DOCUMENTATION

Technical Specification

RELEASE DATA	
VERSION	2.0
RELEASED	June 2026
AUDIENCE	developers and maintainers

01 Introduction and Reading Guide

Portable Video Compressor (PVC) is a Windows-only .NET MAUI desktop application (single project, WinUI front-end) that compresses one or more videos toward a fixed target size. FFmpeg performs all encoding; FFprobe performs all media inspection. The app iteratively adjusts bitrate - optionally calibrated by a sample encode and refined by per-attempt interpolation - to land inside a configurable leniency band. It ships two ways - as a portable folder that runs in place, and as a setup-program install - with the same self-contained build in both. This document is the complete code-level reference. It assumes no prior contact with the project: a first-time reader should come away knowing what every file, type, and member does and how it does it. It names real identifiers throughout (fields, methods, constants, and controls exactly as they appear in source), and every command line, formula, constant, and message comes straight from the code. How it is organized: sections 2-4 cover the build and the application skeleton; sections 5-6 are member-level references for the Models and Services layers; sections 7-10 explain the encoding engine end to end; sections 11-15 document each page of the UI; sections 16-22 cover the cross-cutting subsystems (format rules, tuning profiles, color, pre-flight, persistence, run control, errors). Nothing is documented twice: when a section needs a fact owned by another, it points there instead of restating it.

02 Build and Runtime Environment

2.1 Project configuration (PortableVideoCompressor.csproj)

- SDK: Microsoft.NET.Sdk; TargetFrameworks: net10.0-windows10.0.19041.0; OutputType Exe; UseMaui + SingleProject; ImplicitUsings and Nullable enabled; AllowUnsafeBlocks enabled solely for the auto-generated WinRT marshalling helpers (CsWinRT1030) - there is no hand-written unsafe code.
- Application metadata: ApplicationTitle "Portable Video Compressor"; ApplicationIcon `Platforms\Windows\PVC.ico`; ApplicationId com.PhotoRevisor.PortableVideoCompressor; ApplicationDisplayVersion 2.0; ApplicationVersion 6.
- Packaging: WindowsPackageType=None (unpackaged WinUI). OS floor: SupportedOSPlatformVersion / TargetPlatformMinVersion 10.0.19041.0, matching the TFM and the [SupportedOSPlatform] attributes in

code.

- SatelliteResourceLanguages: en-US (English resources only).

2.2 Assets and dependencies

- NuGet: Microsoft.Maui.Controls ($(MauiVersion)); Microsoft.Extensions.Logging.Debug 10.0.0 referenced in Debug configuration only (used by AddDebug() inside #if DEBUG).
- MauiFont: Resources\Fonts* (ships OpenSans-Regular.ttf, registered as the OpenSansRegular alias).
- FFmpeg binaries: the project's ffmpeg\ffmpeg.exe and ffmpeg\ffprobe.exe are optional None items - each guarded by Condition="Exists(...)" so the build still succeeds when they are absent - linked to a top-level FFMPEG folder in the build/publish output with CopyToOutputDirectory/CopyToPublishDirectory = PreserveNewest; at runtime the app resolves them from an FFMPEG folder (section 4.5) and shows an FFmpeg-not-found message if neither location exists.
- Content folders copied to output/publish: Licenses** and Documents** (PreserveNewest).
- Portable marker: portable.dat in the project root is copied beside the exe on a normal build/publish (None item, PreserveNewest, guarded by Exists). Publishing with -p:PortableBuild=false omits it, which selects installed mode at runtime (section 20.1); the setup program distributes that marker-less build. In the shipped portable layout the packaging script moves the marker from beside the exe to the folder root (sections 2.3, 20.1).

2.3 Release publish configuration

- RuntimeIdentifier win-x64; SelfContained true (bundled .NET runtime); PublishSingleFile false and PublishTrimmed false for MAUI/WinUI reliability.
- Two release artifacts come from this configuration: the portable folder and a Windows setup executable that installs the -p:PortableBuild=false output.
- Package-Portable.ps1 assembles the portable artifact: dotnet publish for the app, dotnet build for the root launcher (section 3.5), then a staged layout with the whole publish output in App\ and FFMPEG, Documents, Licenses, and portable.dat hoisted to the folder root beside the launcher exe. The script verifies every expected file (and that the marker exists only at the root) before zipping the folder as PortableVideoCompressor-2.0-Portable.zip.

03 Solution Layout

Every source file and its role. Names are exact; paths are relative to the project root.

3.1 Application root

FILE	ROLE
MauiProgram.cs	Builds the MAUI host: App registration, font alias, Windows accent-color copy, debug logging
App.xaml / App.xaml.cs	MAUI Application: theme color aliases, main-window creation, bounds restore/save

FILE	ROLE
AppShell.xaml / .cs	Shell hosting the single MainPage route
ButtonInteractions.cs	Attached behavior: press/release scale bounce on every Button
MainPage.xaml	Main window layout: title bar, five cards, run controls
MainPage.xaml.cs	MainPage fields, constructor, lifecycle, path resolution (section 4.5)
MainPage.Compression.cs	The Start pipeline and the per-file encode loop (section 9)
MainPage.FfmpegRunner.cs	The bitrate-retry engine, sample encode, pass runners (section 8)
MainPage.PreflightChecks.cs	Collision prompt and quality pre-flight (sections 9.1, 19)
MainPage.Settings.cs	Settings persistence hooks, load/capture, Reset to Defaults (section 20)
MainPage.FileSelection.cs	Input picker, dropped-file filtering, queue population (section 12)
MainPage.UiState.cs	Status text, control enablement, elapsed timer, title-link hover (section 11)
MainPage.Encoders.cs	Encoder probe wiring and navigation to Tuning/Queue (section 11.4)
MainPage.PlaybackControls.cs	Pause / Resume / Skip File / Stop Batch handlers (section 21)
MainPage.DragDrop.cs	Native window drag-and-drop, replace semantics (section 12.2)
MainPage.FormatFilters.cs	Audio <-> container picker cross-filtering (section 16.2)
QueuePage.xaml / .xaml.cs	Batch Queue page: rows, overrides, tints, reorder, live view (section 13)
QueuePage.DragDrop.cs	Queue page drag-and-drop, append semantics (section 13.6)
EncoderTuningPage.xaml / .xaml.cs	Tuning page fields, constructor, codec predicates (section 14)
EncoderTuningPage.Options.cs	Per-codec picker option lists (section 14.2)
EncoderTuningPage.Sync.cs	Editor <-> picker synchronization engine (section 14.3)
EncoderTuningPage.PickerEvents.cs	Picker handlers that rewrite the editor text (section 14.3)
EncoderTuningPage.SaveReset.cs	Reset and Done handlers (section 14.4)
QualityWarningPage.xaml / .xaml.cs	Modal pre-batch quality dialog (section 15)
`Launcher\PVC.Launcher.csproj` + Program.cs	Root launcher for the portable layout (section 3.5)
Package-Portable.ps1	Assembles and zips the portable release (section 2.3)

3.2 Services (Services\)

FILE	ROLE
AppSettingsStore.cs	`settings.json` location, load/save, window-bounds save (section 6.1)
FFprobeService.cs	The seven ffprobe metadata probes (section 6.2)
EncoderDetector.cs	Startup encoder discovery and tier ladder (section 6.3)
BitrateCalculator.cs	Bitrate, bits-per-pixel, and Auto-mode math (section 6.4)
FfmpegArgsBuilder.cs	Output path, `-vf` filter, audio and movflags fragments (section 6.5)

FILE	ROLE
FfmpegOptionsParser.cs	Options-string tokenizer for the tuning page (section 6.6)
ColorOptions.cs	The color force block and its strip/apply helpers (section 6.7)
MediaFormatCatalog.cs	Audio/container catalog and compatibility rules (section 6.8)
OutputPathPlanner.cs	Output-path planning and collision resolution (section 6.9)
TempFileHelper.cs	Hidden attempt files, sample and pass-log cleanup (section 6.10)
ProcessSuspender.cs	NT-level whole-process suspend/resume (section 6.11)
Win32FolderPicker.cs	Native IFileOpenDialog folder picker (section 6.12)

3.3 Models (Models\)

FILE	ROLE
EncoderOption.cs	One usable video encoder (picker row)
AudioCodecOption.cs	One audio codec choice
ContainerOption.cs	One output container choice
LabeledOption.cs	FFmpeg value + display label for tuning picker rows
EncoderSettings.cs	One encoder's Preset / Tune / ExtraOptions
EncoderTuningConfig.cs	App-wide tuning registry, default profiles, tiers (section 17)
QueueItem.cs	One queue entry (path + optional overrides)
PerFileOverrides.cs	Per-file override bag
QualityFlag.cs	One quality-warning row
AutoModeSettings.cs	Auto-mode result bundle
PersistedSettings.cs	The `settings.json` schema (section 20.2)
FileOutcome.cs	Per-file result enum
BatchFileResult.cs	One file's recorded result
BatchResult.cs	All results of one run plus aggregation helpers
SourceColorInfo.cs	Probed source color/format facts

3.4 Platform and resources

FILE	ROLE
Platforms\Windows\App.xaml(.cs)	WinUI entry point + single-instance enforcement (section 4.1)
Platforms\Windows\app.manifest	Win32 manifest; assemblyIdentity version 2.0.0.0
Platforms\Windows\Package.appxmanifest	MAUI template manifest (unused in unpackaged builds)
`Platforms\Windows\PVC.ico`	Application icon

FILE	ROLE
Resources\Styles\Colors.xaml	Light/Dark source colors for the theme aliases (section 4.3)
Resources\Styles\Styles.xaml	Control styles, incl. the AccentButtonStyle and the bounce behavior
Resources\Fonts\OpenSans-Regular.ttf	The UI font

3.5 The portable launcher (Launcher\)

A self-contained exe has to sit beside its runtime files, so the shipped portable layout tucks the application into App\ and puts a stub at the folder root instead. `PVC.Launcher.csproj` is a tiny net48 WinExe (the .NET Framework

4.8 runtime is part of Windows 10/11, so nothing extra ships) built as

`PortableVideoCompressor.exe`, carrying the PVC icon and the 2.0 version metadata. Program.Main resolves `App\PortableVideoCompressor.exe` relative to its own location (AppDomain.CurrentDomain.BaseDirectory), which is why moving or renaming the portable folder never breaks it; it starts the target with the working directory set to App\, passes its own command-line arguments through re-quoted, exits as soon as the app is running, and shows an error box naming the expected path when the target is missing.

04 Application Startup and Lifecycle

4.1 WinUI entry and single instance (Platforms\Windows\App.xaml.cs)

- PortableVideoCompressor.WinUI.App is the native entry; CreateMauiApp() hands the host MauiProgram.CreateMauiApp().
- Before XAML initializes, EnforceSingleInstance() acquires a named mutex, "Local\com.PhotoRevisor.PortableVideoCompressor.SingleInstance" (session-local, so each Windows user can run their own instance). The Mutex is held in a static field (_singleInstanceMutex) for the process lifetime so the GC cannot release it early.
- If the mutex already exists, TryActivateExistingWindow() finds the running window by title via FindWindow(null, "Portable Video Compressor") (the title must match ApplicationTitle in the csproj), restores it with ShowWindow(`SW_RESTORE`) when IsIconic reports it minimized, brings it forward with SetForegroundWindow, and the new process exits with Environment.Exit(0). The whole check fails open: if the mutex cannot be created, the app starts normally rather than lock the user out.

4.2 MAUI host (MauiProgram.cs)

- CreateMauiApp() registers the App class and the font alias OpenSansRegular (OpenSans-Regular.ttf).
- On Windows, a lifecycle OnLaunched hook reads the OS accent color through Windows.UI.ViewManagement.UISettings.GetColorValue(UIColorType.Accent) and writes three app resources: AccentColor, AccentColorHover (accent with +0.06 luminosity) and AccentColorPressed (-0.12). Any failure leaves the static Colors.xaml defaults, so the UI still renders.

- Debug builds add Logging.AddDebug() inside #if DEBUG; all diagnostic output in the app goes through Debug.WriteLine and is compiled out of Release behavior paths.

