USER MANUAL

ACE Editors' Notes.

Timecoded notes for DaVinci Resolve on macOS — every timecode you type is a clickable link that jumps the playhead to the exact frame.

Note every frame.

macOS · DaVinci Resolve
8 sections
ace-presenter.app

Contents

OVERVIEW

ACE Editors' Notes — User Manual

A complete reference for ACE Editors' Notes, the native macOS note-taking app for DaVinci Resolve editors. Its defining idea: timecodes in your notes …

ACE Editors' Notes — User Manual

A complete reference for **ACE Editors' Notes**, the native macOS note-taking app for DaVinci Resolve editors. Its defining idea: **timecodes in your notes are clickable, and clicking one drives Resolve's playhead** to that exact frame. Take notes next to your timeline, import markers straight from Resolve, and hand off a clean brief as PDF or TXT.

> **Status: public beta.** ACE Editors' Notes is shipping as a public beta (current build **1.6.1**). It is a macOS app for **Apple Silicon**. It works fully **standalone and offline**; the live link to DaVinci Resolve is an optional add-on that needs Resolve and a Python install. Where a feature is partial or gated, this manual says so plainly rather than glossing over it.

> **The app was historically called "CutNotes."** You may still see that name in a few places — the on-disk database folder, some file paths, older download names. It is the same product. This manual uses the current name, **ACE Editors' Notes**, throughout.

How to read this manual

Keyboard shortcuts. ACE Editors' Notes is macOS-only, so shortcuts use Mac modifier symbols: ⌘ Command, ⌥ Option, ⇧ Shift, ⌃ Control. Menu paths use ▸ (e.g. AI ▸ Polish Note).

Status notes. Because this is a beta, availability is flagged inline where it matters:

BADGE	MEANING
(ships today)	Works in the current beta build
(needs Resolve)	Requires DaVinci Resolve running with the Python bridge connected
(needs macOS 26+)	On-device AI feature; requires macOS 26 or later with Apple Intelligence
(coded, blocked)	Implemented in the app but blocked by an external bug — see the note where it appears

The single most important "coded, blocked" item: **creating Resolve markers from your notes.** The code is finished and ready, but a bug in DaVinci Resolve 20.x's scripting API prevents it from working today. The shipping direction is therefore **one-way for markers** — Resolve → Notes (import) — plus click-to-seek. See Chapter 4 for the full story.

What ACE Editors' Notes is (and isn't)

It is a fast, local, distraction-free notes app that understands timecode and talks to your Resolve timeline. Notes live in a hierarchy of **Projects ▸ Timelines ▸ Notes**, stored in a local SQLite database with no cloud dependency by design — post facilities that firewall the edit bay are fully supported.

It isn't an NLE, a media manager, or a collaboration server. It doesn't edit your footage, and (today) it doesn't sync to the cloud or support multiple simultaneous users. It sits beside Resolve and makes your notes navigable.

The main window at a glance

ACE Editors' Notes is a single window with a dark, edit-suite-friendly theme:

- **Project dropdown** (top) — pick or create the project you're working on.
- **Timeline dropdown** — pick or create a timeline within that project; each timeline has its own notes.
- **Search bar** — filter across notes; matches highlight in yellow as you type.
- **The notes editor** — the large central area where you write. Timecodes turn blue and bold automatically and are clickable.
- **Button bar / toolbar** — Insert Timecode, Import Markers, Polish Note, Hold to Record, Export to PDF / TXT, plus the category selector.
- **Resolve connection indicator** — a coloured dot: **green** = connected to Resolve, **red** = standalone/offline.

A short **Welcome tour** spotlights these on first launch and can be replayed from Help ▸ Welcome Tour.

A 60-second tour of the workflow

1. Create a **Project** (the job) and a **Timeline** (the cut) — see Chapter 2.
2. Start typing notes. Type a timecode like `01:02:33:18`, or press **⌘T** to stamp Resolve's current playhead position — see Chapter 3.
3. With Resolve connected, **click any blue timecode** and Resolve jumps to that frame.
4. **Import Markers** to pull existing Resolve markers in as notes — see Chapter 4.
5. Colour-code notes by **category** (VFX, Audio, Color…) and **search** to find anything — see Chapter 5.
6. **Export to PDF or TXT** and hand the brief to the lead editor or director — see Chapter 7.

Everything above works offline. Steps 2–4's live Resolve behaviour needs the Python bridge connected, covered in Getting Started.

SECTION 01

1 · Getting Started

This chapter takes you from download to a running app, explains exactly what you need for the optional DaVinci Resolve link, and gets you connected.

01 · 1 · Getting Started

This chapter takes you from download to a running app, explains exactly what you need for the optional DaVinci Resolve link, and gets you connected.

Requirements

ACE Editors' Notes has a small **core** that runs on its own, and an **optional Resolve bridge** that needs a couple of extra pieces. You can use the app happily with only the core.

