Lacquer — System Design

An archival capture system. “MPD in reverse”: instead of a daemon that plays from a library, Lacquer is a daemon that captures into one.

Version: 0.3.1 (2026-06-04) — see Revision history at the end of this document. **Status:** design document, pre-implementation. **Scope:** the complete intended system. The Phasing section defines what to build first; most of this document describes the destination, not the v1.

Table of contents

1. Concept and philosophy

MPD (Music Player Daemon) is a server that plays audio from a library, controlled by thin clients. Lacquer inverts this: it is a server that *captures audio (and images, and video) into* an archive, fed by thin capture "stations." One archive; many stations; the stations can be anything from a high-end fixed vinyl rig (a strain-gauge cartridge into a quality ADC, say) to a laptop ripping CDs at a friend's house to an ordinary tape deck plugged into a cheap interface.

The governing value is **laziness**, in a specific and deliberate sense: the act of archiving should demand as close to zero effort and zero decisions as possible at capture time. You should be able to clean a record, drop the needle, walk away, and have the system capture a pristine master, file it, attempt to identify it, and — crucially — *remember what it still doesn't know* and surface that to you later, at a good moment, without nagging. Metadata is added incrementally, whenever you feel like it, or never. A bare untagged audio dump is a complete, valid, first-class archive entry, not an error state.

This laziness is not sloppiness. The archive's *integrity* is sacred — masters are immutable and verified — but the archive's *completeness* is allowed to be a work in progress forever. The system's job is to make the pristine capture effortless and to be an excellent, patient librarian about everything else.

The long-term ambition is a small business built around perfecting archiving: ultimately a fully-FOSS appliance an archivist or music lover can plug in and use without touching a config file, that does its sense-making on-device, and that can federate into a trust-based peer network where archivists back each other up and browse each other's collections. But the starting point — and the thing that must be excellent before anything else — is single-box "MPD in reverse."

The deeper mission is preservation through distribution. A great deal of recorded music exists today in only a handful of physical copies — rare pressings, regional releases, private cuts, tapes that were never reissued and never will be. The objects themselves are mortal: vinyl warps, tape sheds, lacquers rot, a house fire or a flood or a death in the family ends a collection. Streaming services are not an archive — they are a licensing arrangement that can drop a recording overnight and that you cannot bequeath. The most reliable way humanity has ever found to keep fragile cultural artifacts alive is *many independent good copies in many independent hands* — the principle behind every surviving manuscript tradition, and behind LOCKSS ("lots of copies keep stuff safe") in the digital era. Lacquer is built to make that happen for recorded sound, almost as a byproduct of how it already works:

- **Many independent copies = survival.** Because masters are immutable and content-addressed,

"backing up to a peer" is just replicating hash-verified blobs (§15.3). A rare cut held by one archivist is one fire away from gone; the same cut mirrored across a handful of trusted peers is, for practical purposes, permanent. No central server has to exist or stay funded for the recording to survive.

- **Equipment diversity is itself a preservation asset.** The system captures from whatever gear an archivist owns and records the exact chain that produced each master (§5.4). When several archivists independently transfer the same rare pressing on different rigs, the network ends up holding *multiple independently-sourced, best-effort masters* of it — and can keep them all, compare them, and converge on the best version, rather than betting everything on one transfer. Better gear anywhere in the network raises the ceiling for everyone; no one's modest transfer is ever the floor.
- **Hash-verified integrity across the whole network.** Every copy everywhere is checkable against its hash, so silent bit-rot or tampering is detectable wherever a master lives — preservation you can *prove*, not just hope for.
- **Collections outlive both the objects and their owners.** This is the part that makes it personal: you cannot hand your children a thousand pounds of vinyl and paper they may not want, and you are certainly not handing them a Spotify account. But you *can* hand them — or the network — the masters: the actual recordings, captured at the best fidelity your equipment could manage, verifiable, playable, and theirs, with the provenance of how each one was made. A collection becomes something that can be inherited, gifted, or simply kept alive by the community of archivists who care about it, long after the shelf of records is gone.

So the network is not merely a convenience for backup and browsing. It is the point at the largest scale: **a permanent, redundant, equipment-diverse underground of archivists, each working at the highest fidelity they can, keeping the rarest recordings in the world alive and circulating — and always reaching for the best possible version of each one.** A living commons: preserve, circulate, and collaboratively improve. Everything below is, in the end, the machinery that makes such a commons safe, automatic, and honest.

---

2. Governing principles

Every design decision in this document reduces to one or more of these. They emerged over long design iteration; their stability (new requirements kept folding into them rather than demanding new mechanisms) is the main evidence the architecture is sound.

2.1 Immutable master + mutable catalog

Every captured artifact (an audio master, a cover scan, a runout photo, a performance video) is stored by content hash in a content-addressed store (CAS) and **never edited**. The editable "what we think this is" layer — identity, tags, track boundaries, relationships — is a *separate* Postgres catalog that points at hashes. You can delete the entire catalog and rebuild it from the immutable sidecars. The masters are the archive; the catalog is an interpretation of it.

2.2 Labeled regions + region-aware renders

A master holds *everything* captured: lead-in noise, the needle drop, the program, mechanical sounds, run-out groove, locked grooves, the lot. Trim points are **not destructive edits** — they are frozen metadata markers that label regions of the master. The files people actually play (the "renders") are produced by selecting which regions to include. The clean version and the full-mechanical-ambience version are two renders over one untouched master.

Two refinements of the master/render line matter enough to state here, because the rest of the document leans on them. Both are developed in full at §5.5–§5.6; the point now is only that they do not bend the principle — they sharpen it.

- **The word "master" means exactly one thing — an untouched capture of a physical source object — and nothing else ever earns the name.** A recording can have *several* masters, at different **source generations**: a commercial pressing, the lacquer it was cut from, the production master tape behind that, and the original multitrack session reel are four physically distinct objects in one recording's lineage, and a capture of *any* of them is a master. None is "more master" than another; they differ by generation, not by status (§5.6). This matters because a capture of the production tape is a categorically better source than a capture of a worn pressing, and the archive should prefer it knowingly.
- **A render may be expensive, but it is still a render.** Some derived outputs — a Dolby-decoded transfer, an analog de-hiss, a painstaking manual restoration — cost real, sometimes non-reproducible effort. That does not make them masters; it makes them renders carrying a `precious` flag that tells the backup layer to protect them like masters even though they are not (§5.5). Pushing "expensive" onto a render flag, rather than letting it promote a derived file to "master," is what keeps the immutability guarantee from eroding by language alone — a thing called "master" that was in fact processed would quietly destroy the one guarantee the principle exists to give.

2.3 Propose-once → confirm → freeze → derive

Anything ambiguous — where the music starts, the audio/video sync offset, the Dolby type, the release identity — is *proposed once* at ingest (by detector, fingerprint, or LLM), surfaced for a single human

confirmation (or auto-accepted only at very high confidence), then **frozen** as stored metadata and reused deterministically forever. Renders never re-guess at render time. This is what keeps the quiet intro of *Wish You Were Here* from getting clipped by a heuristic that runs differently each time.

2.4 Local-first, always

External services (MusicBrainz, Discogs, AcoustID, a remote LLM, even a federated peer) are *caches and conveniences, never dependencies*. Everything retrieved is copied into the local catalog with provenance at retrieval time. The system fully functions offline: external lookups enrich completeness; their absence never blocks ingest, playback, or the VFS (the virtual filesystem that exposes renders as browsable files — see §12). If MusicBrainz vanishes or a peer goes dark, nothing you hold is lost. (The LLM mentioned here and in §7 is one such optional helper — §10 explains what it is and why it earns its place.)

2.5 Three declarative generators over the immutable core

The system's flexibility lives in three small declarative "languages," each interpreted by a generic engine:

- **Source profiles** — how a kind of media is captured and what master it yields.
- **Format profiles** — how masters are rendered into output files.
- **View profiles** — how output is arranged into browsable trees.

Adding a new preservation source, a new output codec, or a new way of organizing the library is *writing a spec*, not modifying engine code. Specs can be authored by hand or by an LLM that understands the schema. This is why "12-bit MP2 for an Amiga," "7.1.4 Atmos," and "reel-to-reel support" are all configuration, not rewrites.

2.6 Equipment-agnostic, across the whole quality spectrum

Lacquer is a general project for many people with wildly different hardware, not a spec for any one person's rig. The system models capture inputs by *type, device binding, and signal chain* (§6.1), and gear by *capability* (e.g. "auto-azimuth vs. fixed-azimuth deck," "discrete-4-channel vs. matrix source"), with the specific component recorded honestly in an equipment registry. This document names real hardware freely — a Nakamichi deck, a strain-gauge cartridge, particular ADCs or cameras — but always as a *concrete example of a phenomenon* (some pre-recorded tapes used Dolby C; some playback chains have real per-play cost) or a *best-case reference point*, never as assumed or required equipment.