4.3 Theme handling (App.xaml.cs)

- The App constructor calls ApplyThemeColors() and re-runs it on every RequestedThemeChanged, so a live Windows light/dark toggle re-themes the app without restart.
- ApplyThemeColors() copies, for each alias in a fixed list (PageBackgroundColor, PrimaryTextColor, SecondaryTextColor, CardBackgroundColor, CardBorderColor, HoverOverlayColor, ControlBackgroundColor, ButtonDisabledBackgroundColor, ButtonDisabledTextColor), the matching "<alias>Light" or "<alias>Dark" source color from Resources\Styles\Colors.xaml into the alias key. XAML consumes the aliases via DynamicResource, so replacing the key value re-styles every usage instantly.
- TryFindResource(key) exists because Application.Resources does not index merged dictionaries by key on its own; it checks Resources first, then each MergedDictionaries entry.

4.4 Window creation, restore, and save (App.CreateWindow)

- The main window opens at a fitted 800 x 808 (constants defaultWidth/defaultHeight in CreateWindow), which is also the minimum size (minWidth/minHeight). The cards sit at their natural height above the fixed bottom bar with no scroll view, so this is the one correct layout height.
- Size is never restored from settings: only the saved position (WindowX/WindowY) is applied, and only after IsPositionOnScreen(x, y, w, h) validates it. That guard reads DeviceDisplay.Current.MainDisplayInfo, converts to device-independent units (Width / Density), and accepts positions inside a generous envelope: x within (-(w + 2000), screenW + 2000) and y within (-2000, screenH + 2000 - 80). The 2000 margin allows secondary-monitor positions (MAUI exposes only the primary display); the -80 keeps an ~80 px sliver of title bar grabbable. On any exception the guard returns false and the window opens centered at the default size.
- Bounds are saved once, on Window.Destroying (not per resize), through AppSettingsStore.SaveWindowBounds; the store swallows failures so a read-only location never breaks shutdown. MainPage adds its own Destroying work (section 20.4).

4.5 MainPage construction (MainPage.xaml.cs constructor)

The constructor runs, in order:

- Initial control state: StartButton, OutputFolderButton, ViewQueueButton, PauseResumeButton, SkipFileButton, and StopBatchButton disabled; StatusLabel = "Ready."; ProgressBar visible at 0.
- FFmpeg/FFprobe path resolution. Two candidate directories are built from AppContext.BaseDirectory:

```
baseDir\FFMPEG          (dev / flat layout, beside the exe)
baseDir\..\FFMPEG       (one level up: the shipped App-subfolder layout)
```

Array.Find takes the first candidate that actually contains `ffmpeg.exe`; if neither does, the beside-the-exe candidate is assumed so the not-found message points somewhere sensible. _ffmpegPath and _ffprobePath are then `<dir>\ffmpeg.exe` and `<dir>\ffprobe.exe`. Folder matching is case-insensitive on Windows, so an older

lowercase ffmpeg folder still resolves.

- EncoderPicker placeholder: a single row reflecting ffmpeg presence ("Software (AVC/H.264)" when `ffmpeg.exe` exists, else "No FFmpeg found"), so a launch without FFmpeg never briefly shows a real encoder before the probe swaps the list in.
- AudioCodecPicker and ContainerPicker are seeded from MediaFormatCatalog (index 0 each: "Same as source (no re-encode)" and MP4), and their SelectedIndexChanged handlers call ApplyCompatibilityFilter (section 16.2).
- SampleEncodeSwitch starts off; UpdateAutoModeUi() runs; the AutoModeSwitch Toggled handler re-runs it; the Leniency slider's initial value seeds _targetGraceFraction and the TargetGraceLabel text.
- HookSettingsPersistence() then LoadAndApplyPersistedSettings() (section 20.3).
- Three Loaded handlers defer the rest to first display: EnsureEncodersInitializedAsync() (the encoder probe, section 11.4), HookWindowPersistence() (section 20.4), and HookDragAndDrop() (section 12.2).

Lifecycle notes: OnDisappearing deliberately does not cancel a running encode (it fires on any navigation, e.g. opening the Queue page mid-run); shutdown teardown lives on Window.Destroying. OnAppearing re-syncs labels and button enablement when returning from another page, re-derives the Preserve-color switch from the tuning profiles (section 18), persists, and rebuilds the input label ("N files selected (first: <path>)" for multi-file queues); it is skipped entirely while IsStartingOrEncoding.

4.6 Process and threading model

- All FFmpeg/FFprobe work runs off the UI thread (Task.Run); UI updates are marshalled back with MainThread.BeginInvokeOnMainThread or the page Dispatcher.
- Exactly one FFmpeg process runs at a time. Its handle is published to the _currentFfmpegProcess field under _processLock only after a successful Start, so Pause/Skip/Stop always target a real OS handle; the handle is cleared in a finally block when the pass ends.
- Two CancellationTokenSource layers exist per run: _batchCts (Stop Batch) and _fileCts (Skip File), combined per file with CreateLinkedTokenSource so the encoder receives one token covering both. Cancellation kills the FFmpeg process tree (Kill(entireProcessTree: true)).
- ffprobe probes are bounded by a watchdog rather than async reads (section 6.2), because stderr is deliberately not redirected anywhere in the app.

05 Data Model Reference (Models\)

Every public type and member. All types are sealed; option types are immutable (constructor-set, get-only), state types use settable properties.

5.1 Picker option types

- EncoderOption: DisplayName (UI label, e.g. "Software (AVC/H.264)"), FfmpegName (the `-c:v` id, e.g. `libx264` or `h264_nvenc`), SupportsTwoPass (whether two-pass works for this encoder here; true only for the software encoders).

- AudioCodecOption: Id (catalog key: "source", "aac", "mp3", "wav"), DisplayName, FfmpegCodec (the `-c:a` id; empty when IsCopy), IsCopy (true remuxes source audio without re-encoding). ToString() returns DisplayName for picker binding.
- ContainerOption: Id ("mp4", "mov", "mkv"), DisplayName, Extension (with the dot), OverheadFactor (fraction of the size budget left after container framing; values in section 16.1), SupportsFaststart (MP4/MOV true, MKV false). ToString() returns DisplayName.
- LabeledOption: Id (an FFmpeg option value) + Display; binds the tuning page's picker rows.

5.2 Queue and overrides

- QueueItem: InputPath (immutable) + Overrides (PerFileOverrides?, settable; null until the user edits the file, keeping memory low for large batches).
- PerFileOverrides: TargetSizeMb (double?), Width, Height, Fps (int?), EncoderFfmpegName, AudioCodecId, ContainerId (string? stable ids so overrides survive re-ordering). A null field means "use the batch default". HasAny is true when any field is set (drives the queue's marker); ClearAll() nulls every field.

5.3 Results

- FileOutcome enum: Succeeded (output exists at OutputPath), Failed (reason in ErrorMessage), Skipped (never processed: already at/under target), Cancelled (user skipped it mid-run; the batch continued).
- BatchFileResult: InputPath, Outcome, OutputPath (Succeeded only), OriginalBytes, FinalBytes, ErrorMessage, FfmpegStderrTail (bounded tail of the failing pass's stderr; error log only, never the summary dialog), EffectiveTargetBytes + GraceFraction (the target and leniency in force when recorded; zero means no snapshot), CompletedAt (DateTime.Now at record time), and BytesSaved (OriginalBytes - FinalBytes for successes, else 0; can be negative).
- BatchResult: Results (list in processing order) plus derived counts (TotalCount, SucceededCount, FailedCount, SkippedCount, CancelledCount), TotalBytesSaved, and filtered views (Failures, Skips, Cancellations, Successes). Three per-path lookups walk Results backwards so the most recent record wins, case-insensitively: OutcomeForPath, FinalBytesForPath (successes only), and SnapshotForPath (returns the (targetBytes, grace) pair when EffectiveTargetBytes > 0). The Queue page tints and labels rows from these (section 13.4).

5.4 Quality, color, auto

- QualityFlag: Item (the same QueueItem instance the queue and encode loop use, so a dialog edit writes straight onto Item.Overrides with no sync step), InputPath, TargetMb (effective target), MinPassableMb (the estimate from section 19).
- SourceColorInfo: the raw ffprobe tokens PixFmt, ColorRange, ColorSpace, ColorPrimaries, ColorTransfer (each nullable). Derived flags: Is10BitOrHigher (PixFmt contains "10", "12", or "16"), Is422, Is444 (substring checks), and `IsHevcUnfriendly422_10Bit` (both 4:2:2 and 10-bit: the combination consumer NVENC/AMF hardware cannot encode, driving the targeted message in section 22). IsUsable(token) treats "unknown", "N/A", "reserved", and blank as missing.

- AutoModeSettings: Width (int?, null means derive from height), Height, Fps, AudioBitrateKbps, CopyAudio; the bundle Auto mode derives (section 6.4) and the encode loop consumes.

5.5 Settings and tuning state

- PersistedSettings and EncoderTuningDto: the `settings.json` schema; documented field by field in section 20.2.
- EncoderSettings: Preset, Tune, ExtraOptions (all strings, default empty). One mutable container per codec, edited by the tuning page.
- EncoderTuningConfig: the app-wide registry; documented in section 17.

06 Service Reference (Services\)

6.1 AppSettingsStore

Static; owns where `settings.json` lives and how it is read and written.

- ResolveSettingsPath() runs once into the SettingsPath field, checking three locations in order: `portable.dat` (PortableMarkerFileName) beside the exe (AppContext.BaseDirectory) keeps `settings.json` beside the exe (a flat portable layout); the marker one level above the exe keeps `settings.json` at that parent folder - the root of the shipped App-subfolder layout (section 2.3); with no marker in either place, `settings.json` lives under %AppData%\PortableVideoCompressor (Environment.SpecialFolder.ApplicationData; the folder is created on demand), because an installed location such as Program Files is not writable for a normal user. Any exception falls back to beside the exe.
- Load() returns the deserialized PersistedSettings, or null when the file is missing, empty, or unreadable (callers then keep defaults). JsonSerializerOptions: WriteIndented, PropertyNameCaseInsensitive.
- Save(settings) serializes to SettingsPath + ".tmp" and then File.Move(temp, SettingsPath, overwrite: true), so an interrupted write can never corrupt `settings.json`. All failures are logged to Debug and swallowed: a read-only location must never break compression.
- SaveWindowBounds(x, y, w, h) loads the current file (or starts a fresh PersistedSettings stamped with the current SchemaVersion), overwrites only the four bounds fields, and saves. The window exists before any page, which is why this path re-reads instead of asking the page for a snapshot.

6.2 FFprobeService

Seven probe helpers, one fact each. All share the same execution contract:

- ffprobe is run with `-v` error; stdout is redirected and read synchronously with ReadToEnd; stderr is never redirected (an undrained redirected pipe can deadlock ffprobe); CreateNoWindow is set.
- Every probe is bounded by StartKillTimer(proc): a one-shot System.Threading.Timer that fires after ProbeTimeoutMs = 30_000 (30 s) and kills the process tree. Killing closes the pipes, which unblocks the synchronous read; the parse path then returns the probe's "unknown" value, so a corrupt file fails cleanly per file instead of hanging the run. The timer is disposed by its using scope when the probe finishes normally.

PROBE METHOD	FFPROBE SELECTION	UNKNOWN VALUE
GetVideoDurationAsync	`-show_entries` format=duration	-1
GetVideoFpsAsync	`-select_streams` v:0 `-show_entries` stream=`r_frame_rate`	-1
GetVideoResolutionAsync	`-select_streams` v:0 `-show_entries` stream=width,height (csv, "x" separator)	(0, 0)
GetVideoColorInfoAsync	`-select_streams` v:0 `-show_entries` stream=`pix_fmt`,`color_range`,`color_space`,`color_primaries`,`color_transfer`	all-null SourceColorInfo
GetAudioCodecNameAsync	`-select_streams` a:0 `-show_entries` stream=`codec_name`	null
GetAudioBitrateKbpsAsync	`-select_streams` a:0 `-show_entries` stream=`bit_rate`	null
GetAudioSampleRateAndChannelsAsync	`-select_streams` a:0 `-show_entries` stream=`sample_rate`,channels	(0, 0)

- Output writers are default (nokey / `noprint_wrappers`) or csv, so the output is bare values needing no scraping. Numeric parses try invariant culture first, then current culture.
- The frame-rate probe parses ffprobe's rational form: an output containing "/" is split into numerator/denominator and divided (a zero denominator fails); a plain number parses directly.
- The color probe splits key=value lines and fills the five SourceColorInfo fields it recognizes.