WHAT	REQUIREMENT	NEEDED FOR
macOS	macOS 12 Monterey or later	The app itself
Mac hardware	Apple Silicon (M-series)	The shipping build is a native `arm64` binary — no Rosetta
DaVinci Resolve	Resolve 18 or later (18.x–20.x; tested against 20.3.2)	Click-to-seek, marker import, playhead timecode stamping
Python 3	A Resolve-compatible Python 3 install (tested with 3.14)	The live bridge to Resolve only
macOS 26+	macOS 26 or later with Apple Intelligence enabled	The on-device AI features (Note Polish, Voice Memo) only — see Chapter 6

Reconciling the macOS version

There are two different floors, and it's worth being clear about them:

- **The app runs on macOS 12 and up.** Notes, timecodes, click-to-seek, marker import, categories, search, and export all work from Monterey onward.
- **The AI features need macOS 26 or later.** Note Polish and Voice Memo are built on Apple's on-device models (the Foundation Models and Speech frameworks), which only exist on macOS 26+. On earlier macOS, those two buttons are present but **disabled**, with a tooltip explaining why. Nothing else is affected.

In short: **macOS 12 to use the app; macOS 26 to also use the AI.**

DaVinci Resolve version support

The Resolve link is **optional**. When Resolve isn't running, the app is a normal offline notes app.

- Supported: **DaVinci Resolve 18 through 20**. Development and testing reference is **20.3.2**.
- The link works with both the free DaVinci Resolve and DaVinci Resolve Studio.

- One important caveat lives entirely in Resolve, not in this app: **Resolve 20.x has a scripting bug that blocks creating markers from notes.** Importing markers from Resolve and click-to-seek are unaffected. Full detail in Chapter 4.

Installing the app

1. Download the current DMG (for example `ACE-EditorsNotes-1.6.1-arm64.dmg`) from the ACE website.
2. Open the `.dmg` and drag **ACEEditorsNotes.app** into your **Applications** folder.
3. Launch it from Applications. The shipping public beta is Apple-signed and notarized, so it should open normally; macOS may ask you once to confirm opening an app downloaded from the internet — click **Open**.

Updates install themselves: ACE Editors' Notes uses **Sparkle** auto-update, so new beta builds download and apply in the background without interrupting your session.

> **If macOS says the app is "damaged" or won't open.** Early beta builds were not signed, and a stale download can still trip Gatekeeper. If that happens, open **Terminal** and run:
>
> `xattr -cr /Applications/ACEEditorsNotes.app`
>
> Then try opening it again. You only need to do this once. (This clears the quarantine flag macOS adds to downloaded files; it is safe.)

First launch

On first launch you'll see the single main window and, a moment later, a **Welcome tour** — a spotlight overlay that walks you through the library dropdowns, Insert Timecode, Import Markers, the AI buttons, and export. You can **Skip** it and replay it any time from Help ▸ Welcome Tour.

If DaVinci Resolve isn't running, the connection dot (top of the window) is **red** — that's expected and fine. Create a project and start taking notes; the app is fully usable offline. See Projects, Timelines & Notes.

Your notes are stored locally in a SQLite database under ~/Library/Application Support/ (the folder retains the legacy CutNotes name). Nothing leaves your Mac unless you explicitly turn on the optional cloud AI upgrade in Chapter 6.

Setting up the DaVinci Resolve bridge

This section is only needed for the live features (click-to-seek, playhead stamping, marker import). Skip it if you're using the app standalone.

How the bridge works

ACE Editors' Notes doesn't touch Resolve directly. It launches a small **Python helper process** (`resolve_bridge.py`) and talks to it over JSON. That helper uses Blackmagic's official **DaVinci Resolve Scripting API** to move the playhead, read markers, and report project/timeline info.

```
ACE Editors' Notes  ──JSON──▶  resolve_bridge.py  ──▶  DaVinci Resolve Scripting API
   (click a timecode)              (seekTimecode)          (playhead moves)
```

Because it uses Resolve's own scripting API, three conditions have to be true for the bridge to connect.

Step 1 — Install a Resolve-compatible Python 3

Resolve's Scripting API needs a Python 3 interpreter it can load. The app has been tested with Python 3.14. The simplest route is Homebrew:

```
brew install python
```

Confirm it's there:

```
python3 --version
```

Step 2 — Enable external scripting in Resolve

1. Open **DaVinci Resolve**.
2. Go to **DaVinci Resolve ▸ Preferences ▸ System ▸ General**.
3. Under **External scripting using**, choose **Local** (or **Network**).
4. Open a **project** and a **timeline** — the bridge needs an active timeline to talk to.

Step 3 — Make the Resolve scripting module findable (if needed)

If the connection dot stays red even with Resolve open, Python may not be finding Blackmagic's scripting module. Add its path to your shell profile (`~/.zshrc`):

```
export PYTHONPATH="/Library/Application Support/Blackmagic Design/DaVinci Resolve/Developer/Scripting/M
```

Then reload it with `source ~/.zshrc` and relaunch both apps.

Step 4 — Launch order and connecting

- Start **DaVinci Resolve first**, open a project and timeline, then launch **ACE Editors' Notes**. (Launching Resolve after the app also works — the connection restores automatically.)
- Watch the connection dot: it turns **green** within about 5–10 seconds of a valid project/timeline being detected.
- The app auto-detects the current Resolve project and timeline and keeps them in sync every few seconds — no manual pairing.

Once the dot is green you're ready for Timecodes & Click-to-Seek.