The guiding stance: for anything you care about, use the best equipment you can get hold of, because better gear yields better masters and an archive is worth doing well. But reference-grade gear is expensive and rare (a top-tier cassette deck can run several thousand USD; an audiophile cartridge and ADC

likewise), and most participants simply won't have it. The system is designed to capture just as happily from an ordinary single tape deck into a cheap interface or a USB turntable into onboard audio. The equipment chain records what was actually used, so the provenance always reflects the true quality of the transfer — and a modest-gear capture is a valid, well-documented, first-class archive entry, not a second-class one. Better gear raises the ceiling; it is never the floor.

A corollary the system honors rather than fights: **capture must never degrade the thing being captured.** On a serious listening chain the playback itself has value — and sometimes a real per-play cost — so Lacquer taps the signal without inserting itself into the path that reaches the speakers (§6.7). The archive is a passive observer of a good listening session, not a toll on it; the best capture and the best listening experience are the same event, not a trade-off.

2.7 A corollary: two metadata planes

Metadata divides into two planes that must never mix:

- **Intrinsic** — about the *work*: artist, title, tracklist, the equipment chain that made the recording. May be baked into file tags and travels with shared files.
- **Administrative** — about *our handling* of the work: federation provenance, fetch dates, completeness scores, task state, capture-session bookkeeping, which peer sent something. Lives **only** in the local catalog. Never written to tags, never embedded in shared files.

This keeps shared files clean and identical regardless of who shares them (which also helps CAS dedup across a federation).

---

3. System architecture

Three programs, mirroring MPD's daemon/client split:

- `lacquerd` — the central archive daemon. Owns the CAS master store, the Postgres catalog, the render engine, the view/VFS layer, the protocol front-ends, the enrichment workers, and the reminder/job engine.
- `stationd` — the capture agent. One static binary, capability-driven by config, runs on every capture machine (fixed rigs and the roaming portable laptop). Pairs to `lacquerd` over an authenticated channel; spools locally and forwards when offline.
- `lq` — the client: a CLI/TUI (the `mpc` analog) for review, metadata, and the reminder queue, plus a

web UI for image-heavy review and (later) appliance/federation setup. `lq` is a *client of* `lacquerd`*'s control protocol* (the minimal MPD-style line protocol established in Phase 1, §16) rather than a privileged path into the daemon — so any future client, including a mobile app, can do everything `lq` can. The control protocol carries browse/review/transition commands and the LLM conversation; it deliberately does **not** carry media playback, which stays on the stream-class front-ends (§13).

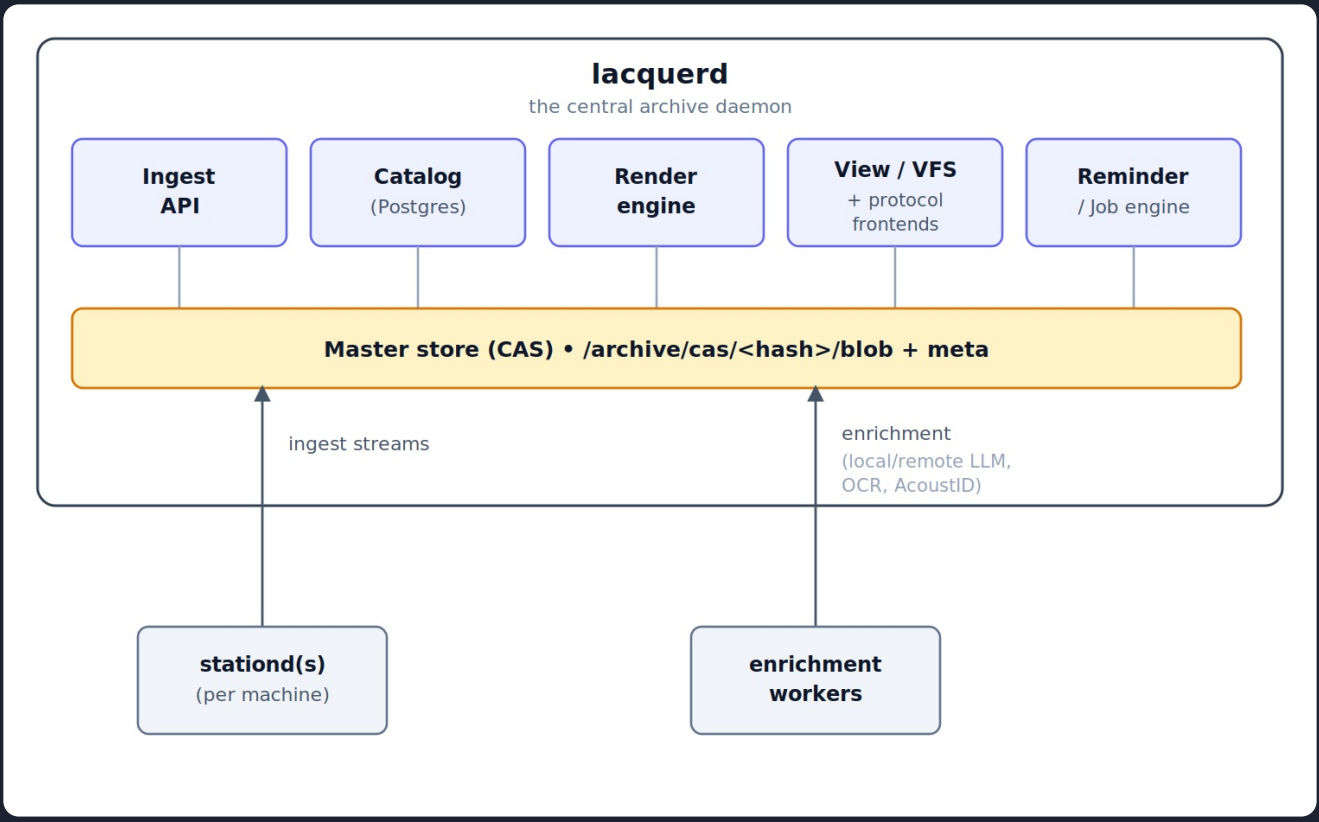


Figure: system architecture. (An editable vector version, `architecture.svg`*, accompanies this document.)*

The whole thing scales from a single box (one machine holding every role) to a fleet, with no architectural difference — multi-machine is a deployment fact, not a design requirement.

---

4. The master store (CAS)

Every immutable artifact is stored content-addressed, by BLAKE3 hash:

```
/archive/cas/<bl/ak/e3…>/blob        # the raw bytes, immutable
/archive/cas/<bl/ak/e3…>/meta.json   # immutable capture facts (a sidecar)
```

The sidecar records only *facts about the capture event* — never editable opinions:

```
{
  "blob_hash": "blake3:8a3f…",
  "kind": "audio_master",
  "captured_at": "2026-06-02T14:11:03Z",
  "machine": "studio-box-01",
  "input": "main-vinyl",
  "source_type": "vinyl",
  "audio": { "rate": 88200, "bits": 24, "channels": 2, "codec": "flac" },
  "duration_s": 2412.6,
  "ingest_session": "uuid-…",
  "equipment_chain": ["soundsmith-sg-210", "soundsmith-preamp", "rme-adi-2"]
}
```

Why CAS:

- **Dedup is automatic** — rip the same CD twice, get one blob.
- **Integrity is intrinsic** — the hash *is* the verification; a blob cannot be silently altered without changing its identity.
- **The catalog is disposable** — rebuildable from sidecars.
- **Federation falls out for free** — two archivists holding the same master hold the same hash, so cross-network dedup, integrity, and "send me the blobs with these hashes" replication all work without extra machinery (see §15).

Renders (derived, lossy or otherwise) are **not** masters; they live in a parallel rendered-output store keyed by `(master_hash, profile)` and are regenerable at will (see §12).

---

5. The catalog data model

The catalog is conceptually MusicBrainz-like but adds a tier MusicBrainz lacks — the *physical copy you hold* — because the system must distinguish "a different pressing" from "a different physical disc of the same pressing," and must keep both forever.

