queued.

The CLI wrappers trim one file too: the one `AUDION_TRIM_FILE` points at, or the first staged in Source, 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:\T00LS\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/codecs 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 Scrub Plays One Frame, Not One Second

A waveform shows where sound is, not what it says, so the fader sounds. The chunk is one frame long and aligned to the frame boundary, the way an editor scrubs: half a frame of drift is a different syllable

at the head of a word. Three frames are available for when one is too short.

The audio track is decoded into memory whole (mono, 24 kHz) rather than cut from disk per movement - 0.84 ms a slice against 0.35 s of starting a player. Above three hours it is held in two-minute blocks instead; a four-hour reel would be 660 MB, and the fader lives in one part of a file at a time.

Holding a resident FFmpeg was considered and rejected: starting the binary costs 0.02 s. What costs is opening and decoding, and that is what memory now holds.

2026-08-26: Sound Answers Before Picture

The redraw used to decode a still (0.69 s) and re-render the waveform (0.13 s) before playing the slice that costs 0.8 ms - so finding a phrase by ear meant waiting a second for the eyes. The slice now goes first on a shorter debounce and the still follows at its own pace.

The same pass removed two repeated costs: `ffprobe` was asked about an unchanged file on every movement, and the waveform window was re-rendered even when the point had not left it.

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]`.

The same trap exists in the older encode and remux paths; it is noted here so the next person does not rediscover it as a mystery.

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.

Plan with the full reasoning, preview and slider design: `Docs/PLAN_TRIM_RU.md`.

2026-05-04: GUI Shell Over CLI

The GUI layer is a shell over the existing CLI, not a rewrite.

Rationale: Audion tools already have working CMD/FZF workflows. The GUI should make common actions safer and more visible while preserving terminal truth.

2026-05-04: Separate Server And Window

Use:

```
launcher_gui.cmd -> system_core/ui_nicegui/window.py -> app.py --no-browser ->
pywebview
```

Do not default to NiceGUI `native=True`.

Rationale: the separate process model survived the real canary better, avoids browser surprises, and makes port conflicts easier to diagnose.

2026-05-04: Dark Terminal-First Layout

Default layout:

- left: staging, folders, actions, parameters;
- right: status and terminal log;
- terminal log around 2/3 of window height or more.

Rationale: many Audion tools were born as CMD/FZF utilities. Their output is not secondary; it is part of the UX.

2026-05-04: Compact Ghost Buttons

Use flat blue text buttons with subtle hover frame instead of large filled buttons.

Rationale: future tools may have 15-20 commands. Fixed large buttons waste vertical space and break with long Russian labels.

2026-05-04: Operation Buttons Left-Align Their Labels

Command rows use a dedicated operation-button class. Labels are aligned left inside a fixed command column and overflow only to the right with ellipsis. Short toolbar/folder buttons may stay centered.

Rationale: NiceGUI/Quasar `q-btn` can clip centered long labels from both sides, hiding the beginning of the command. Wider windows do not fix the internal clipping.

2026-05-04: Laptop Compression Is A Baseline Requirement

The two-column layout must stay active until about `900px` CSS width and remain usable when compressed toward a WUXGA `1920x1200` laptop at Windows 150% scaling.

Rationale: early wide breakpoints such as `1420px` caused the terminal to fall below the commands. Window size alone does not solve layout resilience.

2026-05-07: 1600x900 Is The Roomy Default Window

pywebview windows start at `1600x900` with a practical `1180x720` minimum. Large forms use compact parameter grids, subdued field borders, stable scrollbar gutters, and a draggable splitter between commands and terminal.

Rationale: real Audion Docs AI usage benefits from a spacious first launch, but the GUI must still be resizable and usable on smaller logical workspaces.

2026-05-07: Form Order Follows User Decisions

Group related choices together and order fields by the user's decision flow rather than by CLI argument order. Provider key and model belong together; rules, instructions, filters, and rare parameters can sit below. Use radio buttons for small fixed one-of-several choices and searchable selects for long or dynamic lists.