> **The dot stays red?** Confirm, in order: (1) Resolve is running with a project and timeline open; (2) external scripting is enabled (Step 2); (3) `python3 --version` works; (4) the `PYTHONPATH` line is set (Step 3). A quick relaunch of both apps clears most transient cases. More in Shortcuts & Troubleshooting.

Where this leaves you

- **Offline / standalone:** ready now — go to Chapter 2.
- **Live with Resolve:** green dot means click-to-seek and marker import are live — go to Chapter 3.
- **AI features:** available if you're on macOS 26+ — go to Chapter 6.

SECTION 02

2 · Projects, Timelines & Notes

Everything in ACE Editors' Notes lives in a simple three-level hierarchy. Understanding it takes about a minute and explains how your notes stay organ…

02 · 2 · Projects, Timelines & Notes

Everything in ACE Editors' Notes lives in a simple three-level hierarchy. Understanding it takes about a minute and explains how your notes stay organized across many jobs and many cuts.

The hierarchy: Projects ▸ Timelines ▸ Notes

```
Project  (a job — "Nike Spring Spot", "Documentary Ep. 3")
  └─ Timeline  (a cut — "Rough Cut", "Director's Review", "Final")
       └─ Notes  (your writing for that timeline)
```

- A **Project** is the job. Create one per client, film, episode, or campaign.
- A **Timeline** is a cut or pass within that project. Each timeline keeps its **own independent notes**, so your rough-cut scribbles don't bleed into your final-review brief.
- **Notes** are the text you write for the selected timeline.

You choose the active project and timeline from the two dropdowns at the top of the window. Whatever project and timeline are selected is what the editor shows and what export/search act on.

> **How this maps to Resolve.** When the Resolve bridge is connected, the app auto-detects Resolve's current project and timeline and keeps the names in sync. You can also work entirely on your own project/timeline names offline — the hierarchy is yours to organize however suits the job.

Working with Projects

Create a project

1. Click **New Project**.
2. Enter a name — for example "Client Video Edit — Jan 2026".
3. The project appears in the dropdown, and a first timeline is created for you automatically.

Rename a project

- Use the **Rename** control next to the project dropdown, enter the new name, and it updates immediately.

Delete a project

- Use the **Delete** control and confirm. **This is permanent** and removes every timeline and note inside the project (deletion cascades). There is no undo for a deleted project, so export anything you want to keep first — see Chapter 7.

Working with Timelines

Create a timeline

- With a project selected, create a new timeline and name it for the pass it represents: "Rough Cut", "VFX Review", "Color Pass", "Final".

Rename / delete a timeline

- The **Rename** and **Delete** controls sit next to the timeline dropdown, mirroring the project controls. Deleting a timeline permanently removes its notes.

Find a timeline fast

- On projects with many timelines, use the **timeline filter** box — type a few letters (e.g. "rough") and the list narrows to matches. The filter is debounced so it stays responsive even with a long list.

Suggested convention

- One timeline per editing phase is the sweet spot. It keeps each brief focused and makes exports clean — you can hand off just the "Director's Review" notes without the noise of your working scratchpad.

The notes editor and the one-note-per-timeline model

The large central area is a **rich-text editor**. There is an important design choice to understand here:

Each timeline has one continuous note document. ACE Editors' Notes doesn't ask you to create discrete note "cards." You just write — line after line, timecode after timecode — into the single document for the selected timeline. Switching timelines swaps the document; your text for the previous timeline is preserved.

A typical timeline's document looks like this:

```
01:00:15:10 - Opening scene, director wants a wider lens
01:02:33:18 - Interview segment — great natural light
01:05:42:05 - VFX: remove boom-mic shadow (frames 2735-2890)
01:08:10:22 - Pacing slow, trim 8-10 frames between cuts
```

Each line is just text you typed; the timecodes at the start of each line are automatically recognized, highlighted, and made clickable (see Chapter 3).

> **Marker import is the exception.** When you pull markers in from Resolve, each marker arrives as its own structured note entry with a timecode, category colour, and comment — see Chapter 4. Everything you type by hand, though, flows into the one document per timeline.

Auto-save — there is no Save button

You never manually save. ACE Editors' Notes **auto-saves every 2 seconds** whenever the document has changed, writing to the local SQLite database with smart update logic (it updates the timeline's existing note rather than piling up duplicates). A brief **"Auto-saved"** message in the status area confirms your text is persisted.

Two practical habits:

- After a big edit, glance for the **"Auto-saved"** confirmation before quitting — it appears within a couple of seconds of you stopping typing.
- Because saving is automatic and local, closing the app mid-session is safe; your work for each timeline is already on disk.

Where your data lives

All projects, timelines, notes, and categories are stored in a single local **SQLite** database under ~/Library/Application Support/ (the folder keeps the legacy CutNotes name). This is a deliberate **local-first** design: no account is required to take notes, nothing is uploaded, and the app works in air-gapped edit bays. Back it up like any other project file, or use export (Chapter 7) for portable copies.

Next: the feature the whole app is built around — Timecodes & Click-to-Seek.