5.1 The core hierarchy

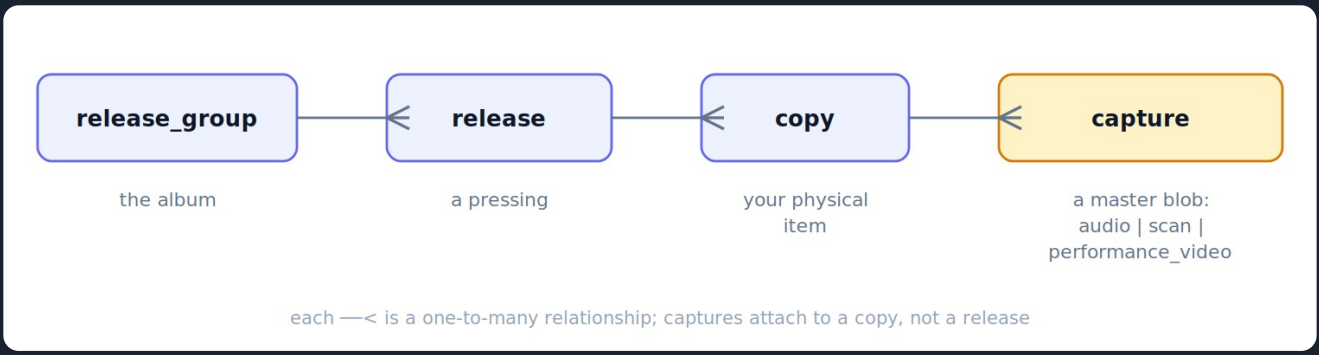


Figure: the catalog hierarchy. (An editable vector version, `hierarchy.svg`, accompanies this document.)

- **`release_group`** — the album as an abstract title (*Kind of Blue*). Groups everything.
- **`release`** — a specific pressing/edition: label, catalog number, country, year, matrix family. "Some pressings are better than others" lives here. Many releases per group.
- **`copy`** — *your physical item*: this specific disc/record/tape, with condition grading and acquisition info. **Captures attach to a copy, not a release.** This is the tier that makes "keep everything, prefer the best" work.
- **`capture`** — an immutable master blob produced from a copy, plus its capture facts. A copy may have **many captures**, including multiple of the same logical content (e.g. the same cassette side transferred on two decks, or with Dolby on and off).

Supporting entities: `artist`, `label`, `artist_credit` (the usual graph); `asset` (non-audio blobs: cover scans, j-cards, labels, matrix photos, performance video); `ingest_session` (one capture event, links machine+input to the blobs produced); `equipment` (the registry of known gear referenced by chains); `task` (the reminder queue); `proposal` (LLM/fingerprint suggestions awaiting confirmation); `peer` and `grant` (federation, §15).

5.2 Preferred copy / preferred capture

Replacement is **non-destructive**. You never overwrite a worse pressing with a better one; you ingest a new `copy`, capture it, and flip a **`preferred`** pointer. The old copy stays archived; it's just no longer the default the VFS surfaces.

```
{
  "copy_id": "uuid",
  "release_id": "uuid",
  "condition": "VG+",
  "acquired_at": "2026-05-30",
  "notes": "light scuff side B",
  "preferred": true,
  "preferred_scope": "release"   // or "release_group"
}
```

Within a copy, individual captures carry `preferred_for_side` / `preferred_for_track`, chosen *after auditioning* (e.g. the CR-7A flat transfer beats the Dragon on a tape with stable azimuth; the Dragon wins on a drifting one). The VFS resolves preferred-copy → preferred-capture when you browse; an `all_copies` / `all_captures` view exposes everything for A/B comparison.

5.3 Completeness

Every release and copy carries a computed **completeness** record. This is the engine of laziness: incompleteness is a valid, tracked, first-class state, not an error.

```
{
  "has_audio": true,
  "has_artist": false,
  "has_title": false,
  "has_cover": true,
  "has_tracklist_split": false,
  "matrix_read": "pending",
  "rip_verification": "accuraterip_match",   // for CDs
  "dolby_decode_status": "needs_manual_pass", // for cassettes
  "score": 0.42,
  "blocking": ["artist", "title"]
}
```

A bare audio dump with no metadata is completeness ~0.15 and **fully ingested**. Anything below 1.0 is a candidate for the reminder engine (§14). Completeness for a release (metadata) and a copy (capture quality) are scored independently.

5.4 Equipment chain as first-class provenance

Every capture records its full signal/imaging chain as an ordered list of typed components drawn from an **equipment registry** — where each component a user owns (whatever it is) is defined once and referenced. The names below are illustrative registry entries, not required hardware:

```
"equipment_chain": {
  "capture_type": "vinyl",
  "stages": [
    { "role": "cartridge",    "ref": "soundsmith-sg-210-strain-gauge" },
    { "role": "phono_stage",  "ref": "soundsmith-preamp" },
    { "role": "turntable",    "ref": "technics-sl1200-mk7" },
    { "role": "adc",          "ref": "rme-adi-2", "rate": 88200, "bits": 24 }
  ]
}
```

A registry (rather than free text) makes provenance queryable (“everything captured on the strain gauge”), consistent, and correctable in one place. It is **intrinsic** metadata — it propagates to tags (§13) and is part of the archival truth of *how the sound was made*. The strain-gauge cartridge is an instructive example of why this matters: it is not a magnetic cartridge and requires its own dedicated phono stage, so a chain that records it documents something real about the transfer — but the same registry equally describes a modest moving-magnet cartridge into an onboard sound card. It captures whatever was actually used, honestly.

The same chain concept covers imaging, not just sound. A visual capture (§8) records *how the image was made* with the identical structure — camera/scanner, optics, lighting, capture geometry — so the archive can reason about visual quality the same way it reasons about transfers:

```
"equipment_chain": {
  "capture_type": "visual",
  "stages": [
    { "role": "camera",   "ref": "canon-eos-r5" },
    { "role": "lens",     "ref": "rf-100mm-macro" },
    { "role": "surface",  "ref": "glass-platen" },
    { "role": "lighting", "ref": "dual-45-diffused" }
  ]
}
```

A flatbed scan records the scanner and DPI; a handheld phone photo records what its EXIF carries, or an explicit “unknown handheld” — never simply blank. So “show me every sleeve still captured by phone” and “everything shot on the copy-stand” are the same kind of first-class query as their audio counterparts, and the provenance honestly reflects the real quality of each image just as it does each transfer.

5.5 Render flags: `precious` and `canonical`

A render is, by default, a cheap and disposable thing: derived from a master by a deterministic recipe, evictable, and re-derivable byte-identically on demand (§12.2). Two independent flags on a render handle the cases where that default is wrong. They answer two different questions — *“how hard would this be to make again?”* and *“is this the version a player should reach for?”* — and a render may carry either, both, or neither.

A useful way to hold the whole master/`precious` distinction in one’s head is to ask: **can the system regenerate this artifact on its own, or did a human (or a tool the pipeline cannot drive) have to make it?** A master can never be regenerated — it is a one-time capture of a physical object, and the object may be gone or degraded by the next play. A cheap render *can* be regenerated by the pipeline at will. The interesting middle case is the artifact that is *derived* (so not a master) but that the pipeline *cannot* reproduce on its own — a hand-run decode, a manual restoration — and that is exactly what `precious` marks. This is a “can the software remake it?” line, not a strict bit-determinism line: ordinary lossy renders are not even byte-identical run to run (a fresh MP3 encode may differ slightly from the last), but the pipeline can still remake an *equivalent* one unattended, which is what matters for whether it needs master-grade backup. `precious` is for the renders where even that unattended remake is impossible.

`precious` **— “back this up like a master.”** Most renders cost milliseconds of CPU and can be thrown away without a thought. But some derived outputs are the product of effort that is expensive or simply *not reproducible by the system*: a Dolby decode run by hand through a proprietary tool (§7.1), an analog-domain de-hiss or click-repair pass, a restoration an archivist labored over, a quad decode that only exists because someone ran the one piece of software that can do it. These are still renders — they have a parent master, they are the output of a transformation, they are honest about not being the capture — but they cannot be cheaply regenerated, so treating them as disposable would mean losing real work. The `precious` flag tells the backup layer (§15.3, §18) to replicate and protect that render with the same seriousness as a master. It changes *how the render is stored and backed up*; it does not change *what the render is*. The master/render line (§2.1) stays intact: `precious` is the system admitting that not all renders are cheap, without ever calling a processed file a master.

`canonical` **— “this is the version to play by default.”** Normally the default render of a master is obvious: the clean program in the universal format set (§12.1). But for some sources the raw master is

deliberately *not* the thing a listener should hear. A flat (Dolby-off) cassette master (§7.1) is archival truth, but the version a player should default to is the *decoded* one. A CD-4 capture is a 2-channel signal carrying an ultrasonic subcarrier (§7.3) — unpleasant heard directly; the canonical playable version is the demodulated/fold-down render. So a render can be marked `canonical`, meaning "when a client asks for this recording without specifying a variant, serve this, not the bare master." `canonical` is itself a frozen, propose-once decision (§2.3): the system proposes which render should be canonical (e.g. "this flat master needs a Dolby-B decode to be its canonical playable version"), a human confirms once, and thereafter the VFS resolves the recording to the canonical render deterministically. Crucially, a master can be flagged as *needing* a canonical render it does not yet have — a flat Dolby-S tape has no decoder anywhere (§7.1), so it is `canonical: needs_manual_pass` indefinitely, and the reminder engine (§14) tracks the gap. The flat master is never lost; it simply is not yet the thing you press play on.