6.3 EncoderDetector

Discovers which encoders the bundled FFmpeg can actually use on this machine.

- DiscoverEncodersAsync(ffmpegPath) clears and repopulates SafeDefaultOverrides (a case-insensitive Dictionary<string, string> exposed read-only), then builds the picker list. Software AVC ("Software (AVC/H.264)", `libx264`, SupportsTwoPass=true) is added without probing as the universal fallback; `libx265`, `h264_nvenc`, `hevc_nvenc`, `h264_amf`, and `hevc_amf` are each added only when ProbeBestTierAsync passes (display names "Software (HEVC/H.265)", "GPU - NVENC (AVC/H.264)", "GPU - NVENC (HEVC/H.265)", "GPU - AMF (AVC/H.264)", "GPU - AMF (HEVC/H.265)"; SupportsTwoPass=false for all GPU encoders). If `ffmpeg.exe` is missing entirely, the single placeholder row "No FFmpeg found" is returned (its internal name stays `libx264` so opening the Tuning page in that state cannot break; compression is blocked separately by the presence check in section 9.1).
- ProbeBestTierAsync walks the tier ladder full -> conservative -> minimal (tier contents in section 17.2), skipping any tier whose whitespace-normalized options equal an already-failed tier. The first passing tier wins; when it is not the full profile, its options string is recorded in SafeDefaultOverrides[codec]. Returning false means the encoder is unusable and is left out of the picker.
- IsEncoderUsableAsync runs the actual probe: a 1-second synthetic encode to the null muxer,

```
ffmpeg -hide_banner -loglevel error
  -f lavfi -i testsrc2=duration=1:size=1280x720:rate=30
  -c:v <encoder> <profile options> -f null NUL
```

bounded by EncoderProbeTimeoutMs = 20_000 (20 s; a healthy probe takes about a second, only a wedged driver reaches the cap). On timeout the tree is killed and the tier reports unusable instead of hanging startup. stderr is not redirected (the probe never reads it and an undrained pipe could deadlock); only the exit code matters; no file is produced.

- PickPreferredEncoderIndex returns the picker default: `h264_nvenc`, else `hevc_nvenc`, else `h264_amf`, else `hevc_amf`, else index 0 (software AVC), so machines with a GPU default to hardware encoding.

6.4 BitrateCalculator

Pure math: no IO, no UI, deterministic from its inputs.

- CalculateVideoBitrate(targetMb, durationSeconds, audioBitrateKbps, containerOverheadFactor) converts the target into a video bitrate:

```
targetBits = targetMb * 8,388,608          (1 MB = 8 * 1024 * 1024 bits)
totalKbps  = targetBits / durationSeconds / 1000 * containerOverheadFactor
videoKbps  = totalKbps - audioBitrateKbps   clamped to [1, 1_000_000]
```

The overhead factor reserves framing headroom so the first estimate does not overshoot; each ContainerOption carries its own factor (section 16.1) and DefaultContainerOverheadFactor = 0.985 is the fallback (also used when a pathological factor outside (0, 1] is passed). A non-positive duration returns 1000 kbps.

- ComputeBitsPerPixel(totalBitrateKbps, audioBitrateKbps, targetHeight, targetFps, srcWidth, srcHeight) scores a candidate: the implied width is srcWidth scaled by targetHeight/srcHeight and snapped to even (Math.Round(w/2)*2), then

```
bpp = (totalKbps - audioKbps) * 1000 / (width * height * fps)
```

Zero is returned for any non-positive input or a non-positive video share.

- ChooseSampleDurationSeconds(fullDurationSeconds) picks the sample-encode window: 25% of the clip (fraction = 0.25), clamped to [minSample 4 s, maxSample 12 s], and never more than half the clip; degenerate results (<= 0, or >= the full duration) return 0, meaning no sample. Whether a sample runs at all is a loop decision (section 9.2).
- ComputeAutoModeSettings(totalBitrateKbps, srcFps, srcWidth, srcHeight, originalAudioKbps, videoCodec) is the Auto-mode search:
  - Baselines: non-positive inputs become 500 kbps total, 30 fps, 1280x720, and DefaultAudioBitrateKbps = 96 audio.
  - Frame-rate range: [24, srcFps] when the source is at least 24 fps; a slower source is both floor and ceiling. Candidates, descending: the source rate, then each of 120 / 90 / 60 / 50 / 48 / 30 / 25 / 24 inside the range (curated so output avoids odd rates like 35 or 55), then the floor itself.
  - Height candidates, descending: the source height, then each of 2160 / 1440 / 1080 / 720 / 480 that is below the source and at least the floor (480, or the source height when the source is smaller).

- Audio budget: the ceiling is max(originalAudioKbps, MinAudioKbps = 64); MinVideoReserveKbps = 150 is held back for video, so audio gets min(ceiling, totalKbps - 150), and is pinned at the 64 floor whenever totalKbps - 150 is at or below 64.
- Quality floor per codec family: minVideoBpp = 0.028 for `libx265` / `hevc_nvenc` / `hevc_amf`, 0.045 for everything else (HEVC's lower floor keeps more resolution and frame rate at small targets).
- The search walks heights outer, frame rates inner; the first (height, fps) whose ComputeBitsPerPixel clears the floor wins, so resolution is preferred over frame rate. If nothing clears the floor, the smallest allowed height and fps are used.
- Result: AutoModeSettings with Width = null (the encode filter derives it via `-2`), the chosen Height and Fps, the audio kbps, and CopyAudio = false.

6.5 FfmpegArgsBuilder

Pure fragment builders shared by every encode path.

- BuildOutputPath(inputPath, outputFolderOverride, containerExtension) yields <name>_compressed<ext> in the override folder, else the input's folder, else the user's Videos folder (SpecialFolder.MyVideos); a blank extension defaults to "`.mp4`" and a missing dot is added.
- BuildVideoFilterString(widthOpt, heightOpt, fpsOverride) joins the needed parts with a comma, or returns null so `-vf` is omitted entirely:

```
scale=W:H     both set        scale=W:-2    width only
scale=-2:H    height only     fps=N         appended when an fps is set
```

`-2` for the unspecified axis keeps the aspect ratio and guarantees even output dimensions.

- BuildAudioArgs(audioCodec, audioBitrateKbps): IsCopy maps to `-c:a` copy; WAV maps to `-c:a pcm_s16le` with no `-b:a` (PCM's rate follows sample rate, channels, and bit depth); every other codec gets `-c:a` <codec> `-b:a` <N>k.
- BuildMovflagsArgs(container) returns "`-movflags` +faststart" when SupportsFaststart, else empty. Faststart moves the moov atom to the front so playback can start before full download at no size cost; MKV omits it.
- TryParseFfmpegTime(text, out seconds) parses FFmpeg's HH:MM:SS.sss into seconds (three colon-split parts; invariant-culture seconds), returning false on malformed input. The pass runners use it for progress (section 7.4).

6.6 FfmpegOptionsParser

Tokenizer between the tuning page's free-text editor and its pickers.

- RecognizedKeys (case-insensitive): preset, tune, rc, multipass, quality, usage.
- ParseNamedOptions(text, out otherTokens) whitespace-splits the text; a token starting with "-" whose dash-stripped key is recognized consumes the following token as its value (unless that token also starts with "-", leaving the value empty); everything else lands in otherTokens verbatim.

- BuildOptionsString(named, otherTokens) re-emits recognized keys in a fixed order (preset, tune, rc, multipass, quality, usage) followed by the pass-through tokens, dropping empty-valued keys, so setting a picker to None removes its flag and the text never reshuffles on edits.

6.7 ColorOptions

Owns the color/format flags PVC manages inside each profile's options string.

- ForceBlock, applied when Preserve color is OFF (8-bit 4:2:0, BT.709 limited range):

```
-pix_fmt yuv420p -colorspace bt709 -color_primaries bt709
  -color_trc bt709 -color_range tv
```

- OwnedFlags: `pix_fmt`, `pixel_format`, colorspace, `color_primaries`, `color_trc`, `color_range`.
- Strip(options) removes each owned flag and one following value with the regex pattern \s*-<flag>(?=\s|$)(?:\s+[^-\s]\S*)?; the boundary lookahead stops a flag matching inside a longer user token (e.g. `-pix_fmts`), and runs of whitespace are collapsed afterward.
- HasForce(options) is true when an explicit `-pix_fmt` or `-pixel_format` flag is present.
- Apply(options, preserve): preserve=true returns the stripped string; preserve=false returns the stripped string with exactly one ForceBlock appended. The options strings are the single source of truth for the toggle (section 18).

6.8 MediaFormatCatalog

The audio/container catalog and every compatibility rule; documented with its tables in section 16 so the data lives in one place. Members: string id constants (CodecIdSource/Aac/Mp3/Wav, ContainerIdMp4/Mov/Mkv), AllAudioCodecs() and AllContainers() (picker-ordered lists), IsCompatible(codecId, containerId), CanCopyAudioInto(sourceAudioCodecName, containerId), DescribeCopyAlternatives(sourceAudioCodecName), and GetDefaultLossyBitrateKbps(codecId).

6.9 OutputPathPlanner

Plans every file's final output path before the batch starts, so collisions are found up front and the user is prompted once. Pure: no UI, no IO beyond guarded File.Exists.

- ResolveEffectiveContainer(item, batchContainer): the per-file container override when set and known, else the batch container.
- ComputeDesiredOutputPaths(queue, outputFolderOverride, batchContainer) maps every item through BuildOutputPath with its effective container's extension.
- DetectCollisions(desiredOutputs) groups paths case-insensitively (Windows) and flags, in one pass, paths that already exist on disk and paths produced by two or more entries. The CollisionReport carries one CollisionEntry per conflicting path (Path, InBatchIndices, ExistsOnDisk) plus HasOnDiskConflicts / HasInBatchConflicts / HasAny, so the prompt can word itself correctly.
- ResolveFinalOutputPaths(desiredOutputs, overwriteExisting): overwrite mode returns the desired paths verbatim (ffmpeg `-y` overwrites; in-batch duplicates overwrite in order); rename mode walks each taken path

through ResolveSingleRenumberedPath, which tries the bare path, then name1, name2, ... (bounded at 1,000,000 as a sanity guard) until a candidate is neither on disk nor already claimed by this batch, so two entries wanting the same name cannot both pick 1.

- SafeFileExists wraps File.Exists so a transient IO error reads as "absent" rather than failing the whole pre-flight.

6.10 TempFileHelper

Manages transient encoder files. Every operation swallows IO errors: cleanup must never abort an encode.

- HideFileIfExists / EnsureNotHidden set and clear the Hidden file attribute; in-progress attempt files are hidden, and the promoted output is un-hidden.
- TryDeleteFile deletes one file if present (used to drop a superseded attempt immediately); a locked file is simply left for the end-of-run sweep.
- TryDeletePassLogFiles(directory) removes FFmpeg's two-pass stats files by exact name: ffmpeg2pass-0.log (+ .mbtree, and both .temp forms) for `libx264`, and `x265_2pass.log` (+ .cutree and .temp forms) for `libx265`, which ignores `-passlogfile` and names its own logs.
- DeleteTempOutputs(directory, baseName, keepPath, extension) sweeps attempt files matching the exact shape ^<baseName>_tmp\d+<ext>$ (regex-anchored so a user file that merely resembles a temp, e.g. `video_compressed_tmp_backup.mp4`, is never swept), optionally keeping one path (used when a locked destination forces keeping the best attempt).
- DeleteSampleFile(directory, baseName, extension) removes <baseName>_sample<ext>.