SECTION 03

3 · Timecodes & Click-to-Seek

This is the heart of ACE Editors' Notes. Every timecode in your notes is a live link to your Resolve timeline: click it, and Resolve's playhead jumps …

03 · 3 · Timecodes & Click-to-Seek

This is the heart of ACE Editors' Notes. Every timecode in your notes is a live link to your Resolve timeline: **click it, and Resolve's playhead jumps to that exact frame.** No copy-paste, no alt-tabbing, no hunting.

The problem it solves

The old loop: you keep notes in one app and edit in Resolve. Every time a note references a moment, you copy the timecode, switch to Resolve, click into the timecode field, paste, hit enter, then switch back. Dozens of times a session. It's slow and it breaks concentration.

ACE Editors' Notes collapses that to a single click. Your notes sit beside the timeline and navigate it.

How timecodes are recognized

You don't mark up timecodes or use any special syntax. As you type, ACE Editors' Notes watches for timecode patterns and, the instant one is complete, renders it **blue and bold** and makes it clickable. This highlighting is done live by a syntax highlighter — it never alters your underlying text, so exports and searches see the plain characters you typed.

Recognized formats:

YOU TYPE	INTERPRETED AS
`01:02:33:18`	Hours : minutes : seconds : **frames** (standard, frame-accurate)
`01:02:33`	Hours : minutes : seconds (no frame field)
`1:22:14`	Single-digit hours are accepted

For frame-accurate work, prefer the full **HH:MM:SS:FF** form. That's the format the **Insert Timecode** button produces and the format Resolve markers import as.

Two ways to put a timecode in a note

1. Type it

Just write it inline with your note. This is ideal offline, or when you're transcribing a director calling out times:

`01:05:42:05 - tighten this cut`

The moment `01:05:42:05` is complete it turns blue and becomes clickable.

2. Stamp Resolve's current playhead — Insert Timecode (needs Resolve)

When the bridge is connected, you can grab exactly where Resolve is parked instead of typing digits:

1. Park Resolve's playhead on the frame you're noting.
2. In ACE Editors' Notes, click **Insert Timecode** (the button reads "Insert Timecode (00:00:00:00)"), or press **⌘T**.
3. Resolve's current playhead timecode is inserted at your cursor, ready for you to type the note after it.

This is the fastest way to build accurate notes while you scrub: land on a frame, ⌘T, type the thought, repeat.

> **Offline behaviour.** With no Resolve connection, Insert Timecode still lets you place a timecode — you enter it manually in HH:MM:SS:FF form. The click-to-seek jump simply waits until Resolve is connected again.

Click-to-seek: clicking a timecode drives Resolve (needs Resolve)

With the connection dot **green**:

1. **Click any blue timecode** in your notes.
2. ACE Editors' Notes sends a seek command through the Python bridge.
3. **Resolve's playhead jumps** to that frame — effectively instantly.

The status area confirms the clicked timecode. That's the whole interaction: read your note, click the time, you're there.

What clicking needs to succeed:

- The connection dot is green (Resolve running, project + timeline open, bridge connected — see Getting Started).
- The timecode is a valid `HH:MM:SS:FF` / `HH:MM:SS` form.
- The timecode falls **within the active timeline's range**.

Timeline start offset — why `01:00:00:00` matters

Broadcast timelines very often **start at** `01:00:00:00` rather than zero. ACE Editors' Notes accounts for the timeline's start timecode when it seeks, so a note that says `01:00:10:00` lands where you expect on an hour-based timeline.

The practical consequences:

- On a timeline that starts at `01:00:00:00`, a timecode like `01:00:10:00` seeks correctly, while `00:00:10:00` may fall before the start of the timeline and not jump.
- If clicks seem to land in the wrong place, check **Resolve ▸ Timeline ▸ Timeline Settings ▸ Start Timecode**, and confirm your note timecodes are expressed in the same base as the timeline.
- Match your timeline **FPS** to the project FPS too — a frame field only means the same instant if the frame rate agrees.

Clicking does nothing? Quick checks

1. **Is the dot green?** Click-to-seek is a live feature; a red dot means standalone mode. Reconnect per Getting Started.
2. **Is the format valid?** It must read as `HH:MM:SS:FF` or `HH:MM:SS` and appear blue/bold. If it isn't highlighted, it isn't recognized.
3. **Is it in range?** A timecode before the timeline's start (the `01:00:00:00` offset case above) or past its end won't jump.
4. **Is the timeline active and unlocked in Resolve?** The playhead can only move on the active timeline.

More fixes in Shortcuts & Troubleshooting.

A note on the reverse direction

Clicking a note-timecode to move Resolve works today. The opposite automation — the app **creating a marker in Resolve** at a note's timecode — is written and ready but currently blocked by a Resolve 20.x API bug. That story, and the marker import that works great right now, is Chapter 4.

SECTION 04

4 · Markers: Importing from Resolve

Markers are how a lot of feedback already lives in Resolve — a colorist's flags, a director's review pins, your own scene markers. ACE Editors' Notes …

04 · 4 · Markers: Importing from Resolve

Markers are how a lot of feedback already lives in Resolve — a colorist's flags, a director's review pins, your own scene markers. ACE Editors' Notes can pull those markers straight into your notes as structured, clickable entries. This chapter covers importing (which works today) and is honest about marker creation (which is coded but blocked).