Rationale: Audion Docs AI showed that complex LLM/API tools become understandable when the first visible block answers "what access/model will run this?" and secondary constraints do not interrupt

that pair.

2026-05-07: One Visible Action Per User Outcome

Do not expose duplicate actions when one command already produces the complete user-facing result. Also avoid duplicate favorite controls: one explicit action or one checkbox, not both for the same list on the same screen.

Rationale: labels such as "audit" vs "full workflow" or checkbox "favorite" vs button "favorite" created more cognitive load than power. CLI/TUI may keep expert wrappers, but the GUI should present the clean user outcome.

2026-05-07: Actions Must Name Their Object

Avoid detached, objectless commands. If a screen contains several possible targets, the action label must name the target object or be visually bound to exactly one field: `Favorite model`, `Favorite API key`, `Favorite instruction`, `Check model`, `Delete quick instruction`.

Rationale: Audion Docs AI showed that a lone `Favorite` checkbox or button becomes ambiguous as soon as key, model, task instruction, and quick instruction coexist in one form.

2026-05-07: Advanced Fields Collapse By Default

When a form grows beyond the primary decision path, put rare fields such as chunk sizes, retries, timeouts, manual overrides, and strictness toggles into a collapsible advanced block. Persist the block state in GUI settings.

Rationale: Audion Docs AI showed that power settings are useful, but seeing them all the time makes the main action feel further away than it is.

2026-05-07: Model List Is Not Model Access

For API projects, keep dynamic model list refresh separate from selected-model smoke checks. The smoke check is explicit, small, cached, and dated.

Rationale: a provider can list historical models that the current account cannot run. A cached selected-model status prevents confusing access errors with code regressions.

Corollary: avoid hand-curated fixed model ids in project YAML or sidecar list files when a live provider list/cache exists. A developed LLM model selector with cache, favorites, and selected-model smoke status replaces old files such as `models.yaml`, `models.txt`, or static provider profile lists. Static config should hold stable provider settings, env names, prompts, and runtime limits; model ids belong to the GUI cache/favorites/smoke layer.

2026-05-07: Visual Smoke Screenshots Are Part Of Porting

After layout changes, save screenshots of the root screen, a representative command form, any changed advanced state, and the terminal after a short successful run.

Rationale: GUI regressions are often visual and contextual. A screenshot catches crowding, duplicated controls, clipped labels, harsh borders, and missing final status faster than code review alone.

2026-05-04: Stage External Input Locally

GUI should offer `Add files...` and `Add folder...` actions that copy selected items into `input`.

Rationale: safer than long operations directly on network paths, removable drives, or deep user folders.

2026-05-04: Language And Theme Are Conservative

Dark mode is the default. Runtime language switching may reload the UI. Light theme and full international edition are later refinements.

Rationale: live switching caused instability during the canary. Reliability wins.

2026-05-06: Completion State Is Persistent

The right terminal panel keeps a small final-status indicator below the log: idle grey, running blue, success green, failure red.

Rationale: transient notifications can be missed when the GUI window is inactive. The terminal footer is a permanent, low-noise place to show that the last operation really finished.

2026-05-06: Hide Windows CLI Helpers At Process Creation

Windows subprocess helpers, especially `pwsh.exe` and `powershell.exe`, must be hidden through Python process creation flags (`STARTUPINFO/SW_HIDE` and `CREATE_NO_WINDOW`). `-WindowStyle Hidden` may stay as a second layer, but it is not sufficient by itself.

Rationale: PowerShell can briefly create a console window before its own flags take effect. In batch tools this creates distracting per-file flashes. The GUI terminal should mirror output; separate CLI mirror windows should not be the default UX.

2026-05-06: CMD Encoding Is A Build Gate

All project-owned `.cmd` files are UTF-8 without BOM and strict CRLF. The template provides `install\Check-CmdEncoding.cmd -Fix` / `install\Repair-CmdEncoding.ps1 -Fix`, and portable build, offline install, verify, doctor, tests, and release packaging check this automatically.

Rationale: CMD encoding and line endings are too easy to break during GUI porting. The standard should live in the template tooling, not in repeated verbal reminders.