6.11 ProcessSuspender (Windows-only, #if WINDOWS)

- Wraps the undocumented ntdll exports NtSuspendProcess and NtResumeProcess, the only way to suspend/resume a whole process (all threads) without killing it; no managed API exists. SuspendProcess/ResumeProcess pass Process.Handle and log a nonzero NTSTATUS to Debug. Used by Pause/Resume (section 21).

6.12 Win32FolderPicker

Native folder picker replacing MAUI's WinRT FolderPicker, which cannot select the folder currently being viewed and cannot open at an arbitrary path.

- PickFolder(ownerHwnd, initialFolder, title) COM-creates the FileOpenDialog coclass (CLSID DC1C5A9C-E88A-4DDE-A5A1-60F82A20AEF7) through a hand-declared IFileDialog interface (the vtable order matches `shobjidl_core.h` exactly; unused methods are still declared to keep the slots aligned), ORs in `FOS_PICKFOLDERS` | `FOS_FORCEFILESYSTEM` | `FOS_PATHMUSTEXIST` | `FOS_NOCHANGEDIR`, sets the title, and, when initialFolder exists, forces the start location past any remembered MRU with SHCreateItemFromParsingName + SetFolder.
- Show(ownerHwnd) returning `S_OK` yields the selection; GetResult + GetDisplayName(`SIGDN_FILESYSPATH`) marshal the path (CoTaskMem freed in a finally). Cancel returns null. The dialog runs synchronously on the calling STA thread and the COM object is released with FinalReleaseComObject.

07 FFmpeg Command Assembly

The one place the generated command lines live; every encode path composes from the fragments in sections 6.5 and 7.1.

7.1 Video codec arguments (EncoderTuningConfig.BuildVideoCodecArgs)

- The encoder's options string is authoritative and inserted verbatim (color flags included; newlines collapsed to spaces); only rate control is added outside it, with a conventional VBV buffer at twice the bitrate (bufsize capped at int.MaxValue; a bitrate below 1 is raised to 1):

```
-c:v <codec> <options string> -b:v <N>k -maxrate <N>k -bufsize <2N>k
```

A codec with no registered settings gets just `-c:v` <codec> plus the rate control.

7.2 The generated command lines

- Single pass (software 1-pass and all GPU encodes):

```
ffmpeg -y -i "<input>" [-vf "<filter>"] <codec args> <audio args>
  [-movflags +faststart] "<output>"
```

- Two pass (`libx264`/`libx265` with 2-pass on). Pass 1 analyzes to the NUL device with audio disabled; pass 2 muxes the real output:

```
ffmpeg -y -i "<input>" [-vf "<filter>"] <codec args> -pass 1 -an -f mp4 NUL
ffmpeg -y -i "<input>" [-vf "<filter>"] <codec args> -pass 2 <audio args>
  [-movflags +faststart] "<output>"
```

- Sample encode (single pass over a mid-clip slice; `-ss`/`-t` formatted invariant with 0.### precision):

```
ffmpeg -y -ss <start> -t <len> -i "<input>" [-vf "<filter>"] <codec args>
  <audio args> "<name>_sample<ext>"
```

Sample placement: start = min(half the clip, duration - sample length), i.e. roughly the middle, which is usually representative and avoids credits.

- Every encode's ProcessStartInfo sets UseShellExecute=false, RedirectStandardError=true (the only redirected stream; progress and diagnostics are parsed from it), CreateNoWindow=true, and WorkingDirectory = the output folder, so FFmpeg's own two-pass log files (section 6.10) land there and are swept by cleanup.

7.3 Pass execution and progress (RunFfmpegPassAsync / RunFfmpegSinglePassAsync)

- Each runner starts the process, publishes the handle under _processLock, and, when the run is paused at that moment, immediately suspends the new process (startedSuspended) so a pause taken at a pass or file

boundary still takes effect; the status then reads "Paused.".

- The stderr loop reads line by line until exit: each line feeds the diagnostics tail (section 8.6), and lines containing time= are parsed with TryParseFfmpegTime into a fraction of the known duration, clamped to [0, 1] and mapped into the overall range as ((pass - 1) + fraction) / totalPasses, so pass 1 of 2 covers 0.0-0.5. A cancellation request inside the loop kills the process tree and exits.
- On exit the runner reports pass/totalPasses as the progress floor and returns the exit code; the handle is cleared in a finally block.

08 The Encode Attempt Loop (MainPage.FfmpegRunner.cs)

EncodeWithBitrateRetryAsync owns one file's search for a size inside the band. Its inputs are the resolved effective settings from section 9.2; its output is a bool (true only when the final move in 8.5 succeeded).

8.1 The leniency band

- Band: [grace x target, target] bytes, where grace is the Leniency slider fraction (_targetGraceFraction); its center is the interpolation goal:

```
lower  = round(targetBytes * goodLowerFraction)
upper  = targetBytes
center = (lower + upper) / 2
```

8.2 Sample calibration (when enabled, section 9.2)

- The sample window comes from ChooseSampleDurationSeconds (section 6.4); the sample is encoded single-pass at the initial bitrate to <name>_sample<ext> and its measured size rescales the starting total bitrate toward the band center:

```
refinedTotalKbps = center * totalKbps * sampleLen
                   / (sampleBytes * duration * twoPassFactor)
```

- twoPassFactor is 1.04 when the real encode will be software two-pass (libx264/libx265): two-pass packs roughly 3-5% more data at a given bitrate than the single-pass sample, so the estimate is nudged up. NVENC/AMF show no such bias, so their factor is 1.0.
- The sample file is deleted in every path (success, failure, cancel).

8.3 The attempt cycle

- Hard cap maxAttempts = 12; stagnation normally stops the search first, the cap only bounds rare oscillation. Each attempt encodes to <name>_tmp<n><ext>, hidden (file attribute) while it is a candidate; a two-pass attempt runs pass 1 then pass 2 and breaks out on any nonzero exit or cancellation (recording diagnostics per section 8.6).

- Disk discipline: only the best under-target (closestUnderPath) and best over-target (closestOverPath) attempts stay on disk; a displaced best or a non-improving attempt is deleted immediately (TryDeleteFile). The latest under and over points (lastUnder/lastOverVideoBitrate + sizes) are kept as plain numbers, so deleting files never affects the search.
- Stop conditions, in order: result inside the band; stagnation (below); attempt cap; a computed next bitrate equal to the current one (forces the stagnation stop); cancellation.
- Stagnation: relative improvement toward the band center, (lastDelta - delta) / lastDelta, below minRelativeImprovement = 0.10 for maxStagnationSteps = 2 consecutive attempts stops the search; a good step resets the counter.

8.4 Choosing the next bitrate

- Both sides known (an under and an over result exist at different bitrates and sizes): fit a line through the two latest points (total kbps including audio vs bytes) and solve it for the band center:

```
m = (sizeOver - sizeUnder) / (kbpsOver - kbpsUnder)
q = sizeUnder - m * kbpsUnder
nextTotalKbps = (center - q) / m
```

- One side known: pick the recorded point closest to the center (the current attempt, the best under, or the best over) and scale it proportionally:

```
nextTotalKbps = refTotalKbps * (center / refSizeBytes)
```

- The video part (next total minus audio) is clamped to [1, 1_000_000] kbps (maxVideoBitrate), then the per-step swing is clamped to [minScaleDown 0.25x, maxScaleUp 4.0x] of the current bitrate, guarding against noisy size-to-bitrate behavior on a given clip.

8.5 Finalization and promotion

- On completion the best under-target attempt is preferred; the best over-target attempt is the fallback. If neither exists, all temp/sample/pass-log artifacts are swept and the file fails with the status "Unable to produce output at or below target size."
- The chosen attempt is promoted by an explicit move onto the final path (deleting a pre-existing final first; File.Move with overwrite: false), then un-hidden. Success is reported only when this move actually happens, never inferred from a file merely existing at the destination.
- If the move throws (destination locked or permission denied), the chosen attempt is un-hidden and kept beside the output under its _tmpN name, everything else is swept, the file records the locked-output message (section 22.1), and the status reads "Encoding failed: output file is locked or in use."
- Cancellation deletes every temp, sample, and pass-log artifact and returns false: a cancelled file leaves nothing behind.

8.6 Failure diagnostics

- A rolling stderr tail (Queue<string> _stderrTail, StderrTailMaxLines = 25, lock-guarded) is reset per pass and fed by CaptureStderrLine, which skips per-frame progress lines (starting with frame= or containing time=) so real errors are not flushed out. On a nonzero pass exit, RecordEncodeFailureDiagnostics snapshots the tail into _lastEncodeFailureTail for the batch log.
- When the source is 4:2:2 10-bit (SourceColorInfo.`IsHevcUnfriendly422_10Bit`) and a hardware encoder fails, the tail is matched case-insensitively against HwFormatInitFailureSignatures ("10 bit encode not supported", "no capable devices found", "openencodesessionex failed", "provided device doesn't support", "initializeencoder", "amf failed to initial", "incompatible pixel format", "impossible to convert between the formats", "not supported"); on a match the recorded reason becomes the targeted 4:2:2 10-bit message (section 22.1) instead of raw FFmpeg text.

09 The Start Pipeline and Per-File Sequence (MainPage.Compression.cs)

9.1 OnStartClicked, in order

- Re-entrancy: the handler exits if IsStartingOrEncoding; otherwise _startRequested flips true immediately and RunStateChanged is raised, so every mirrored Start button grays for the whole locked window (press through preflight through run) instead of blinking during the preflight. AbortStartReenable() undoes this for any validation exit below.
- Empty-queue guard ("Missing File"), then the batch membership is frozen: runItems = _queue.ToList(). Every later step iterates this snapshot, so a queue edit during the non-modal preflight cannot shift files onto another item's planned output path.
- Platform guard (non-Windows shows "Unsupported"), then the ffmpeg/ffprobe presence check; its dialog names the `FFMPEG` folder (message text in section 22.1).
- EnsureEncodersInitializedAsync() is awaited (a no-op after first launch) and PersistSettings() checkpoints the on-screen state.
- Target parsing: blank falls back to DefaultTargetSizeMb = 10; otherwise invariant-then-current-culture double parse, rejecting non-numbers and values <= 0 ("Invalid value"). targetBytes = targetMb * 1,048,576; targetBits = targetMb * 8,388,608.
- Nothing-to-do check: every runItem's on-disk size is compared against its effective target (ResolveEffectiveTargetBytes: per-file override else batch). When every checkable file is already at or under target, the run aborts with the "Nothing to do" dialog. The same pass accumulates the projected output bytes of the files that will run.
- Low-disk-space check: the projected bytes plus a 25% margin are compared with the output drive's free space (ResolveOutputDriveRoot: the explicit output folder's root, else the first input's root; DriveInfo.AvailableFreeSpace, skipped when unqueryable). When the projection exceeds free space, a Continue/Cancel dialog states both figures and warns the run may fail partway.
- Manual mode only: FPS/Height/Width parse (blank = leave alone; "Invalid value" on non-positive input), with odd dimensions snapped down to even (h = Math.Max(2, h - 1)), the Entry rewritten so the user sees the snapped value, and one status note: "Adjusted manual Width/Height to even values (required by the video

encoders)."

- Encoder resolution (selected picker row, else the first discovered, else a `libx264` fallback object) and the two-pass decision: forced on in Auto Mode for capable encoders, otherwise SupportsTwoPass AND the switch state. Batch audio/container come from the pickers (defaulting to catalog index 0 if a picker is stale).
- Preflight 1: RunCollisionPromptAsync (section 9.3) returns finalOutputPaths, one per runItem.
- Preflight 2: RunQualityWarningAsync (section 19) returns Cancelled (abort, status "Cancelled.") or the per-file skip set (qualitySkipPaths).
- Enter the run: fresh _batchCts; _fileCts cleared; _isEncoding = true; pause/elapsed state reset (_totalPausedTime zero, _encodeStartTime = now, elapsed ticker started, per-file times cleared); controls locked (Start/Select/Output/slider disabled, UpdateAutoModeUi() disables the rest, Pause/Skip/Stop enabled); ProgressBar reset to 0; a Progress<double> is created that drives the bar and republishes through UpdateRunState so the Queue page mirrors it; _currentBatchResult = new BatchResult().