Import markers from your timeline (needs Resolve)

With the connection dot green:

1. Click **Import Markers** in the toolbar.
2. The **Marker Import** dialog opens, listing every marker on Resolve's active timeline in a table.
3. Review the list, choose whether to skip duplicates (see below), and confirm.
4. Each imported marker becomes a note entry in the current timeline's notes.

What comes across

For every marker, ACE Editors' Notes preserves the full metadata:

MARKER FIELD	BECOMES
Timecode (converted from Resolve's internal frame position)	A clickable blue timecode, offset-corrected for the timeline start
Name	The note's heading/title text
Note / comment	The note body
Colour	Retained and mapped to a category colour (Resolve's marker palette)

Because the timecode arrives as a normal recognized timecode, every imported marker is immediately **click-to-seek** — click it and Resolve jumps back to that frame (Chapter 3). And because colour is preserved, imported markers slot naturally into the category system (Chapter 5).

Frame-to-timecode conversion and the start offset

Resolve stores marker positions as **frame numbers**, not timecodes. The app converts each frame to a timecode using the timeline's frame rate and **start-timecode offset** — so a marker on an `01:00:00:00`-based broadcast timeline imports at the right hour-based timecode rather than at zero. If imported timecodes look shifted, check the timeline's start timecode and FPS in Resolve (same guidance as Chapter 3).

Duplicate detection

Re-importing after a few new markers were added shouldn't flood your notes with copies. The import dialog handles that:

- A **"Skip duplicates"** checkbox is **on by default**.
- Before import, the app scans your existing notes and flags any marker that already exists **at the same timecode**.
- With the box checked, those are left out and reported as skipped; only genuinely new markers are added.

This makes "import again to catch what's new" a safe, repeatable habit throughout a review cycle. Uncheck the box only if you deliberately want every marker brought in regardless.

Creating markers in Resolve — coded, but blocked (coded, blocked)

Here's the honest status, because it shapes how you should use the app.

The intended feature: turn a note into a Resolve timeline marker — write a note at a timecode, push a button, and a matching coloured marker appears on the Resolve timeline (colour driven by the note's category).

Why it isn't switched on: DaVinci Resolve **20.x** ships a bug in its scripting API where `Timeline.AddMarker()` returns failure for every call — even though manual markers (the **M** key) work fine and every other scripting call the app uses (seeking the playhead, reading markers, reading project/timeline info) works perfectly. It was tested exhaustively across parameter combinations; the method simply refuses under the current API.

What that means for you today:

- Markers flow **one way**: **Resolve → Notes** (import) plus **click-to-seek**. That's the shipping direction, and it's the workflow the app is built around.
- The marker-creation code is complete and in the app, waiting on Blackmagic. When a fixed Resolve build lands, the capability is expected to light up **without any change on your side** — just update Resolve.
- Until then, if you need a marker to exist in Resolve, place it manually with **M** in Resolve. You can still keep the authoritative, navigable record of it in your notes.

> **Why "one-way" is still the productive workflow.** The high-value move for most editors is the reverse of what you'd guess: not scattering markers onto the timeline, but pulling existing feedback out of Resolve into a searchable, categorizable, exportable brief — and being able to click any line to jump back. That is exactly what ships today.

Roadmap context

Bidirectional marker sync — including colour driven by category and delete-in-sync — is designed and partially built; it's gated on the Resolve API fix above. Deeper marker features (templates, presets, bulk operations) are longer-term roadmap items and are not part of the current beta. Nothing else in this manual depends on them.

Next: making notes readable and findable — Writing, Formatting, Categories & Search.

SECTION 05

5 · Writing, Formatting, Categories & Search

The day-to-day craft of using ACE Editors' Notes: rich text, colour-coding by department, and finding any note instantly.

05 · 5 · Writing, Formatting, Categories & Search

The day-to-day craft of using ACE Editors' Notes: rich text, colour-coding by department, and finding any note instantly.

Rich-text formatting

The editor supports the formatting you already know, applied to selected text or to what you type next:

ACTION	SHORTCUT
Bold	⌘B
Italic	⌘I
Underline	⌘U
Undo	⌘Z
Redo	⌘⇧Z

Standard macOS text editing (Copy ⌘C, Cut ⌘X, Paste ⌘V, Select All ⌘A) works throughout. A light convention that keeps briefs scannable: **bold for must-fix items, italic for open questions.**

Formatting is preserved when you export to PDF (Chapter 7). Timecodes are styled automatically (blue and bold) by the app and are separate from your own bold/italic/underline — you don't need to format them yourself.

Categories — colour-coding by department

Categories let you tag notes by the department or purpose they belong to, so a mixed review splits cleanly into VFX, Audio, Color, and so on. You assign a category with the **category selector** in the toolbar — a button showing the current category's icon and coloured dot; click it to pick another.

The six built-in categories

Every database ships with six built-ins:

CATEGORY	ICON	COLOUR
VFX	🎬	Red
Audio	🔊	Yellow
Color	🎨	Cyan

CATEGORY	ICON	COLOUR
Edit	✂	Green
Review	?	Blue
Note	📝	Pink

Note is the default safety-net category — any note without an explicit category falls back to it, and it can't be deleted.

Custom categories

You can add your own from the **category manager** (reached from the selector's menu). There you add, rename, recolour, and delete categories.