The two flags interact cleanly and predictably. A hand-made Dolby decode of a difficult tape is typically *both* `precious` (it took manual work through a proprietary tool, back it up) *and* `canonical` (it is the version to play). A cheap automated Dolby-B decode is `canonical` (play it) but not `precious` (the pipeline can regenerate it from the flat master at any time). A one-off Opus encode for some device is neither. And a flat master that is waiting on a decoder that does not exist yet is a master with an unfilled `canonical` slot — perfectly valid, fully ingested, tracked as incomplete (§5.3, §14). Nothing here is a new kind of object; it is two attributes on the render entity, both honoring propose-once-freeze, both leaving "master" meaning exactly one thing.

5.6 Masters at source generations

The hierarchy of §5.1 quietly assumed one master per recording. The reality of serious archiving breaks that assumption in a way the data model already accommodates — and naming it makes a powerful preference fall out for free.

A recording exists at several stages of a **production lineage**, and a physical object can survive at any of them:

```
multitrack session reel  →  production master tape  →  lacquer / acetate  →  metal stamper  →  commercial pressing
   (raw session tracks)        (the mixed master)         (the cutting)          (the mold)         (what you buy in a shop)
```

Each is a *physically distinct object*, and a capture of any of them is, by the §2.1 definition, a **master** — an untouched digitization of a real source object, immutable, the truth of that capture. A vinyl rip is a master (of the pressing). A transfer of the production master tape is *also* a master (of a more upstream source). A capture of the multitrack reel is *also* a master (of the most upstream source of all). These are not derivatives of one another; they are independent captures of different physical objects that happen to be different generations of the same recording's lineage. None is "more of a master" than the others — but

they sit at different **generational depths**, and the depth is preservation-critical information worth recording.

The data model bends to this without a new entity. §5.1 already has `release_group → release → copy → capture`, and a more-upstream source object is cleanly **a different** `copy` under the same `release_group`: a different physical item you hold, with its own captures. What §5.6 adds is one field on `copy` — its generation in the lineage:

```
{
  "copy_id": "uuid",
  "release_group_id": "uuid",
  "source_generation": "production_master_tape",  // multitrack_reel | production_master_tape |
                                                   // lacquer | stamper | commercial_pressing | unknow
  "generation_confidence": "user_confirmed",      // or llm_suspected_from_provenance, etc.
  "condition": "EX",
  "acquired_at": "2026-05-30",
  "notes": "1/4-inch 15 ips, ex-studio; Dolby A encoded",
  "preferred": true,
  "preferred_scope": "release_group"
}
```

The payoff is that the **`_preferred` mechanism (§5.2, §12.3) becomes lineage-aware.** When a `release_group` holds captures at several generations, "the best source" is no longer just "the cleanest pressing" — it is, other things equal, the capture nearest the original. A transfer of the production master should out-prefer a transfer of a commercial pressing of the same recording, and the system can propose exactly that ("you hold a production-master-tape capture and three pressing captures of *Kind of Blue*; prefer the master tape as the release-group default?") as an ordinary propose-once-freeze decision. Nothing is overwritten; the pressing captures remain, fully archived, available in the `all_copies` view for comparison and for the cases where a specific pressing is itself the object of interest (a particular mastering, a regional cut).

Two consequences are worth drawing out, because they make the master/render split do real work it could not do before:

- **The multitrack reel is a master whose channels are *session tracks*, not a listening layout.** This is a different axis from the surround handling of §7.3/§12: quad and 5.1 are *playback* channel layouts, whereas a board reel's "channels" are drums, bass, vocal — the raw constituents of the recording, not anything you would route to speakers. So a multitrack master must **not** be shoved into `/multichannel/` (§12.3); it belongs in the source-only `/sessions/` namespace (§12.5), and any **mixdown of it is a render** — which is exactly where the master/render line wants to fall. The reel is the master; a stereo or surround mix derived from it is a render of that master, `precious` if the mix took real work.
- **"My analog chain sounds better than the CD" becomes a precise statement, not a vibe.** An

> archivist who captures the production master tape and prefers its sound is not making an aesthetic claim the system has to argue with — they hold a master at a more fundamental generation, and the system records that fact and prefers it on the merits. Any further-processed version they favor (a particular decode or restoration) is a `precious` render of that master, with the preference recorded as ordinary render metadata. Taste and provenance stop competing; the model represents both.

---

6. Capture: the station model

6.1 Machine → input → chain

A "station" in physical terms is a **machine that may have many capture inputs**. These are three nested concepts, and only the outermost is a deployed node:

- **Machine** — the physical box running `stationd`. Has a hostname and a role; gets deployed.
- **Input** — a capture endpoint on that machine: a specific ADC channel-set, a CD drive, a camera, a flatbed scanner, or just a watched import folder. A machine has *many*. Each has a type (vinyl/cassette/8track/cd/visual/…), a device binding (which PipeWire/ALSA device, which `/dev/srN`, which camera or scanner, or which folder), and its own default settings and calibration.
- **Chain** — the signal path of *one input* (cartridge → phono stage → ADC; or camera → lens → lighting). This is provenance, attached to the input and stamped onto every capture from it.

```
# illustrative stationd settings for one machine
inputs = {
  "main-vinyl" = {
    type = "vinyl";
    device = "pipewire:RME-ADI-2-ch1-2";
    chain = [ "soundsmith-sg-210" "soundsmith-preamp" "rme-adi-2" ];
    masterRate = 88200;
    cd4Rate = 88200;          # empirically calibrated for THIS cartridge
  };
  "second-table" = {
    type = "vinyl";
    device = "pipewire:RME-ADI-2-ch3-4";
    chain = [ "ortofon-2m" "schiit-mani" "rme-adi-2" ];
    masterRate = 96000;
  };
  "cd-drive" = { type = "cd"; device = "/dev/sr0"; };
  "visual-copystand" = {              # best-case reference rig (§8)
    type = "visual";
    device = "gphoto2:ceiling-canon";
    chain = [ "canon-eos-r5" "rf-100mm-macro" "glass-platen" "dual-45-light" ];
  };
  "visual-flatbed" = {                # a scanner is just another visual input
    type = "visual";
    device = "sane:epson-v600";
    chain = [ "epson-v600" ];         # modest gear, still recorded honestly
    dpi = 1200;
  };
  "visual-import" = {                 # phone photos / loose files dropped in
```

```
    type = "visual";
    device = "folder:/import/visual";
    # no fixed chain: provenance is read per-file from EXIF where present,
    # else recorded as "unknown handheld" — never simply blank.
    chainFromExif = true;
  };
};
```

Two inputs of the same type on one machine (the two turntables) have distinct identities, chains, and calibration; a capture always knows which input produced it, so provenance is correct per-capture. **All capture logic below — signal-gating, RAM buffering, the provisional state machine — runs per-input, not per-machine.** (The specific gear named above is illustrative; an input is defined by its *type*, *device binding*, and *chain*, whatever hardware fills those roles — a USB turntable into onboard audio is as valid an input as a strain gauge into a pro ADC.)

6.2 Always-listening capture with a RAM ring buffer

Analog inputs (vinyl/cassette/8-track/reel) are *always listening* when powered, so you never announce "I'm archiving." But the system must not wear out SSDs writing silence all day. The mechanism:

1. A continuous **RAM ring buffer** holds the last several seconds of each analog input's audio. While there is silence, nothing touches disk — the buffer just overwrites itself.
2. **Signal onset** (a level gate with hysteresis) starts writing to disk *including the buffered pre-roll*. This is the key trick: the pre-roll captures the very beginning — the needle drop, the tape leader, the first mechanism sound — that happened *before* the gate tripped. Nothing is lost at the front.
3. **Sustained silence** (past a hysteresis window, e.g. >4 s) stops disk writing and returns to RAM-only buffering, having captured the trailing run-out/eject through the window.

This is exactly how field recorders and broadcast loggers do pre-record buffering. SSDs never see idle silence; the full artifact including pre-signal mechanical sound is still captured.

6.3 The provisional → commit/discard state machine

Not every needle-drop is an archival intent. People play records for pleasure, and the system should let them enjoy a record *while* it happens to be archiving it — capturing opportunistically without forcing a "now I am archiving" mode. This matters all the more on a high-end chain where a play has real per-play cost: a Soundsmith strain-gauge cartridge, for instance, carries a meaningful per-side cost in stylus life (on the order of a dollar a side), so you don't want to burn a play on a throwaway capture, but you also shouldn't have to choose between listening and archiving. So capture is provisional until intent is established:

- Every signal-onset opens a **provisional** capture (it records — you can't retroactively capture what you

didn't, and storage is cheap).