9.2 The per-file loop

For each index in runItems (breaking when the batch token fires):

- A fresh _fileCts is created and linked with the batch token (one token covers Skip and Stop); SkipFileButton is re-enabled.
- Effective settings resolution: batch defaults for target (Mb/bytes/bits), encoder, two-pass, audio, container, and the user FPS/W/H are copied, then each per-file override replaces its value. An encoder override also recomputes two-pass eligibility against the override (Auto forces on; manual uses the durable _manualTwoPassChoice).
- The InputPathLabel becomes "File <i+1> of <total>[*]: <path>" (the * marks overridden files); one line so middle truncation keeps the prefix and filename visible, full path on the hover tooltip.
- Quality-dialog skips: a path in qualitySkipPaths is recorded Cancelled with "Skipped by user (quality warning)." before anything else, so the explicit choice wins even over the under-target probe.
- Under-target skip: a file already at or under its effective target is recorded Skipped (re-encoding could only grow it).
- BeginFileTiming starts this file's pause-excluded clock (section 21.3); a missing or unreadable input records Failed here.
- Fail-fast format guards: with "Same as source", the probed source audio codec is checked by CanCopyAudioInto and a veto records Failed with the copy-incompatible message; an explicit codec is checked by IsCompatible against the container (a stale per-file override can produce this) and records Failed with the mismatch message. Both messages name working containers (section 22.1) and neither launches FFmpeg.
- Probes: duration (a non-positive value fails the file: "Could not read video duration (ffprobe returned no valid duration)."), frame rate (rounded; fallback 30), resolution (fallback 1280x720), color info (drives the 4:2:2 10-bit flag), and audio bitrate (fallback DefaultAudioBitrateKbps = 96).
- totalBitrateKbps = effectiveTargetBits / duration / 1000 (a non-positive result fails the file). The codec-preferred audio bitrate is: the probed source bitrate for "Same as source"; the PCM rate sampleRate x

channels x 16 / 1000 for WAV (fallbacks 44100 / 2; floor guard 1411); else the catalog's lossy default (section 16.1).

- Auto mode: first the impossible-target rejection: minimum audio for the check is 64 kbps (or the full PCM rate for WAV), and when that alone exceeds the target bytes over the duration, the file fails with "Impossible target (auto mode): ..." (section 22.1). Otherwise ComputeAutoModeSettings picks height/fps/audio; per-file W/H/FPS overrides then win over Auto's picks (dimensions are all-or-nothing: any user axis pins the pair, the missing one derives by aspect; FPS merges independently); merged odd dimensions are even-snapped; and no-op filters are dropped: a non-pinned height at or above the source (with an already-even source) clears both axes, and a non-pinned fps at or above the source clears the fps filter, so re-scaling or CFR-stamping to the source value never wastes a pass (an odd source keeps the scale filter as its even-pixel fixup). WAV forces the audio budget to the PCM rate.
- Manual mode: the effective user values are used directly (blank = no filter at all), odd values even-snapped, and the same audio-alone-over-target rejection applies with the selected codec's rate ("Impossible target (manual mode): ...").
- The initial video bitrate comes from CalculateVideoBitrate with the container's overhead factor; the `-vf` string from BuildVideoFilterString; the output path is finalOutputPaths[index].
- Sample decision: only clips of at least 10 seconds qualify; Auto mode enables sampling automatically for clips of 60 seconds or longer, manual mode follows the Sample switch.
- Status "Encoding video..." and EncodeWithBitrateRetryAsync runs (section 8), wrapped so an OperationCanceledException records Cancelled ("Skipped by user (Skip File).") unless the batch token fired, and any other exception records Failed with the exception message plus the stderr tail. After the call, a fired batch token breaks; a fired file token records Cancelled.
- A false return (or a missing output file) records Failed with, in priority order: the targeted reason set by the runner (_lastEncodeFailureReason), else the custom-options hint when EncoderTuningConfig.IsCustomized says this encoder's profile differs from its machine default, else the generic retry-budget message (all in section 22.1).
- Success records Succeeded with the output path and sizes, and FinalizeCurrentFileTiming freezes the file's clock.

9.3 The collision prompt (RunCollisionPromptAsync)

- Desired paths come from OutputPathPlanner.ComputeDesiredOutputPaths; with no collisions they are returned as-is. Otherwise one "Replace existing files?" dialog is shown whose body names the collision kinds, lists up to 8 conflicting filenames each annotated ("already exists", "produced by N batch files", or both), and spells out both consequences: Yes overwrites (later batch duplicates overwrite earlier ones), No renumbers (for example `hello_compressed1.mp4`, `hello_compressed2.mp4`) so nothing is overwritten. The answer feeds ResolveFinalOutputPaths (section 6.9).

9.4 Batch summary and teardown

- After the loop: the elapsed ticker stops; counts come from BatchResult; the progress bar snaps to 1.0 on a clean all-success run and 0.0 when nothing succeeded; the elapsed label freezes ("Cancelled after N of M

file(s). Time elapsed: hh:mm:ss" on a stop); the status line shows "Compression complete." (kept visible so the layout does not shift).

- When failures exist, TryWriteErrorLog writes the log (section 22.2). One merged dialog then shows the summary (BuildSummaryMessage: processed/succeeded/failed/skipped-by-you/skipped-already-small/not-processed counts, "Space saved" as signed MB/GB, up to 5 inline failures with an error-log pointer, and the folder question) with an Open Folder button when any output exists (TryOpenFolderInExplorer launches `explorer.exe` on the folder, best-effort).
- The finally block always runs the teardown: freeze times (SetLastRunElapsed), reset pause state, dispose both CTS, move _currentBatchResult to _lastBatchResult (so tints persist after the run), clear _isEncoding/_startRequested/_isPaused, kill any straggler FFmpeg process, reset the run-state mirror (CurrentQueueIndex = -1), re-enable idle controls (ResetUiAfterRun), disable Pause/Skip/Stop, and settle the status line: "Cancelled." after a stop, and transient statuses (Encoding/Analyzing/Retrying/Starting/Running sample/Encoding failed) are cleared so stale text never survives a run.

10 A Worked Example

Deterministic math from sections 6.4 and 8, end to end. Scenario: a 60.0 s clip, 1920x1080 at 30 fps, source audio 128 kbps, target 10 MB into MP4 with `libx264`, Auto mode on, Leniency 80%.

- Target to bitrate (section 6.4):

```
targetBits = 10 * 8,388,608               =  83,886,080 bits
totalKbps  = 83,886,080 / 60 / 1000 * 0.985 =  1377 kbps
```

- Audio budget: ceiling = max(128, 64) = 128; total minus the 150 kbps video reserve is 1227, well above 128, so audio gets its full 128 kbps and the initial video bitrate is 1377 - 128 = 1249 kbps.
- Auto search: candidate heights are 1080, 720, 480; candidate rates are 30, 25, 24. AVC's floor is 0.045 bits per pixel:

```
1080p30: 1,249,000 / (1920*1080*30) = 0.0201   below the floor
1080p24: 1,249,000 / (1920*1080*24) = 0.0251   below the floor
720p30:  1,249,000 / (1280*720*30)  = 0.0452   clears the floor
```

720p is derived as 1280 wide from the source aspect ratio. The first pass to clear the floor wins, so the pick is 1280x720 at 30 fps: resolution steps down one tier before any frame rate is sacrificed.

- Sample: the clip is 60 s, so Auto enables sampling on its own. The window is 25% of 60 s clamped to 12 s, taken from the 30 s mark; its measured size rescales the 1377 kbps starting point toward the band center before the first real attempt.
- The band (section 8.1): 80% leniency on 10 MB gives [8 MB, 10 MB], center 9 MB. Suppose attempt 1 lands at 10.6 MB (over): with only one side known, single-point scaling gives 1377 * (9 / 10.6) = 1169 kbps total, i.e. 1041 kbps video, within the 0.25x-4x swing clamp. If attempt 2 then lands at, say, 8.7 MB, it is inside the

band and the search stops with that attempt promoted.

11 Main Window UI Reference (MainPage.xaml)

11.1 Layout and named controls

The window is a fixed-height layout (no scroll view): a title bar, a two-column card area, and a bottom run strip. Named controls by card:

- Title bar: TitleLink (the app title, a tap target opening the website) and ResetSettingsButton ("Reset to Defaults", section 20.5).
- Files card: SelectButton ("Select Input Video(s)"), InputPathLabel, OutputFolderButton, OutputFolderLabel. Both path labels use LineBreakMode=MiddleTruncation with the full path in a tooltip.
- Video Encoder card: EncoderPicker, EncoderTuningButton, SampleEncodeSwitch (tooltip: applies to clips 10 seconds or longer), TwoPassSwitch.
- Format card: ContainerPicker, AudioCodecPicker, PreserveColorSwitch (caption notes it applies to all encoder profiles).
- Target Size card: TargetSizeEntry (header "Target Size (MB)"), AutoModeSwitch, TargetGraceSlider + TargetGraceLabel (the Leniency section below a divider).
- Resolution & Frame Rate card: HeightEntry, WidthEntry, FpsEntry.
- Run card: StartButton, ViewQueueButton.
- Bottom strip: ProgressBar, PauseResumeButton, SkipFileButton, StopBatchButton, TimeElapsedLabel, StatusLabel.

11.2 Shared control behavior

- Every Button gets the ButtonInteractions attached behavior through a style setter: Pressed animates Scale to 0.96 over 70 ms, Released springs back to 1.0 over 90 ms (CubicOut). Only Scale is animated; colors and the disabled gray-out stay with the VisualStateManager styles. The handlers are detached before attach so re-applying a style cannot double-subscribe.
- The title link fades its text color on pointer enter/exit between the themed PrimaryTextColor resting color and the live Windows accent color (read via UISettings at hover time; falls back to the AccentColor resource, then RoyalBlue). MAUI has no ColorTo, so FadeTitleColorTo commits a 200 ms Animation that interpolates each channel per frame (the VisualStateManager could only snap). Tapping opens `https://portablevideocompressor.ca/` with Browser.Default.OpenAsync (best-effort).

11.3 Enablement rules (MainPage.UiState.cs)

- SetStatus(text) writes StatusLabel on the UI thread (empty hides it) and republishes to CurrentRunStatus + RunStateChanged so the Queue page mirrors it.
- UpdateAutoModeUi(): while encoding, everything configuring the run is disabled. Idle: FpsEntry/HeightEntry/WidthEntry and SampleEncodeSwitch are also disabled while Auto mode is on;

EncoderPicker, TargetSizeEntry, AutoModeSwitch, PreserveColorSwitch, AudioCodecPicker, ContainerPicker, TargetGraceSlider, and EncoderTuningButton follow only the encoding flag.

- UpdateTwoPassEnabled() reflects encoder support and Auto mode in TwoPassSwitch: an encoder without two-pass forces the switch off and disabled; Auto mode forces it on and disabled; otherwise it is enabled and shows the durable preference _manualTwoPassChoice. Every programmatic write is wrapped in _suppressTwoPassWrite so the Toggled handler never mistakes it for a user choice (section 20.3).
- ResetUiAfterRun() re-enables Select/Start/Output for a non-empty queue and ViewQueueButton for 2+ files (a single file has nothing to manage or reorder), leaving ProgressBar.Progress as-is so the final value stays visible until the next Start.

11.4 Encoder probe wiring (MainPage.Encoders.cs)

- EnsureEncodersInitializedAsync() caches the one probe task (_encoderInitTask) so concurrent and later callers share it; both call sites run on the UI thread, so no lock is needed.
- InitializeEncodersAsync(): DiscoverEncodersAsync fills _encoders; SyncDefaultsToDetectedCapability re-tiers non-customized profiles (section 17.3); the picker is bound; PickPreferredEncoderIndex selects the default; UpdateTwoPassEnabled runs and re-runs on every SelectedIndexChanged; then ApplyPersistedEncoderSelection restores the saved encoder and 2-pass preference and opens the persistence gate (section 20.3).
- RedetectEncodersAsync() is the Reset-button variant: it re-probes from scratch, rebinds (ItemsSource nulled first to force a refresh), re-picks the preferred encoder, and marks the cached probe task complete.
- OnEncoderTuningClicked pushes EncoderTuningPage(encoder.FfmpegName, encoder.DisplayName) for the active encoder, refused while encoding so edits cannot race a run. OnViewQueueClicked pushes QueuePage(_queue, _encoders, this) whenever the queue is non-empty.