One deliberate constraint: **category colours are limited to Resolve's marker palette** (Red, Yellow, Green, Cyan, Blue, Purple, Pink). This isn't arbitrary — it means a category maps cleanly onto a Resolve marker colour, so that when marker creation is unblocked (Chapter 4) your categories translate to Resolve markers with no colour guesswork. It's also why imported markers, which carry Resolve colours, drop straight into the category scheme.

> **Colour continuity, both directions.** A marker imported from Resolve keeps its colour as a category; a category you create is a colour Resolve understands. The palette is the shared language.

Search — find anything, instantly

The **search bar** at the top of the window filters your notes as you type:

- Results update in **real time** — no enter key needed.
- Matching text is **highlighted in yellow** so you can spot it at a glance.
- The status area reports how many matches were found (or "No results").
- Search is **case-insensitive**.

Clear the search (delete the text, or the clear control) to return to the full document.

Search protects unsaved edits

Search is careful with work in progress. If you have **unsaved changes** in the editor, clearing a search won't blow them away by reloading from the database — your visible text is preserved. It only reloads from disk when your content is already saved. In practice you can search and clear freely without fear of losing a half-typed thought.

Search + export = a targeted brief

Search doubles as an export filter. When a search is active, **export operates only on the filtered (visible) notes** — so searching `VFX` and exporting gives you a VFX-only PDF for the compositor. This is one of the most useful handoff tricks in the app; see Chapter 7.

About AI-assisted categorization and search

Two AI conveniences relate to this chapter but live in Chapter 6 because they require macOS 26+:

- **Smart Categorization** (roadmap) — suggests a category from the note's content.
- **Semantic Search** (roadmap) — finds notes by meaning, so "audio problems" matches "boom shadow" and "wind noise" without those exact words.

The keyword search and manual categories described in this chapter work on **every supported macOS** and need no AI. The AI versions are enhancements layered on top, not replacements.

Next: the on-device AI features and what they require — On-Device AI.

SECTION 06

6 · On-Device AI: Note Polish & Voice Memo

ACE Editors' Notes has AI features built on a firm principle: your notes are about unreleased footage under NDA, so AI runs on your Mac by default. No...

06 · 6 · On-Device AI: Note Polish & Voice Memo

ACE Editors' Notes has AI features built on a firm principle: **your notes are about unreleased footage under NDA, so AI runs on your Mac by default.** Nothing is sent to a server unless you deliberately opt in to the optional cloud upgrade. This chapter covers the two AI features that ship, what they need, and the private-by-default design.

The privacy stance, stated plainly

Editors routinely work on material that can't leave the building. A director's note about "act two pacing" is not something to hand to a third-party API by default. So the design is **local-first, cloud-only-if-you-ask**:

- The shipping AI features run on **Apple's on-device models** — no network, no API key, nothing uploaded.
- A cloud option exists as an **explicit, per-feature opt-in** with a clear indicator when it's active (covered at the end of this chapter).
- The default, in every case, is on-device.

Requirement: macOS 26 or later (needs macOS 26+)

Both shipping AI features depend on Apple frameworks that only exist on **macOS 26+** with **Apple Intelligence** enabled and downloaded on supported hardware:

- **Note Polish** uses Apple's on-device **Foundation Models** (the built-in system language model).
- **Voice Memo** uses Apple's **Speech** framework in its on-device mode.

On earlier macOS, the two buttons are still visible but **disabled**, each with a tooltip explaining the requirement (for example, "Note Polish requires macOS 26+ with Apple Intelligence enabled"). This is the second, higher macOS floor referenced in Getting Started: the app runs on macOS 12; the AI needs macOS 26.

Everything else in this manual works without any of this.

Note Polish — raw notes into a director-ready brief

You take fast, messy notes during a review. Before sending them on, **Polish** rewrites them into a clean, professional, well-organized brief — same content, better presentation.

How to use it:

1. Write your note as usual.

2. Click **Polish Note** in the toolbar, or press ⌥⌘P (also under AI ▸ Polish Note).
3. The on-device model rewrites the text and shows you the result.

Design guarantees:

- **Nothing is auto-applied.** The polished version is presented for you to accept or reject — your original is never silently overwritten.
- **Timecodes are preserved verbatim**, so click-to-seek keeps working on the polished text.
- The model is instructed not to invent details — it reorganizes and tightens what you wrote, it doesn't add facts. (As with any language model, give the result a read before sending it on.)

Available when: macOS 26+, Apple Intelligence enabled, supported hardware. Otherwise the button is disabled with an explanatory tooltip.

Voice Memo — dictate a note, transcribed on-device

Sometimes it's faster to talk than type — especially mid-scrub in Resolve. Voice Memo captures a spoken note and drops the transcript into the editor.

How to use it:

1. **Press and hold** the **Hold to Record** button.
2. Speak your note — e.g. "VFX department needs to remove the boom-mic shadow here, frames 2735 to 2890."
3. **Release** the button. The transcript is inserted into the current note.

Timecode stamping: if Resolve is connected, the moment you start recording the app captures Resolve's playhead timecode and stamps the transcribed note with it — so a dictated note is anchored to the frame you were looking at, and is click-to-seek like any other (Chapter 3).