- The system **infers intent from behavior**: a full side played through to run-out → propose *commit*; a partial play, needle bounced around, repeated restarts → propose *set aside / discard*; the same content captured again shortly after → treat as a redo and supersede the abortive one.
- It **asks, it never auto-destroys.** Abandoned-looking provisional captures go to a "needs triage" tray, surfaced as a light prompt ("3 provisional captures today look like casual plays — keep any?"). A default policy ("auto-discard provisional captures that never reached run-out after 7 days unless I say keep") handles the obvious junk. Nothing *committed and confirmed* is ever auto-discarded.
- A one-tap **"keep this"** on the station screen promotes a provisional capture immediately, skipping inference when you already know.

This directly serves the "real per-play cost, don't waste it, but don't make me babysit it" tension: listening for pleasure leaves a provisional capture that quietly ages out; an intentional archival play gets kept, with at most one tap.

6.4 Trim points and labeled regions

The master holds the full capture. Over it, the system stores **labeled regions** — proposed by detection at ingest, confirmed once, then frozen (principle 2.3). Rather than a single `music_in` / `music_out`, a master carries regions like `lead_in_noise`, `program`, `mechanism_sound`, `lead_out`, `locked_groove`, `oscillating_deadwax`.

A short glossary for the regions, since they come from vinyl playback physically: when you play a record, you lower the stylus (the "**needle-drop**") onto the smooth spiral **lead-in groove** at the outer edge, which carries no music — just the thunk of the drop and some surface rumble — before the music ("the **program**") begins. At the end, a **lead-out groove** (and on some records a locked or eccentric groove, §7.4) follows the music. The capture, recall (§6.2), starts a few seconds *before* the music thanks to the RAM pre-roll, so it includes that lead-in noise — which we want in the master but not in a clean render.

The detection target for the program start is deliberately chosen to be unambiguous: **`music_in` = a few milliseconds after the lead-in/transport noise ends**, not "where the music starts." Detecting the *end of needle-drop + lead-in groove rumble → silent groove* transition is a sharp, reliable event; "where does the music begin" is ambiguous for very quiet intros. The quiet intro of *Wish You Were Here* (which opens with extremely faint sound that builds for a long time) sits safely after the lead-in-noise boundary and is fully included — whereas a detector trying to find "the first music" might clip it. The rare oddball (lead-in bleeding straight into near-silent content) is flagged for a one-time human confirmation. This generalizes: every analog source has a "transport noise → content" transition (tape hiss/leader → program) that is the

real detectable boundary.

Because regions are metadata over an immutable master, a wrong marker is fixed by editing the marker and re-rendering; the audio behind it was never touched. **Renders select which regions to include** (§11–12): the default clean render is `program` only; an `-AMBIENT` variant includes `mechanism_sound` and `lead_in_noise`.

6.5 Repeat-play detection (hardened against false merges)

When a new capture arrives, it is fingerprinted (Chromaprint) and compared against recent captures and existing copies. A match suggests "another take of the same content." **This is dangerous if done naively** — two genuinely different records played back-to-back with no metadata must never be silently merged. So clustering is conservative and evidence-gated:

- Fingerprint matching identifies the *recording*, not the *copy*. A match does exactly one thing: **propose** a sibling link as a reviewable, non-destructive task. It never auto-merges or auto-discards.
- The default is always "new distinct capture." Every play creates its own copy+capture; clustering only ever *adds a proposed relationship* on top.
- A merge proposal requires **corroboration beyond the fingerprint**: same session within a time window, matching duration, matching track structure, or (once present) matching release identity. A bare fingerprint similarity with no corroboration is logged as a weak hint, never acted on.
- The threshold is high (favor precision): a false merge silently corrupts the archive, while a false split costs one tap to link later. The asymmetry is baked in.
- **Genuinely-identical content is a real phenomenon, not always an error to merge away.** Some records deliberately carry the *same program on both sides* (single-sided-equivalent pressings, certain promos, locked-groove or test pieces, some 78s and novelty cuts), and the fingerprint of side A will legitimately match side B. The clustering rules above already protect against the naive failure here — a fingerprint match only ever *proposes* a sibling link and never auto-merges — but this case deserves an explicit guard: when two sides of the *same physical copy* match, the default is to **keep both captures distinct** (they are different physical surfaces and may differ in wear, azimuth, or pressing detail) and, at most, propose a "these two sides appear to hold the same content" annotation for review, never a merge. Visual capture can corroborate intent here the same way it does for locked grooves (§7.4) and 8-track running order (§7.2): the label/j-card or matrix often states outright that both sides carry the same program, turning an ambiguous audio coincidence into a confirmable metadata fact. As ever, the expensive error is destroying a legitimately distinct capture, so the system errs toward keeping both.

The system can then *recommend* preferred-capture selection from the chain metadata (e.g. “two captures of side A — one from an auto-azimuth deck, one from a fixed-azimuth deck — pick a preferred”), and, where it can detect a known failure mode, pre-flag it (“the auto-azimuth capture shows a possible lock-on transient in the first 8 s; the fixed-azimuth one is clean there — that one pre-selected, confirm?”).

6.6 CD ripping: accuracy-first tiered

Why ripping a CD “correctly” is non-trivial. A CD is already digital, so unlike an analog transfer there’s no playback chain to worry about and track boundaries are exact (read from the disc’s table of contents / cue sheet) — no onset detection, no trim ambiguity. But a CD drive does *not* simply hand over perfect data: it reads at speed, can mis-read scratched or marginal discs, and different drives start reading at slightly different points (a fixed per-drive **read offset** of a few samples). Historically, **EAC** (Exact Audio Copy, a Windows tool) earned its reputation by re-reading uncertain sectors and, crucially, by checking the result against **AccurateRip** — an online database of checksums from many other people’s rips of the same disc. If your rip’s checksum matches the consensus, you have strong evidence it’s bit-perfect; that database check, not the drive’s own cleverness, is what proves correctness. A related database, **CUETools DB (CTDB)**, goes further: it stores error-correction data, so a slightly-damaged rip can sometimes be *repaired* rather than merely flagged as wrong.

A few drive features matter and are worth defining: **read offset** (the per-drive sample offset, looked up from AccurateRip’s drive database and compensated for); **Accurate Stream** (a drive property meaning it reads audio data positioned consistently — most modern drives have it); and **C2 error pointers** (a drive’s self-report of which samples it thinks it read badly — useful in theory, but most drives implement it poorly or fake it, so it’s better left off).

The goal is *verified bit-perfect extraction*, and because verification now lives in those databases rather than in the drive, the drive matters far less than in the EAC era. Guidance: any decent modern drive; set its read offset from the AccurateRip drive-offset database; enable Accurate Stream; **disable C2** (most implementations are unreliable or faked); trust database verification over hardware heroics; for a portable rig, a quality internal drive in a good enclosure beats a cheap external.

The pipeline is tiered, accuracy over speed:

1. **First pass:** `whipper` — the closest EAC-equivalent on Linux: cdparanoia-based reading with read-offset detection/compensation, cache-bypass over-reading (defeats the drive’s internal cache to force genuine re-reads), AccurateRip v1/v2 verification, cue generation, and EAC-style logs.
2. **Failed-track recovery:** `Rip Rip Hooray!` — a Rust ripper optimized for *recovery* of damaged discs, keeping each sample’s read-state and history across runs so problem regions can be re-ripped

repeatedly (even across different drives, since a different drive may succeed where one failed) without losing prior progress. Any track failing verification is handed to it for progressive multi-pass recovery.

3. **Verification gate: AccurateRip + CUETools (CTDB).** Adding CTDB covers whipper's blind spot (CTDB carries error-correction info, not just a pass/fail checksum) and provides *repair*, not only detection. The rip log + verification result are stored as a CAS sidecar; verification status feeds completeness and the copy-preference mechanism — an unverified rip stays flagged until a clean copy supersedes it. (For a rare disc not in any database, the fallback is to rip twice and confirm the two rips agree.)

`cyanrip` (fast, also AccurateRip-verifying) is an optional first-pass for the high-volume portable/friend's-wall case, but the fixed archive default is whipper for its audit-grade logs.

6.7 Non-intrusive capture on a high-end chain

§2.6 states the principle; this is how it is honored in practice. On a serious analog front end the playback chain is the point — it is what the owner spent the money for — and the archive must observe it without ever standing in its path. Two concrete commitments follow.