12 File Selection and Window Drag & Drop

12.1 Picker path (MainPage.FileSelection.cs)

- OnSelectInputClicked uses FilePicker.PickMultipleAsync with FilePickerFileType.Videos; _inputPickerOpen blocks a rapid double-click from opening two dialogs; a start in progress refuses the handler entirely. Results replace the queue (PopulateQueueFromPaths with replaceExisting: true).
- PopulateQueueFromPaths is the shared population routine: replace mode clears the queue and drops _lastBatchResult (so stale tints vanish); paths already queued are skipped case-insensitively; the input label shows the single path or "N files selected (first: <path>)"; the implicit default output folder follows the new selection unless the user made an explicit choice (_outputFolderIsExplicit); Start enables, ViewQueue enables at 2+ files, stale elapsed text clears, and the status returns to "Ready.".
- OnSelectOutputFolderClicked resolves the WinUI window handle (WindowNative.GetWindowHandle) and opens Win32FolderPicker.PickFolder starting in the current output folder (else the queue head's folder). A selection sets _outputFolder, marks it explicit, and persists. The Reset flow (section 20.5) is what returns the folder to the implicit follow-the-input default; with an empty queue that state shows the disabled button text

"Use input folder".

12.2 Window drop (MainPage.DragDrop.cs)

- HookDragAndDrop (Loaded) enables AllowDrop on the native root FrameworkElement and subscribes DragOver/Drop once (_dragDropHooked); on non-Windows it is a no-op so the shared call site compiles.
- DragOver accepts Copy only when the payload contains StorageItems and no start is in progress (the run's output paths are already frozen); the drag caption reads "Replace queue". Drop extracts StorageFile paths under a deferral and hands them to AddDroppedFiles(paths) (append: false).
- AddDroppedFiles filters to AcceptedVideoExtensions (a case-insensitive set: `.mp4` .mov .mkv .avi .wmv .flv .webm .m4v .mpg .mpeg .m2ts .ts .3gp .ogv .mts .vob .divx), silently dropping folders and non-video files, then populates the queue (replace from the main page; append from the Queue page, section 13.6). The native picker path does not need this list; it uses the OS video filter.

13 Batch Queue Page (QueuePage)

13.1 Construction and modes

- The constructor captures the live queue list (the same List<QueueItem> MainPage holds, so edits are visible on return), the machine's encoder list, the static audio/container catalogs, and the MainPage reference used for run state; _liveMode starts from MainPage.IsEncoding. Row building is deferred to OnAppearing (via Dispatcher.Dispatch) so the page-push animates smoothly.
- OnAppearing subscribes MainPage.RunStateChanged and starts a one-second mirrored ticker (elapsed label + live-row refresh); OnDisappearing unsubscribes and stops it. OnRunStateChanged rebuilds all rows when the live/edit mode flips, then refreshes the footer, per-row progress, and the elapsed label.
- Named controls: SummaryLabel, QueueTimeElapsedLabel, QueueDragHintLabel, QueueScrollView, QueueRowsLayout, mirrored QueuePauseButton / QueueResumeButton / QueueSkipFileButton / QueueStopBatchButton / QueueStartButton, ClearAllOverridesButton, DoneButton.

13.2 Edit-mode rows (BuildRow)

Each row is a rounded Border holding a header Grid (drag handle, name, caret, Remove) and detail lines:

- The drag handle is a two-dot glyph Label with a PanGestureRecognizer (the reorder gesture, section 13.5) and a hover tint to the accent color.
- The name label reads "<position>. <filename>" plus an override marker suffix when Overrides.HasAny; it middle-truncates. Below it: the full path (middle-truncated), the size line, and the process-time line.
- The size line (BuildSizeText) always shows the input size ("Input: X"; a dash when unreadable), appends the output once finished ("Input: X -> Output: Y"), and ends with the effective target ("Target: N MB") resolved snapshot-first (RunSnapshotForPath), then per-file override, then the live batch default. Bytes format in binary units (1 KB = 1024 B) via FormatBytes.

- The process-time line (BuildProcessTimeText) ticks "Elapsed: ..." for the running file (CurrentElapsed - CurrentFileStartElapsed) and freezes to "Took: ..." once finished (ProcessTimeForPath); durations format compactly (4s / 1m 23s / 1h 02m).
- The Remove button (an accent "X") confirmation-gates, then removes the item from all three index-aligned collections (_queue, the layout children, _rowVisuals), collapses expanders, renumbers, and refreshes (RemoveItem re-checks the run gate after the dialog).
- Tapping the header toggles the override expander; the editor content is built lazily on first expand (BuildExpander) so large queues stay cheap.
- Row hover swaps the border to the hover stroke, but never on a row that already carries an outcome tint.

13.3 The override expander (BuildExpander)

- Fields, in order: "Target size (MB)" Entry; a three-column Grid of "Width (px)", "Height (px)", "FPS" Entries; a "Video encoder" Picker; then the paired "Audio codec" and "Container" Pickers; then the "Reset this file to batch defaults" button (nulls Overrides and rebuilds). Every Entry uses the numeric keyboard and the placeholder "(batch default)"; every Picker's first row is the "(batch default)" sentinel.
- Setters parse invariant-then-current culture, positive-only; each routes through AttachOrUpdate, which lazily creates the PerFileOverrides bag, applies the mutation, prunes the bag back to null when nothing remains set, updates the name marker and size line in place (so the open expander survives the edit), and refreshes the header summary. Every mutation is frozen once a start is accepted (IsStartingOrEncoding).
- The audio/container pair stays compatible via one RefreshFormatPickers pass: audio is primary (an audio override incompatible with the effective container switches the container to the first compatible one, considering the batch default container when no container override is set); each picker's option list is then filtered by the other's selection; and both override values are written back. A suppression flag (_suppressOverrideCompatFilter) stops the repopulation's own SelectedIndexChanged from re-entering.

13.4 Row tints and the live view

- Outcome tints (ApplyOutcomeTint, colors as ARGB hex constants): Succeeded inside the band = green (#28A745 stroke over a #2628A745 fill); Succeeded outside the leniency band = amber (#FFC107 / #26FFC107); Failed = red (#DC3545 / #26DC3545) with a red strikethrough on the name; Skipped/Cancelled = gray (#808080 / #22808080) with a gray strikethrough; no outcome = the neutral #919191 stroke. IsOutsideLeniency computes fractionOfTarget = outputBytes / targetBytes and reads amber when it is below grace or above 1.0, preferring the run-time snapshot so moving the slider or target after a run never recolors finished rows.
- Live mode (BuildLiveRow) renders read-only rows carrying a hidden per-row status Label and a 6 px ProgressBar; RefreshLiveProgress highlights the running row (accent #0078D4 stroke over #160078D4), shows its bar/status from the mirrored run state, ticks its Elapsed line, and applies outcome tints plus refreshed size/time lines to every other row. The running row wins over its own not-yet-recorded outcome.
- The header summary states the count, the drag hint, and how many files carry overrides; in live mode it notes the batch is running and editing is disabled.

13.5 Drag-to-reorder (the pan algorithm)

- BeginRowDrag (refused once a start is accepted) collapses the dragged row's expander, snapshots every row's resting Y and height, seeds the auto-scroll state (scroll offset at grab, the dragged row's center), and lifts the row (ZIndex 10, opacity 0.92, accent dragging stroke/fill).
- ApplyDragLayout is a pure function of the pan offset plus any scroll delta: the dragged row's TranslationY tracks the cursor; its projected drop slot is the count of other rows whose resting centers sit above the dragged center; rows between the source and target slide by exactly one dragged-row height (TranslateToAsync, 90 ms CubicOut) to open the gap.
- Auto-scroll: a 40 ms ticker nudges QueueScrollView by AutoScrollStep = 24 px whenever the dragged center is within AutoScrollEdgeZone = 48 px of a viewport edge, re-applying the drag layout each nudge so the row keeps tracking the cursor.
- EndRowDrag cancels animations, clears translations and drag state, then either commits MoveItem(source, target) or just restores styling. MoveItem moves the item across the three index-aligned collections incrementally (no rebuild), collapses expanders, renumbers, restores styling, and refreshes the summary; it re-checks the run gate.

13.6 Footer, mirrored controls, and page drop

- UpdateFooterForMode: ClearAllOverridesButton and the drag hint only while idle; the Pause/Resume pair swaps on IsPaused; Skip/Stop enabled only live; the mirrored Start enabled only when idle with a non-empty queue (keyed off IsStartingOrEncoding so it does not blink during the preflight). DoneButton pops back ("Back").
- The mirrored playback buttons forward to MainPage.RequestPause/Resume/SkipFile/StopBatch (identical behavior; RunStateChanged re-syncs the buttons); the mirrored Start disables itself, forces a dispatcher round-trip so the gray-out paints before the synchronous preflight prefix, then calls RequestStartCompression. Queue Skip also disables itself immediately so a double-tap cannot fire twice.
- OnClearAllOverridesClicked confirmation-gates ("This will reset N file(s) back to the batch defaults."), nulls every item's Overrides, and rebuilds; it reports "Nothing to clear" when no file has overrides.
- The page's own drag-and-drop (QueuePage.DragDrop.cs) mirrors the window's but APPENDS: the caption reads "Add to queue" and the drop calls MainPage.AddDroppedFiles(paths, append: true), rebuilding rows only when the shared queue actually grew. The same start gate refuses drops mid-run.

14 Encoder Tuning Page (EncoderTuningPage)

14.1 Construction and state

- Constructed with (codecName, encoderDisplayName); _settings is the shared EncoderSettings from EncoderTuningConfig.Current (edits therefore reach the next run with no plumbing). Named controls: EncoderTitleLabel, PresetPicker, TunePicker, RateControlSection/RateControlPicker, MultipassSection/MultipassPicker, ExtraOptionsEditor, CustomNoticeLabel, ResetButton, DoneButton.

_suppressEvents guards every programmatic picker/editor write. IsAmfCodec special-cases the AMF encoders, whose preset is really `-quality` and whose tune is really `-usage`.

14.2 Per-codec option lists (InitializeOptionLists)

Every picker starts cleared and disabled with the title "Not available for this encoder"; the codec switch then enables what applies. Add* helpers keep each option list, its id-to-index map, and the picker rows in lockstep.

CODEC	PRESET ROWS	TUNE ROWS	RATE CONTROL	MULTIPASS
`libx264`	ultrafast...placebo (10 steps)	None, film, animation, grain, stillimage, fastdecode, zerolatency, psnr, ssim	hidden	hidden
`libx265`	ultrafast...placebo	None, animation, grain, fastdecode, zerolatency, psnr, ssim	hidden	hidden
`h264_nvenc` / `hevc_nvenc`	p1 "P1: Fastest" ... p7 "P7: Slowest"	None, hq, ll, ull	`vbr_hq`, vbr, cbr	fullres, qres, Disabled
`h264_amf` / `hevc_amf`	quality, balanced, speed	transcoding, `high_quality`, lowlatency, ultralowlatency, webcam	vbr, hqvbr, hqcbr	hidden

14.3 The sync engine (Sync.cs + PickerEvents.cs)

- LoadSettingsIntoUi copies ExtraOptions into the editor (newlines collapsed) and runs the analyzer. Typing commits on every keystroke: OnExtraOptionsTextChanged writes the collapsed text into _settings.ExtraOptions and re-analyzes, so Back and Done are equivalent and the editor always mirrors the active profile.
- AnalyzeOptionsAndSyncPickers parses the text with FfmpegOptionsParser and selects the matching row in each picker (AMF reads `-quality`/`-usage` in place of `-preset`/`-tune`). A present value with no matching row puts that picker into a Custom state (SetPickerCustomState): selection cleared and, except for rate control, the picker disabled with the title "<label>: Custom (edit text below)"; rate control stays interactive so a valid value can be picked without clearing the text. CustomNoticeLabel then lists the custom pickers as "Custom from text for: ...". Parsed values also write back into _settings.Preset/Tune so the stored picker fields stay truthful.
- Picker changes go the other way: each handler rewrites its one flag inside the editor text in place via UpdateCoreOptionInEditor, preserving every unrecognized token the user typed; the "None"/"Disabled" rows remove the flag (allowEmpty), and AMF rewrites the canonical key to the real flag name. The rewrite re-runs the analyzer so the other pickers reflect the change.