Privacy design:

- Transcription is **on-device** (the Speech framework's offline mode is enforced), so audio isn't sent anywhere for recognition.
- **Audio is never written to disk.** It's transcribed live and discarded — only the resulting text stays.
- First use prompts once for **Microphone** and **Speech Recognition** permission; allow both for the feature to work.

Available when: macOS 26+ with on-device Speech support. On-device recognition quality is strongest for English; the app enforces the on-device path rather than falling back to cloud. If your Mac lacks Speech support the button is disabled with a tooltip.

The optional cloud upgrade — strictly opt-in

For anyone who wants frontier-model quality and accepts the trade-off, ACE Editors' Notes can route AI through the cloud instead of the on-device model. This is **off by default** and designed to be unmistakable when on.

- Configured in **Preferences**: you supply your own **Anthropic (Claude) API key**, stored in the macOS **Keychain** — never in the app's database or a plist.
- **Per-feature toggles:** even with a key stored, each feature (e.g. Polish) stays on-device unless you explicitly switch that feature to cloud. A feature uses the cloud only when a key is present **and** its toggle is on.
- **Visible when active:** a cloud-powered run is labelled (a "via Claude" badge) so you always know when note text left your Mac.
- Note text is sent **only** when you invoke a cloud-enabled feature — there's no background syncing.

If you have an ACE Suite account, the app can also sign in from Preferences to carry your entitlement; the AI defaults remain on-device regardless. When in doubt, the safe default — and the shipping default — is **local**.

Roadmap AI (not in the current beta)

The AI plan sequences further on-device features after Polish and Voice Memo — **Smart Categorization** (suggest a note's category), **Semantic Search** (find notes by meaning), and **Cross-note Insights** (e.g. "summarize all VFX notes for this timeline"). These are designed local-first as well but are **not part of the current beta**; treat them as direction, not shipped features. The keyword search and manual categories in Chapter 5 cover today's needs without AI.

Next: getting your notes out of the app — Exporting & Handoff.

SECTION 07

7 · Exporting & Handoff

Notes are only useful if they travel. ACE Editors' Notes exports to PDF and TXT in one click, so you can email a brief to the director, hand the cut t…

07 · 7 · Exporting & Handoff

Notes are only useful if they travel. ACE Editors' Notes exports to **PDF** and **TXT** in one click, so you can email a brief to the director, hand the cut to the lead editor, or drop a plain-text list into a task tracker.

Export to PDF

For a formatted, presentable document — the usual choice for sending to a director, colorist, or client:

1. Click **Export to PDF**.
2. Choose a save location.
3. A formatted PDF is generated (a progress bar shows during export).

The PDF preserves your **rich-text formatting** (bold, italic, underline) and your **timecodes**, and is headed with the project and timeline names so the recipient knows exactly which cut the notes belong to. It's ready to email or print.

Export to TXT

For a lightweight, universally openable file — good for pasting into email, a task tracker, or a shot list:

1. Click **Export to TXT**.
2. Choose a save location.
3. A plain-text file is written with all the notes for the current timeline.

TXT drops styling (it's plain text) but keeps the content and the timecodes as written.

Export exactly what you need — filtered export

This is the export feature worth remembering: **when a search is active, export includes only the filtered notes** (Chapter 5).

So the targeted-handoff pattern is:

1. **Search** for the slice you want — e.g. `VFX`, or a shot name, or a reviewer's initials.
2. Confirm the editor is showing just those matching notes.
3. **Export to PDF or TXT** — the output contains only that slice.

Search `Audio`, export → an audio-only brief for the mixer. Search `VFX`, export → a shot list for the compositor. One document, one department, no manual trimming.

To export the entire timeline instead, clear the search first so the full document is visible.

Print

Because PDF export produces a fully formatted document, printing is simply "export to PDF, then print" — the PDF is print-ready and carries the same project/timeline heading and formatting. This is the intended path for a paper handoff in the edit bay.

The handoff workflow, end to end

A typical assistant-editor-to-lead handoff after a director's review:

1. During the review, take notes — stamp Resolve's playhead with **⌘T** (Chapter 3) and categorize as you go (Chapter 5).
2. Optionally **Import Markers** to fold in anything already flagged in Resolve (Chapter 4).
3. Optionally **Polish** the notes into a clean brief (Chapter 6).
4. **Export to PDF** for the full brief, and/or **search + export** department-specific slices.
5. Send them on. Every timecode in the notes stays meaningful — anyone with the project open in Resolve and ACE Editors' Notes installed can click a timecode and land on the frame.

The result is a brief that travels with the cut: stamped, categorized, searchable, and navigable — no transcription lag, no lost sticky notes.

What export is not (today)

- There's **no round-trip back into Resolve** from an export — exports are documents for people, not data for Resolve. (Marker creation, the app-to-Resolve direction, is the separate blocked feature in Chapter 4.)
- There's **no cloud share link or multi-user handoff** — sharing today means sending the exported file. Cloud sync and collaboration are longer-term roadmap items, not part of the current beta.

Reference material — the full shortcut list and fixes for common problems — is in the appendix.