The capture tap is passive and out-of-path. Lacquer takes its signal from a point that does not alter what reaches the speakers — a dedicated ADC fed from a tape-out / record-out / a buffered split, or a second set of outputs — never by inserting an A/D/A round-trip into the listening signal. The always-listening RAM buffer (§6.2) and the provisional state machine (§6.3) already assume this: the system is a quiet observer of whatever is being played, adding nothing to the path and removing nothing from it. The reason this is a stated principle and not merely good manners is the per-play-cost case: a Soundsmith strain-gauge cartridge carries a real per-side cost in stylus life (on the order of a dollar a side, §6.3), so the worst possible design would be one that made you *play the record an extra time for the archive's benefit*. The capture must ride the play you were going to do anyway. The best capture and the best listen are the same event.

Calibration uses the LLM as a coach, not as a gate. The §7.5 per-rig calibration (sample rate for CD-4) and the §5.4 equipment chain together already capture *what* the rig is; the open opportunity on a high-end chain is helping the owner get the most out of it without demanding expertise. A calibration pass can play a known test record, *measure* the result with DSP (channel separation, subcarrier SNR, azimuth error, channel balance, speed/wow), and then have the LLM *explain* the numbers in plain language and suggest the physical adjustment ("left channel is ~1.2 dB hot and azimuth looks off — try a small anti-skate and azimuth tweak; here's what each does"). The DSP measures; the model translates measurement into

guidance. This stays firmly inside the existing rules: the model only ever *proposes* (§10), the measurements and any resulting settings are stored as ordinary input calibration (§7.5) and equipment-chain facts (§5.4), and nothing here is required — a user who ignores it loses only the coaching, never the capture. It is the §2.6 “use the best gear you can, but the system never assumes you have one” stance turned into active help for the people who *do* have the gear and want to wring the most from it.

Hardware as a possible premium SKU, not a committed box. Everything above is software riding whatever front end the user owns. There is a natural product thought here — a higher-end appliance SKU (§15.2) that ships with a known-good capture tap and a guided calibration routine baked into its hardware profile — but it is noted as a *possibility for the appliance line*, deliberately not specified as a required component of the system. The system’s floor remains an ordinary input into an ordinary interface (§2.6); the audiophile path raises the ceiling for those who want it.

---

7. Format-specific capture quirks

The messy reality of analog formats. Each of these folds into the same primitives (immutable master + labeled regions + region-aware renders + propose-once-freeze), which is why none required new mechanisms.

> **Background for the non-specialist.** The rest of this section assumes no prior knowledge of analog audio formats. Each subsection first explains *the physical reality* — what these media are, how they were recorded and played, and what goes wrong when you digitize them today — and then specifies what Lacquer does about it. If you are building this and have never spliced a tape or dropped a needle, read the prose, not just the bold sentences: the quirks are not arbitrary engineering choices, they are consequences of how these objects physically work. A few terms used throughout:
>
> - **Master vs. render** (recap from §2): the *master* is the raw, complete, untouched digital capture stored forever; a *render* is a processed output file derived from it. We never edit a master; we change which regions of it a render includes.
> - **Transfer / needle-drop / capture**: playing a physical record or tape in real time while recording its analog output to digital. Unlike ripping a CD (a data copy), an analog transfer happens at playback speed and is influenced by the entire playback chain (cartridge, deck, preamp, ADC).
> - **Sample rate / bit depth**: how finely the analog signal is digitized. This document writes them as *rate / bits* — so “96/24” means 96,000 samples per second (96 kHz) at 24 bits per sample, and “44.1/16” means 44.1 kHz at 16 bits (the CD standard). Higher numbers capture more, at the

cost of larger files. Human hearing tops out around 20 kHz, and the Nyquist theorem means a sample rate of *R* can represent frequencies up to *R/2* — so 96 kHz can represent up to 48 kHz, well beyond hearing, which matters for one specific format below (CD-4) that hides data above the audible range.

- **The LLM / vision model**: several quirks below are easier to resolve if the system can *read a photo of the packaging* or *reason about what a release probably is* — for example, recognizing a Dolby logo on a cassette's j-card or shell and inferring that a particular cassette issue was likely Dolby C. Lacquer can use an AI model for this, but it is strictly an **optional helper** that only ever *proposes* (you confirm), and the system works fully without it. §10 explains what these models are and why they fit an archival system; for now, "the LLM reads the label" just means "an optional assistant looks at the photo and suggests an answer for you to approve."

7.1 Cassette: flat-master-always, Dolby as derivative

What a cassette is, and why Dolby is a problem. A compact cassette stores audio as magnetic patterns on a thin tape. Tape has an inherent high-frequency hiss. To fight it, most pre-recorded and many home tapes were recorded using **Dolby noise reduction** — a *companding* system: during recording, quiet high-frequency sounds are boosted by a known curve; during playback, a Dolby decoder applies the exact inverse curve, pushing the hiss down along with the boost. The catch is that decoding only undoes the encoding correctly if it happens at the **exact reference level** the encoding assumed. If the playback level is even slightly off — which is common, because old decks drift out of calibration and tapes age — the decode curve doesn't line up, and the result is audibly wrong: typically a dull, **muffled** sound, because the decoder pulls down treble that should have stayed up. There were several Dolby noise-reduction variants over the decades, and the ones relevant to *cassettes* are a specific family: **Dolby B** (the ubiquitous consumer standard), **Dolby C** (stronger, less common), and **Dolby S** (strongest, rare, mostly early-to-mid-1990s). (A separate, earlier system — **Dolby A** — was a professional studio NR format used on master tapes, not on consumer cassettes; it can appear upstream in a recording's lineage, e.g. on a production master tape (§5.6), but it is not what a pre-recorded or home cassette carries.) Each cassette variant needs its own matching decoder.

Why we capture flat. Given the above, decoding Dolby in hardware *at capture time* bakes a possibly-wrong decode permanently into the master — and you can't undo it later. So Lacquer **always captures flat (Dolby decode OFF), at 96 kHz / 24-bit (written 96/24), as the master**, even when Dolby-on sounds better through the deck. (That rate is generous headroom for tape, whose usable bandwidth sits well under 20 kHz — see §7.6 for why each source gets a different rate.) Modern software decoders can *normalize to the correct reference level before decoding*, so feeding them a clean flat transfer often

produces a better result than the deck's hardware decode did — and, crucially, you can re-decode the same flat master later with a better decoder, or as the correct variant once you know which it is. The flat master is archival truth; every decoded version is a regenerable derivative.

This is the textbook case for the `canonical` **render flag** (§5.5): the flat master is the thing you *preserve*, but it is deliberately **not** the thing a player should reach for by default — the *decoded* render is. So a Dolby tape's flat master carries an unfilled-or-filled `canonical` slot pointing at its correct decode: once the right decoder runs (auto or manual), that decoded render is marked `canonical` and the VFS serves it by default, while the flat master stays in the archive untouched. A manually-produced decode (below) is additionally marked `precious`, since it cost real hand-work through a tool the pipeline cannot rerun. The flat-vs-decoded distinction the rest of this section describes is exactly the master-vs-canonical-render distinction in §5.5, applied to tape.

Deck provenance is mandatory and inseparable from the decode, because *which deck and how it was set up* materially changes the captured sound. A second reason, beyond Dolby: **azimuth.** Azimuth is the angle of the playback head relative to the tape; if it's misaligned (the recording deck and your playback deck disagree, or a head has drifted), high frequencies are lost and the stereo image smears. The system models decks by their relevant *capabilities* — azimuth handling, NR support, head/transport quality — rather than assuming any particular make, and records the specific deck per capture from the equipment registry. One capability distinction matters enough to model explicitly: **automatic vs. fixed azimuth.**

- **Auto-azimuth decks** (the classic example being the Nakamichi Dragon, with its NAAC system — Nakamichi Auto Azimuth Correction) continuously sense and correct head azimuth as the tape plays — superb on tapes with drifting or mis-recorded azimuth, but the correction needs a moment to find alignment, so it can produce a brief **lock-on artifact** (a wobble or smear) at the very start of a side while it hunts.
- **Fixed-azimuth decks** (the large majority, including excellent ones like the Nakamichi CR-7A and any ordinary consumer deck) hold a manually-set azimuth — no lock-on artifact, but no help on an azimuth-mismatched tape (you'd lose treble on a tape recorded on a misaligned machine).

Neither class is strictly better, so **where a user has more than one deck, a side is often worth capturing on each** as sibling captures under one logical side, each tagged with its `chain.deck`, with `preferred_for_side` chosen after auditioning — their failure modes differ, so the best take depends on the tape. The system records the deck in metadata and, eventually, in tags.