14.4 Reset and Done (SaveReset.cs)

- Reset calls EncoderTuningConfig.Current.ResetToDefaults(codec) (which restores this machine's detected tier, section 17.3) and reloads the UI from the shared settings.
- Done does a final SaveUiToSettings (collapse text; re-derive Preset/Tune with the AMF fallbacks) and pops; the system Back arrow is equivalent because everything already committed on change.

15 Quality Warning Page (QualityWarningPage)

The modal pre-batch heads-up, pushed with PushModalAsync; the caller awaits ResultTask.

15.1 Structure and contract

- The Result record carries Cancelled plus SkipPaths (case-insensitive input paths the user opted out of). The page resolves a TaskCompletionSource created with RunContinuationsAsynchronously (so the caller resuming on the UI thread cannot re-enter), and OnDisappearing also resolves it as Cancelled: closing the page any other way (back gesture, OS pop) can never hang the caller, and TrySetResult's idempotence means a deliberate Continue/Cancel is never overridden.
- Each flagged file becomes one row (RowState record: the QualityFlag, the include CheckBox defaulting to checked, the editable target Entry pre-filled with the effective target, and the original target for edit detection): checkbox, bold filename over a "Minimum passable: ~N.N MB" readout, and a "Target: [Entry] MB" group. SelectionSummaryLabel tracks "N of M flagged file(s) will be compressed" as boxes toggle.

15.2 Continue and Cancel

- Continue validates every target first (all-or-nothing: TryParseTargetMb mirrors Start's rules, invariant then current culture, > 0; invalid rows are listed, capped at 5 inline, and the page stays open). Changed targets (beyond a 0.005 MB epsilon so cosmetic re-typings like 10 vs 10.0 are not edits) are written back as per-file overrides onto the very QueueItem instances the queue and loop use, so the new targets are immediately visible everywhere. Unticked rows collect into SkipPaths; the result resolves with Cancelled=false and the page pops (SafePopAsync guards a navigation race with app teardown).
- Cancel resolves Cancelled=true and pops; the Start pipeline then abandons the run (section 9.1).

16 Format Compatibility (Services\MediaFormatCatalog.cs)

The catalog is the single source of truth; the main page's picker filtering, per-file override validation, and the "same as source" guard all consult it.

16.1 The options

AUDIO OPTION	ID	FFMPEG CODEC	NOTES
Same as source (no re-encode)	source	`-c:a` copy	default; subject to 16.3
AAC	aac	aac	default lossy bitrate 128 kbps
MP3	mp3	libmp3lame	default lossy bitrate 192 kbps
WAV (PCM 16-bit)	wav	`pcm_s16le`	no `-b:a`; rate follows the source

CONTAINER	EXTENSION	OVERHEAD FACTOR	+FASTSTART
MP4	`.mp4`	0.985	yes

CONTAINER	EXTENSION	OVERHEAD FACTOR	+FASTSTART
MOV	.mov	0.985	yes
MKV	.mkv	0.975	no

GetDefaultLossyBitrateKbps returns 128 for AAC, 192 for MP3, and 128 for any other id (PCM/WAV never uses it). MKV's lower overhead factor reflects its higher EBML/cluster/cues framing cost.

16.2 Picker compatibility

- IsCompatible(codecId, containerId) encodes exactly one hard rule: WAV (PCM) into MP4 is disallowed; every other pairing is allowed (MOV and MKV both accept PCM).
- The main page's two pickers cross-filter from this rule (MainPage.FormatFilters.cs): whichever picker the user did not touch is rebuilt to only the options compatible with the touched one, keeping the prior selection when still valid, else the first option. _suppressCompatibilityFilter stops the repopulation's own SelectedIndexChanged from re-entering and clobbering the selection. The per-file override pickers apply the same rule with audio as the primary axis (section 13.3).

16.3 Stream-copy deny lists ("Same as source")

- CanCopyAudioInto vetoes a copy when the probed source audio codec name starts with a denied prefix for the chosen container. Unlisted codecs fall through to allowed, so a stale list can only miss a warning, never block a working copy; a null/blank codec (no audio) is always allowed. MKV maps to an empty list and is never denied.

CONTAINER	DENIED SOURCE-CODEC PREFIXES
MP4	pcm, adpcm, flac, vorbis, opus, truehd, mlp, wma
MOV	flac, vorbis, opus, truehd, mlp, wma
MKV	(none)

- Opus is on the MP4 list because the mp4 muxer gates Opus-in-MP4 behind `-strict` experimental. PCM is QuickTime-native, so MOV allows it.
- DescribeCopyAlternatives builds the remedy sentence for a veto: it offers re-encoding and names every container that would accept a copy of this codec (MKV always qualifies).

17 Encoder Tuning Profiles and Machine Tiers (Models\EncoderTuningConfig.cs)

17.1 The registry

- EncoderTuningConfig.Current is the app-wide singleton; it owns one EncoderSettings per codec (Libx264, Libx265, H264Nvenc, HevcNvenc, H264Amf, HevcAmf), seeded through ResetToDefaults for each in the private constructor. GetSettings(codec) maps an FFmpeg name to its container, null for anything else. The Main page, Queue page, Tuning page, and the encode loop all read the same instances, so tuning edits apply to the next run with no plumbing.

17.2 Default profiles and tiers

Each profile is Preset + Tune + an authoritative options string; the color force block (section 18) is part of the string. FullDefaultExtraOptionsFor:

CODEC	FULL DEFAULT OPTIONS
`libx264`	`-preset` medium + force block
`libx265`	`-preset` medium + force block
`h264_nvenc`	`-preset` p5 `-tune` hq `-rc` `vbr_hq` `-multipass` fullres `-rc`-lookahead 32 `-spatial_aq` 1 `-temporal_aq` 1 `-aq`-strength 8 `-bf` 3 + force block
`hevc_nvenc`	as `h264_nvenc` plus `-b_ref_mode` middle (HEVC NVENC benefits from B-frames as references; AVC NVENC does not) + force block
`h264_amf`	`-usage` transcoding `-quality` quality `-rc` hqvbr `-preanalysis` 1 `-vbaq` 1 + force block
`hevc_amf`	`-usage` transcoding `-quality` quality `-rc` hqvbr `-preanalysis` 1 `-vbaq` 1 + force block

- CapabilityGatedFlags: `-b_ref_mode`, `-multipass`, `-temporal_aq`, `-preanalysis`, `-vbaq` (NVENC b-ref/multipass/temporal AQ want Turing or newer; AMF pre-analysis/VBAQ want newer drivers). ConservativeDefaultExtraOptionsFor is the Full string with those flags and their values stripped (StripGatedFlags uses the same boundary-lookahead pattern as ColorOptions.Strip); it is identical to Full for the software encoders. MinimalDefaultExtraOptionsFor is the color force block alone, distinct only for the hardware codecs (HardwareCodecNames); software returns its Full profile.
- ResetToDefaults picker values: `libx264`/`libx265` preset "medium" with no tune; NVENC preset "p5", tune "hq"; AMF empty (its picker values come from `-quality` and `-usage`).

17.3 Machine-aware defaults, customization, and sync

- DefaultExtraOptionsFor(codec) consults EncoderDetector.SafeDefaultOverrides first (the tier the launch probe found, section 6.3) and falls back to the Full profile. Reset and IsCustomized route through it, so Reset restores a working profile for this machine, and a probe-chosen reduced tier is never blamed as a user customization.
- IsCustomized(codec) compares whitespace-normalized options strings (Norm) against that machine-aware default; it feeds the failure-message choice in section 9.2.
- SyncDefaultsToDetectedCapability() runs after every probe: for each codec whose current options equal ANY canonical tier (full, conservative, or minimal - i.e. the user has not customized), it swaps in the machine's detected tier in both directions (upgrade after a driver fix, downgrade after a GPU swap) and re-derives Preset/Tune from the new string with the AMF quality/usage fallbacks, so the pickers stay truthful. User-customized strings are never touched.
- BuildVideoCodecArgs is documented in section 7.1.

18 Color Handling

- The force block and its helpers are section 6.7; this section is the policy. The options strings are the single source of truth: the main page's Preserve-color switch derives its state from ALL six profiles (DeriveGlobalPreserveColor reads OFF when any profile carries a pixel-format flag, the conservative reading for mixed hand-edited states) and writes to all six (ApplyPreserveColorToAllEncoders calls ColorOptions.Apply per profile), exactly as its caption states.
- OFF (the default) guarantees each profile carries exactly one force block: 8-bit 4:2:0 BT.709 limited range, the maximum-compatibility output. ON strips every owned color flag so the encoder passes the source characteristics through.
- The toggle is re-derived (suppressed, no cascade) whenever the main page re-appears after the Tuning page, so a hand-edit that adds or removes color flags is reflected instead of overwritten (section 4.5).
- Preserving source color raises the quality pre-flight's minimum estimate (section 19.2) and is the usual trigger of the hardware 4:2:2 10-bit failure (sections 8.6, 22.1).

19 Quality Pre-flight (MainPage.PreflightChecks.cs)

Before encoding, every file gets an independent minimum-passable-size estimate; files whose target is below it are listed in the QualityWarningPage (section 15). Estimates run in parallel, capped by a SemaphoreSlim at maxParallel = 4; results land in an index-aligned array so flag order matches the queue. Preserve-color for the check is read from the batch encoder's profile (!ColorOptions.HasForce). A file already at or under its effective target, or one whose duration cannot be probed, is never flagged (it will be skipped or fail cleanly in the loop instead).

19.1 The judged configuration (mode-aware)

- Effective values first: every axis (target, W/H/FPS, encoder, audio, container) is per-file override else batch, exactly like the loop.
- Auto mode: an axis with a per-file override is judged at that override (the loop honors overrides); axes Auto controls are judged at the smallest fallback Auto could choose - 480p and 24 fps, each clamped to the source. This answers "could this file pass at any configuration Auto might pick?".
- Manual mode: judged at the user's effective configuration (override -> batch fields -> source passthrough), because that is exactly what will be encoded. Pinned high resolutions or frame rates therefore flag more files - by design.
- Dimensions are resolved all-or-nothing and even-snapped exactly like the loop: both set = both used; one set = the other follows the source aspect ratio; none = the mode's fallback; every result is snapped down to even with a floor of 2.

19.2 The estimate

- Minimum video rate from the bits-per-pixel floors (0.028 HEVC / 0.045 AVC; the codec is HEVC when its name is `libx265` or contains "hevc", mirroring section 6.4):

```
minVideoBps = width * height * fps * bppFloor
```

- Preserving source color can carry more data per pixel than 8-bit `yuv420p`, so the minimum is raised multiplicatively from the probed SourceColorInfo: x1.25 for 10-bit or higher, and x1.50 for 4:4:4 else x1.25 for 4:2:2.
- The audio allowance mirrors the loop: "Same as source" = the probed bitrate (96 kbps fallback); WAV = PCM math, sampleRate x channels x 16 bits (44100 / 2 fallbacks, 1411 kbps floor guard); any lossy codec = the 64 kbps floor.
- Minimum size, expressed as the on-disk figure the Target Size entry represents:

```
minTotalMb = (minVideoBps + minAudioBps) * duration / 8 / 1,048,576
           / containerOverheadFactor
```

- A file is flagged (QualityFlag) when its effective target is below the estimate. The flag holds the same QueueItem instance the queue uses, so the dialog's skip and target edits propagate with no synchronization step (section 15.2).