SECTION 08

Appendix - Keyboard Shortcuts & Troubleshooting

One place for every shortcut and the fixes for the problems people actually hit. ACE Editors' Notes is macOS-only, so all shortcuts use Mac modifiers:...

08 · Appendix · Keyboard Shortcuts & Troubleshooting

One place for every shortcut and the fixes for the problems people actually hit. ACE Editors' Notes is macOS-only, so all shortcuts use Mac modifiers: ⌘ Command, ⌥ Option, ⇧ Shift.

Keyboard shortcuts

Editing

ACTION	SHORTCUT
Bold	⌘B
Italic	⌘I
Underline	⌘U
Undo	⌘Z
Redo	⌘⇧Z
Copy / Cut / Paste	⌘C / ⌘X / ⌘V
Select All	⌘A

Timecodes & Resolve

ACTION	SHORTCUT
Insert Timecode (stamps Resolve's playhead when connected)	⌘T
Seek Resolve to a timecode	**Click** the blue timecode

Search & AI

ACTION	SHORTCUT
Search notes	⌘F
Polish Note (AI)	⌥⌘P
Voice Memo (AI)	**Press and hold** the Hold to Record button

> Voice Memo has no key shortcut by design — it's a press-and-hold ("hold-to-talk") control, so you use the toolbar button.

Requirements recap

CAPABILITY	NEEDS
Notes, timecodes, categories, search, export	macOS 12+ (Apple Silicon)
Click-to-seek, Insert-from-playhead, marker import	DaVinci Resolve 18–20 running + Python 3 bridge connected
Note Polish, Voice Memo	macOS 26+ with Apple Intelligence
Cloud AI (optional)	Your own Anthropic API key, opt-in per feature

Full setup detail is in Getting Started.

Troubleshooting

"ACE Editors' Notes is damaged and can't be opened"

A stale or quarantined download (or an early unsigned build) can trip Gatekeeper. In **Terminal**:

```
xattr -cr /Applications/ACEEditorsNotes.app
```

Then open the app again. One-time fix. The shipping public beta is Apple-signed and notarized, so a fresh download from the ACE site normally opens without this.

The Resolve connection dot stays red

Work through these in order:

1. **Resolve is running** with a **project and timeline** open.
2. **External scripting is enabled** — Resolve ▸ Preferences ▸ System ▸ General ▸ External scripting using ▸ Local (or Network).
3. **Python 3 works** — `python3 --version` returns a version.
4. **Resolve's scripting module is findable** — the `PYTHONPATH` line from Getting Started is in `~/.zshrc` and sourced.
5. **Relaunch both apps.** Start Resolve first, then ACE Editors' Notes. Allow 5–10 seconds for the dot to turn green.

Clicking a timecode does nothing

1. The dot must be **green** — click-to-seek is a live feature.
2. The timecode must be a valid `HH:MM:SS:FF` / `HH:MM:SS` form and appear **blue and bold** (if it isn't highlighted, it isn't recognized).
3. It must be **within the active timeline's range** — note the `01:00:00:00` start-offset case in Chapter 3.
4. The timeline in Resolve must be **active and unlocked**.

Imported markers land at the wrong timecode

Resolve stores markers by frame; the app converts using the timeline's FPS and start timecode.

1. Check **Resolve ▸ Timeline ▸ Timeline Settings ▸ Start Timecode** (broadcast timelines often start at `01:00:00:00`).
2. Confirm the timeline **FPS** matches the project FPS.
3. As a test, a timeline starting at `00:00:00:00` removes the offset from the equation.

I can't create markers in Resolve from my notes

That's expected right now — it's the **coded-but-blocked** feature. DaVinci Resolve 20.x has a scripting bug (`AddMarker()` fails for all calls) that blocks it. Import (Resolve → Notes) and click-to-seek are unaffected. The capability is expected to work automatically once Blackmagic ships a fix — full explanation in Chapter 4. In the meantime, place markers manually with **M** in Resolve.

The Polish or Voice Memo button is greyed out

Those need **macOS 26+** with Apple Intelligence enabled on supported hardware. On earlier macOS the buttons are disabled with a tooltip. Everything else in the app still works. See Chapter 6.

Exported PDF is blank

Export captures the current editor content for the selected timeline. Make sure notes are actually written (and, if a search is active, that some notes match — export only includes filtered notes). Clear the search to export the whole timeline.

My notes "disappeared" after searching

They didn't. Search filters the view; **clearing the search restores everything**, and any unsaved edits are deliberately preserved rather than reloaded away (Chapter 5).

Auto-save

There is no Save button — the app auto-saves every ~2 seconds when the document changes, and shows an **"Auto-saved"** confirmation. Glance for it before quitting after a large edit.

Known limits of the current beta

- **macOS only, Apple Silicon.** No Windows, no Intel build in the shipping beta.
- **Marker creation into Resolve is blocked** by the Resolve 20.x API bug (import + click-to-seek work).
- **AI needs macOS 26+.** Note Polish and Voice Memo are unavailable on earlier macOS.
- **Local-only.** No cloud sync, no multi-user collaboration — sharing is via exported PDF/TXT. Both are longer-term roadmap items, not part of this beta.
- **One note document per timeline** — the model is a continuous document per timeline, not discrete note cards (imported markers being the structured exception).

Back to the manual index.