This is a spectrum, not a requirement. A reference-grade three-head deck (a Dragon, a CR-7A, a Tascam/Studer transport) is the best-case input and yields the cleanest masters — and at multiple thousands of USD each, well out of most people's reach — but the system is designed to capture from

whatever a user owns, down to an ordinary single deck. The equipment chain records honestly what was used, so provenance always reflects the real quality of the transfer. Better gear yields better masters; modest gear still yields a valid, well-documented archive entry. *The guidance is simply: use the best transport you can get hold of for anything you care about — but the system never assumes you have one.*

Dolby type is inferred from the physical artifact, turning an otherwise unsolvable signal problem (you generally cannot reliably tell *from the audio alone* which Dolby variant was used) into the metadata problem the visual-capture stage already solves — by looking at the tape and its packaging:

- **Dolby S tapes are physically marked** with a distinctive double-D "S" logo on the shell and usually the j-card → the vision model reads it → `dolby_type: "S"`. There is **no software S decoder anywhere** (FOSS or proprietary), so S tapes are captured flat and flagged `needs_manual_pass` indefinitely, awaiting a future decoder applied to the preserved flat master.
- **Unmarked + older than a cutoff → almost certainly B** (Dolby B was the consumer default for decades, and absence of a C/S marking on an older tape is itself evidence). Inferred `B`, routed to the auto-decode path. *This is a default assumption, not a certainty* — home-recorded tapes were often left unmarked regardless of what NR was used — so the muffled-ness heuristic (below) backstops it.
- **Dolby C** is comparatively rare on pre-recorded tapes and clusters in specific contexts (Nakamichi reference/demonstration tapes, certain CBS/Sony classical issues, audiophile labels). The LLM cross-references the release against known-C contexts and produces a *ranked suspicion*, surfaced as a one-tap confirm: "This looks like it might be a Dolby C tape (CBS Mastersound classical, 1982) — confirm?" *(Research note, unconfirmed: it is worth checking whether Dolby C was also common on promo/demo cassettes circulated to labels, radio, and press, which would add another known-C context to weight. Flagged for domain research, not assumed.)*

`dolby_type` carries its own provenance/confidence: `marked_on_shell` (certain), `inferred_from_date` (probable B), `llm_suspected_from_release` (needs confirm), `user_confirmed`.

Decode engines: the open-source Pascal decoder (B solid, C possibly incomplete) is the Nix-packaged in-pipeline default for auto-decode. DDi Codec is the quality benchmark but is a proprietary, Store-DRM'd, GUI-only app with no headless/CLI mode and no Linux build — *not viable as an automated engine* (the Wine-headless-in-a-Nix-stack path is a fragile dead end). So high-quality decode of a difficult tape is a **manual pass** (run DDi by hand on a workstation, re-ingest the result as a `precious` + `canonical` render with `dolby_decode: "software-DDi-manual"`) until a FOSS replacement exists. The `precious` flag is what guarantees that hand-work, which the pipeline cannot reproduce, is backed up like a master (§5.5).

Muffled-ness heuristic (the backstop): since mistracking manifests as treble loss, compare the high-

frequency energy of a decoded candidate against the flat master; if the "decoded" version is suspiciously dull, flag `needs_manual_pass` regardless of the type inference. This catches wrong B-assumptions and bad hardware decodes — it is a cheap automated check that the decode *probably* went wrong, not a guarantee, so it raises a review flag rather than discarding anything.

`dolby_decode_status`: `not_needed` | `auto_applied` (+confidence) | `needs_manual_pass` | `manual_done`. If decoding isn't solved well by production, affected tapes simply carry `needs_manual_pass` and the reminder engine tracks them — the flat master guarantees nothing is lost.

7.2 8-track: dual-seam loop correction

What an 8-track cartridge is. The Stereo 8 cartridge (1960s–70s, mostly automotive) holds a single **continuous loop** of tape — there is no rewind; the tape is pulled from the center of the spool and wraps back onto the outside, running forever in one direction. The tape width is divided into **four stereo "programs"** (8 tracks = 4 programs × 2 channels). The player reads one program at a time; to switch programs it physically **shifts the playback head** up or down. That shift is triggered when a metallic **foil splice** — the single join that makes the loop continuous — passes a sensor once per lap. The head shift is mechanical and abrupt, producing the format's infamous audible **"ka-chunk"** at each program change. Two consequences make digitizing an 8-track unlike any other format, and both must be handled:

Problem 1 — the loop seam ("start anywhere, wait for it to come around"). Because there's no rewind and no defined start, you simply start recording wherever the tape happens to be and let it play until it returns to where you began — one full lap, plus a little overlap so the end re-records material you already captured at the start. Now you have a recording whose end overlaps its beginning, and you need to join them into one clean loop with no audible seam. Lacquer does this with fingerprint **autocorrelation**: it slides the overlapping tail against the head to find the exact point where the same audio recurs (the lap point). It then places the join, in priority order: (1) **inside an inter-track silence** within the overlap region if one exists — joining during silence means there's no musical content to align, the safest case; (2) otherwise an **equal-power crossfade at the matched sample** — and because the two pieces are recordings of the *identical* moment of the same tape, they are phase-coherent, so the crossfade is inaudible. Either way: no pops, clicks, or ticks at the seam.

Problem 2 — the program-change ka-chunk. Detected as a sharp broadband transient plus a level discontinuity. The four programs are kept as separate `logical_program`s, **not** spliced into one continuous thing (they really are four distinct ~15-minute segments), and each program is independently loop-corrected per Problem 1.

Track order comes from the cartridge photo, not assumed. Here is a quirk that surprises people: an

album's track order on 8-track is **frequently different** from the LP or CD. Because the four programs each had to be roughly equal in length (~15 minutes) to keep the head-shifts evenly spaced, labels **resequenced the album** to balance the programs — and sometimes **split a single song across a program boundary**, fading it out at the end of one program and resuming it at the start of the next. So the system cannot assume the canonical album order. The visual capture → LLM reads the cartridge label to get the authoritative program/track order and to detect the song-split-across-programs cases; fingerprint/transient detection finds the *physical* boundaries; agreement → high confidence, disagreement → review task. The split-song case is reassembled by metadata (the two fragments are marked as one logical track). This is also why the "capture the cartridge label *now*" active prompt (§14) exists — the cartridge in your hand is the only authority on its own running order.

The 8-track ambience contradiction, resolved. Loop-correction produces a *clean continuous* program — which by definition means smoothing away the ka-chunks and mechanical noises. But some users *want* those sounds preserved: the ka-chunk, the eject, the mechanism, are part of the authentic 8-track experience and a piece of cultural-artifact history. These are opposite goals, so the system does both as separate region-aware renders over one untouched master: a **clean-seamed render** (default listening) and an `-AMBIENT` **render** that keeps the ka-chunks, eject, and mechanism sounds. The raw continuous master — ka-chunks and all — is never modified, so neither render is ever destroyed by the other.

7.3 Quad / multichannel

What quadraphonic is, and why it's a thicket. In the early 1970s, before modern surround sound, the industry attempted **four-channel "quadraphonic"** audio — front-left, front-right, back-left, back-right. Several *incompatible* systems competed and none won, which is why this is messy. The crucial split for digitizing is how the four channels were physically stored on a two-channel medium (a stereo record groove or stereo tape):

- **Matrix systems (SQ, QS)** mathematically *fold* four channels into a normal two-channel signal using phase relationships, to be *unfolded* by a decoder on playback. This is lossy — the four recovered channels aren't perfectly separated — but the encoded record plays acceptably as ordinary stereo on non-quad equipment. **SQ** (Columbia/CBS) and **QS** (Sansui, "Regular Matrix") are the two common ones. For digitizing: you **capture the normal 2 channels and decode to 4 in software.**
- **Discrete-via-subcarrier (CD-4 / Quadradisc)** keeps all four channels genuinely separate by carrying the front channels in the normal audible groove signal and the *difference* information needed to recover the rear channels on an ultrasonic **~30 kHz subcarrier** cut into the same groove — above human hearing, recovered by a demodulator. This is why CD-4 needed special cartridges and styli that could read 30+ kHz. For digitizing: you still **capture 2 channels, but at a high enough sample**

rate to record the 30 kHz subcarrier, then demodulate in software. By Nyquist, capturing 30 kHz requires a sample rate above 60 kHz; in practice 96 kHz is comfortable and 88.2 kHz is borderline-adequate (hence the per-rig calibration in §7.5).

- **Discrete on tape (Quad 8-track, "Q8")** simply records four genuinely separate channels on the tape — no encoding, no decoding. For digitizing: you **capture 4 real channels.** This is the *only* source in the entire system whose master is natively more than 2 channels.