20 Settings Persistence

20.1 Location and writes

Owned by AppSettingsStore (section 6.1). The portable download ships `portable.dat` at the folder root, one level above App\, so `settings.json` lives at that root; a marker beside the exe (a flat layout) keeps settings beside the exe instead; with no marker in either place (the setup-program install), `settings.json` lives under %AppData%\PortableVideoCompressor. Writes are atomic temp-then-move and always best-effort.

20.2 Schema (Models\PersistedSettings.cs)

- SchemaVersion: CurrentSchemaVersion = 2; the property defaults to 1 so a file predating the field reads as the old schema. Files stamped 1 are migrated at load (20.3): their tuning strings were saved without color flags (color used to be injected at encode time), so migration re-applies the force block from the stored PreserveColor value; files stamped 2 load their options strings verbatim, because the strings are authoritative.
- Every field is nullable or defaulted, so an old or partial file degrades to defaults rather than throwing. Per-file overrides are deliberately not stored: they belong to a batch, not the defaults.

FIELD	MEANING
SchemaVersion	on-disk schema (defaults 1; new saves stamp 2)
TargetSizeText	Target Size entry as raw text (blank round-trips)
AutoMode	Auto mode switch
HeightText / WidthText / FpsText	manual entries as raw text

FIELD	MEANING
AudioCodecId / ContainerId	picker selections by stable id, matched on load
EncoderFfmpegName	chosen encoder; honored only if still available
SampleEncode / TwoPass	switch states (TwoPass is the durable manual preference)
PreserveColor	the migration input for v1 files; a snapshot in v2+
LeniencyPercent	slider percent (50-95; default 80)
OutputFolder	explicit output folder, or null for per-input folders
EncoderTuning	per-codec EncoderTuningDto map (Preset / Tune / ExtraOptions)
WindowX/Y/Width/Height	window bounds; only the position is restored (section 4.4)

20.3 Load, apply, and the persistence gate (MainPage.Settings.cs)

- Two-phase load, because the encoder list needs the runtime probe. Phase 1 (LoadAndApplyPersistedSettings, in the constructor) applies everything probe-independent under _suppressPersist: tuning profiles (ApplyTuningFromSettings, with the v1 color migration per profile: a v1 string without a pixel-format flag gets ColorOptions.Apply from the saved PreserveColor), the four raw-text entries, container and audio by id, the Sample and Auto switches, the derived Preserve-color state, the clamped leniency value, and an explicit output folder; the saved encoder name and 2-pass value park in _pendingEncoderName / _pendingTwoPass. Phase 2 (ApplyPersistedEncoderSelection, at the end of the probe) selects the saved encoder when it is still available, seeds _manualTwoPassChoice, re-derives the switch, and only then sets _settingsLoaded = true, opening the gate: PersistSettings() is a no-op until then, so applying loaded values can never write a half-applied snapshot back.
- CaptureCurrentSettings builds the save snapshot from the live UI and the tuning registry, stamps CurrentSchemaVersion, persists the durable _manualTwoPassChoice (not the possibly-forced switch state), stores the output folder only when explicit, and carries the window bounds through untouched from the existing file (they are owned by the window lifecycle).
- HookSettingsPersistence wires save-on-change once, all routed through the gated PersistSettings(): entries save on Unfocused and Completed; the Auto/Sample switches, three pickers, and Preserve-color on change; the leniency slider on DragCompleted (drag ticks update the label live but defer the save; keyboard changes save immediately); and the TwoPass Toggled handler records the value into _manualTwoPassChoice only when _suppressTwoPassWrite is false, which is how programmatic forces (section 11.3) never overwrite the preference.

20.4 Window lifecycle saves

- HookWindowPersistence (on Loaded, once) adds Window.Deactivated -> PersistSettings (a safety net for edits never unfocused) and Window.Destroying -> PersistSettings plus teardown of any in-flight FFmpeg child: ResumeIfSuspendedForCancellation then KillCurrentFfmpegProcess, so a paused/suspended child is never orphaned by closing the app. The window's own Destroying handler saves the bounds (section 4.4).

20.5 Reset to Defaults (OnResetSettingsClicked)

Confirmation-gated and refused mid-run. Under _suppressPersist it: re-runs hardware detection (RedetectEncodersAsync, which also re-tiers profiles), blanks the four entries, turns Auto/Sample/Preserve-color off (rewriting the force block into all profiles), resets leniency to 80, restores both pickers to their full lists and canonical defaults (Same as source + MP4), resets every codec's tuning via ResetToDefaults, returns the output folder to the implicit follow-the-input default (ResetOutputFolderToDefault; with an empty queue the button shows the disabled "Use input folder" state), and sets _manualTwoPassChoice = true. It then refreshes enablement and persists once. Per-file overrides are untouched.

21 Run Control: Pause, Resume, Skip, Stop (MainPage.PlaybackControls.cs)

21.1 Pause and Resume

- The single PauseResumeButton branches on _isPaused. Pause records the intent (_isPaused, _pauseStartTime) under _processLock and suspends the live process via ProcessSuspender (CPU only; file handles stay open); the intent is recorded even when no child is alive at a pass or file boundary, and each pass runner re-applies suspension to its next process (section 7.3), so a boundary pause still takes effect. ffprobe probes are never paused. The button flips to "Resume" and the status reads "Paused.".
- Resume unsuspends any live process, accumulates the completed pause interval into _totalPausedTime, clears _pauseStartTime, flips the button back, and restores "Encoding video...".

21.2 Skip and Stop

- Skip File (no confirmation) disables its button, resumes a suspended child first (killing a suspended process can hang: ResumeIfSuspendedForCancellation is Resume minus the UI updates), then fires _fileCts and kills the process tree; the loop records the file Cancelled and continues (section 9.2).
- Stop Batch confirmation-gates with a remaining-file count (ComputeRemainingFileCount: queue entries without a recorded result, minus the in-flight file tracked by _currentTimingPath) and states that finished files are kept; on confirm it re-checks state (Skip may have ended the run during the dialog), disables Skip/Stop, resumes-if-suspended, fires both _batchCts and _fileCts, and kills the tree. KillCurrentFfmpegProcess always kills the entire process tree, since GPU encoders can spawn helpers.
- The internal mirrors IsPaused / RequestPause / RequestResume / RequestSkipFile / RequestStopBatch let the Queue page drive the identical handlers (section 13.6).

21.3 Elapsed accounting

- GetEffectiveElapsed() = now - _encodeStartTime, minus the in-progress pause (now - _pauseStartTime when paused) and minus _totalPausedTime; clamped at zero. A one-second Dispatcher timer paints "Time elapsed: hh:mm:ss" while _elapsedTimerRunning.
- Per-file times: BeginFileTiming (called after the skip gates) freezes the previous file's span into _fileProcessTimes (pause-excluded, keyed case-insensitively by path) and records CurrentFileStartElapsed;

FinalizeCurrentFileTiming freezes the current file (idempotent). ProcessTimeForPath serves the Queue page's "Took:" line; the live "Elapsed:" line is CurrentElapsed - CurrentFileStartElapsed.

- LastRunElapsed freezes the whole run's clock at teardown so both pages can keep showing it after the run.

22 Error Handling and the Message Catalog

22.1 The messages

All user-visible failures carry targeted, actionable text; the exact wording, by cause:

SITUATION	MESSAGE (ESSENCE)
No input selected	"Missing File - Please select one or more video files first."
Non-Windows	"Compression is currently only implemented on Windows."
FFmpeg missing	"`ffmpeg.exe` and/or `ffprobe.exe` were not found. Place `ffmpeg.exe` and `ffprobe.exe` in the "`FFMPEG`" folder (next to the app folder) and try again."
Bad target / field	"Invalid value" with the field's rule (e.g. "Target Size must be a valid number (for example: 9.9)")
Everything already small	"Nothing to do - All N selected file(s) are already at or below their target size."
Low disk space	worst-case size vs free space, "Continue anyway?"
Input vanished	"Input file not found at the recorded path."
Unreadable input	"Could not access input file: <exception>"
No duration	"Could not read video duration (ffprobe returned no valid duration)."
Copy veto	"Source audio is "<codec>", which the <container> container can't carry with "Same as source (no re-encode)"." + DescribeCopyAlternatives
Explicit codec mismatch	"The <container> container can't carry <codec> audio." + a remedy naming compatible containers
Impossible target (auto)	"Impossible target (auto mode): Even with minimum allowed audio bitrate (64 kbps), audio alone would exceed the requested target size." (WAV variant names the PCM rate)
Impossible target (manual)	"Impossible target (manual mode): ..." (same shapes)
Retry budget exhausted	"Encoding failed: ffmpeg could not produce a usable output within the bitrate-retry budget."
Customized profile suspected	"Encoding failed. This encoder profile has custom FFmpeg options, which may be the cause. Try "Reset to Defaults" ... " (only when IsCustomized, section 17.3)

SITUATION	MESSAGE (ESSENCE)
Output locked	"Could not write the output file - it may be open in another program: <path> (<exception>) The best encoded attempt was kept next to it as "<name>_tmpN.<ext>"."
HW 4:2:2 10-bit	"<encoder> can't encode this file's 4:2:2 10-bit format on this system. Use a software encoder (`libx264` or `libx265`). If "Preserve source color" is on, turning it off also avoids this."
Skips	"Skipped by user (quality warning)." / "Skipped by user (Skip File)." / skipped-already-under-target

- Status-line vocabulary during runs: "Starting compression...", "Analyzing input video...", "Pre-flight check: estimating quality at target size...", "Running sample encode for bitrate estimation...", "Encoding video...", "Retrying with adjusted bitrate...", "Paused.", "Skipping current file...", "Cancelling batch...", "Cancelled.", "Compression complete.", plus the per-file Skipped/Failed one-liners.

22.2 The error log (TryWriteErrorLog)

- Written only when failures exist, to the explicit output folder, else the first success's folder, else the current input's folder; named `PVC_batch_errors_<timestamp>.txt` with timestamp `yyyyMMdd_HHmmss` (invariant).
- Contents: a header with the app name and generation time, the failure count, then one numbered block per failure: the input path, the CompletedAt time, the original size in MB, the reason, and, when captured, the FFmpeg stderr tail indented line by line. Best-effort: any write failure returns null and the summary simply omits the pointer.

22.3 Dialog-vs-log policy

- The summary dialog stays readable: at most 5 failures inline with a "...and N more (see error log)" tail; the stderr tail never appears in a dialog, only in the log. The Queue page is the visual error surface (tints + strikethrough, section 13.4).

23 Limitations and Notes

- Windows-only by design (WinUI front-end, NT-level process suspension, the NUL output device in probes and pass 1, Win32 folder picker and drag-and-drop).
- Hardware encoders do not support two-pass; Auto mode compensates with sample-encode calibration and retry interpolation.
- The default NVENC profiles use `-rc vbr_hq`, which upstream FFmpeg has deprecated in favor of `-rc` vbr with the newer preset/tune system. The bundled build accepts it; if a future FFmpeg drops the flag, the launch-time tier probe falls back to a working profile automatically (see the README's FFmpeg-upgrade checklist).
- Format compatibility tables are conservative allow/deny lists; an out-of-date table can only miss a warning, never block a working combination (MKV is never denied).
- The single-flight FFmpeg model (one child process at a time) is deliberate: it keeps Pause/Skip/Stop unambiguous and disk/GPU contention predictable.

- Settings writes are best-effort everywhere: a read-only install location degrades to session-only settings, never to a failed encode.

24 Release

- This document describes Portable Video Compressor 2.0.
- Version identity and where each value lives: ApplicationDisplayVersion 2.0 and ApplicationVersion 6 (the integer build counter) in `PortableVideoCompressor.csproj`; assemblyIdentity version 2.0.0.0 in Platforms\Windows\app.manifest; the application license (`Licenses\PVC.txt`) carries the same 2.0 designation.
- The shipped document set (Documents\README.pdf, Documents\User's Guide.pdf, and this file) is generated from a single documentation source and versioned together with the application.

END OF TECHNICAL SPECIFICATION