Specifics:

- **SQ and QS (matrix)** — capture **2 channels at 96/24**, decode to 4 in software. **QS has a FOSS decoder** (`quarkquad/qs` , a JUCE-based multiband decoder modeled on the Sansui Vario-Matrix hardware) — needs packaging into a headless CLI worker (tally item). **SQ has no mature FOSS decoder** — good SQ decoding is proprietary (Pspatial Stereo Lab) or was historically done with manual scripts; routed to `needs_manual_pass` .
- **CD-4 / Quadradisc** — capture **2 channels at a high rate** to get the subcarrier. Software demodulation exists essentially only in proprietary Stereo Lab → `needs_manual_pass` . The needed sample rate is **empirically calibrated per rig** (§7.5), not assumed: a given cartridge may resolve the subcarrier well enough at 88.2 kHz, or may want 192 kHz — measure, don't guess.
- **Quad 8-track (Q8)** — genuinely **discrete 4-channel on tape**; capture 4 real channels, no decode needed. The only native multichannel master in the system.

A note that unifies these with §7.1: wherever the only available decoder is **proprietary and has no headless mode** (SQ via Stereo Lab, CD-4 demodulation via Stereo Lab), placing it "in the chain" means running it *by hand* on a workstation and re-ingesting the result. Such a decode is therefore in exactly the same category as a hand-made Dolby decode or a manual restoration (§5.5): it is a **render** of the captured master, but one the pipeline cannot reproduce, so it is marked `precious` (backed up like a master) and, where it is the version a listener should hear, `canonical` . The captured 2-channel master is preserved untouched, and the manual decode is treated as the precious work it is — until a FOSS decoder exists and the same decode becomes a cheap, regenerable (non-precious) render.

`quad_encoding` : `SQ | QS | CD-4 | discrete-8track | none` , detected by the LLM from cover/label markings (records and sleeves were typically marked "Quadradisc," "SQ," "QS," etc.) plus, for CD-4, automatically detecting the presence of the ~30 kHz subcarrier in a high-rate test capture. Vinyl quad keeps a **2-channel master** (that *is* what's physically in the groove); the 4-channel version is a derivative. Q8 keeps a 4-channel master.

Output handling (see §12 for the namespace): a quad release appears as a **labeled 2-channel fold-down** in

the universal stereo tree (`[Quad Fold-Down]` , playable everywhere, ancient-client-safe) *and* a discrete/decoded 4-channel version in the parallel `/multichannel/` namespace. SQ's fold-down can be the stereo master itself (an SQ record is designed to be stereo-compatible); CD-4 needs an explicit fold-down derivative because its raw 2-channel capture contains the ultrasonic subcarrier and isn't meant to be heard directly.

7.4 Locked and eccentric grooves

What these are. A normal record groove spirals inward continuously from edge to center. Two deliberate exceptions exist and both break naive "the side is over when the music stops" logic:

- **Locked groove** — a perfectly **concentric** (circular, non-spiraling) groove, so the stylus loops the same ~1.8-second segment (one revolution at 33⅓ rpm) **forever** instead of running into the lead-out. The famous example is the inner-groove loop at the end of the Beatles' *Sgt. Pepper's Lonely Hearts Club Band*. Detection is **multi-signal and proposal-only** (principle 2.3), because the obvious single signal is a trap: plenty of records are *deliberately* the same short passage repeated for a whole side (minimal/drone/loop pieces), and some even *sample or simulate* a locked groove — all acoustically periodic, none mechanical. So Lacquer weighs several independent pieces of evidence:
- **Periodicity** (necessary, never sufficient) — the same short segment repeating. Measured two independent ways: raw audio self-similarity *and* acoustic-fingerprint self-similarity (the Chromaprint machinery turned inward — not a database lookup, just "does this side's fingerprint repeat against itself?", which needs no external catalogue and so works on the obscure/self-pressed long tail). Two agreeing periodicity measures are stronger than one, since they fail differently (the fingerprint is more robust to surface noise and wow/flutter). But periodicity *alone* cannot tell a mechanical loop from an intentional musical one — both repeat.
- **Seam pop** (the mechanical tell) — the brief click/discontinuity where the groove jumps back across itself at the loop seam, recurring exactly once per revolution. This is the one *audio* signal that speaks to physical mechanism rather than content. It is subtler on some pressings (confidence-weighting evidence, not a hard gate), and note it is *still* spoofable: a record that samples a locked groove reproduces the pop faithfully too.
- **Tonearm fixed-radius** (§9 video, when present) — the only signal that cleanly separates a *real* mechanical locked groove from a perfect audio *simulation* of one. A genuine locked groove halts the arm's inward travel (it holds a fixed radius indefinitely); a sampled/simulated loop keeps spiraling because it is just audio on an ordinary cut. Every audio-domain signal above is reproducible by a recording *of* a locked groove; only the physical groove behaving like one can be witnessed here — exactly the "video as sensor, not just content" case §9 is built for.

This also works wherever the locked groove physically sits, even mid-side (the rare case where it isn't at the very end, which would otherwise fool a tonearm-position end-detector). Labeled `locked_groove`. **When the signals agree** (periodic + seam pop + fixed-radius arm), confidence is high and the default policy applies: **fade out after N loops**; options: keep M loops then hard-stop, preserve the loop as its own region for the curious, or flag for review. **When they disagree** (periodic but no pop, or periodic with a pop but the arm keeps creeping inward — i.e. a likely sampled/simulated loop, or intentional loop music), Lacquer does *not* fade or truncate anything; it **preserves the audio intact and raises a review task**, because the expensive error is cutting off music the artist intended, not leaving a few extra mechanical revolutions. One case is irreducibly human: an artist who cuts a *real* locked groove *as the piece* is mechanically identical to a runout locked groove — only intent differs — so the ambiguous case correctly ends at a human decision, not a cleverer heuristic. Per-release override, global default. - **Eccentric / oscillating deadwax** — a run-out groove cut **off-center** so the tonearm visibly swings side to side once per revolution (common on some 78 rpm records and novelty pressings; sometimes used as a deliberate end-of-side signal). Detected as **periodic visual tonearm motion** in the performance video (the arm rocking back and forth at the revolution rate) and corroborated by audio "wow" (cyclic pitch variation) at the same rate. Labeled `oscillating_deadwax`, same fade-after-N default. This is a case that is genuinely *easier to detect by watching the arm than by listening* — see §9, where the performance video acts as a sensor.

7.5 Per-rig capture-rate calibration

Rather than hardcoding sample rates per format, the right rate for an ambiguous case (notably CD-4's ~30 kHz subcarrier, §7.3) is **measured once per rig and stored as input config** — because whether a given cartridge/stylus/ADC chain actually resolves the subcarrier cleanly at 88.2 kHz versus needing 96 or 192 kHz depends on the specific hardware, not on theory. A calibration protocol captures a known reference (e.g. a CD-4 test disc) at several rates, runs the relevant decode/demodulation on each, and measures quality (channel separation, subcarrier signal-to-noise, demodulation artifacts). The best-but-not-wasteful rate is recorded in the input's settings (`cd4Rate = …`), to be re-run only if the cartridge or ADC changes. Same philosophy as AccurateRip drive-offset calibration: measure your actual hardware once, store the result, stop thinking about it. For the appliance (§15), this becomes a *factory* step baked into the hardware profile, so the buyer never calibrates anything.

7.6 Source-appropriate sample rates (auto-selected)

Laziness here means "never make me choose," not "max everything" — recording at a higher rate than the source contains just stores larger files of the same information. The station picks the rate from the source:

- **CD → 44.1/16** (a CD *is* already 44.1 kHz / 16-bit digital — it's a data copy, not an analog transfer, so

the bit-perfect rip at native rate is the true master; upsampling it would only store bigger zeros).

- **Cassette / 8-track → 96/24** (tape's usable bandwidth is well under 20 kHz; 96 kHz is already generous headroom).
- **Vinyl (normal) → 88.2 or 96/24** — enough to faithfully capture what a good cartridge and stylus actually pull out of the groove, and *not* casually dismissible as overkill: a high-resolution analog front end (a strain-gauge cartridge into a clean phono stage and ADC, the §2.6 quality-ceiling case) resolves real low-level detail and air that this rate preserves and a lower one would not. The rate matches the chain's capability rather than a folk belief about what vinyl "deserves."
- **CD-4 vinyl → the calibrated high rate** (§7.5), the one case needing ≥96 kHz for the subcarrier.

The principle is *match the rate to what the source and chain actually contain*, not "max everything." Blindly forcing 192 kHz on a CD rip or a cassette transfer roughly doubles storage and encode load while capturing only larger copies of the same information — there is nothing above ~22 kHz on the tape or in the CD data to record. (Where a source and chain genuinely do carry more — CD-4's ultrasonic subcarrier, or a top-tier vinyl front end — the higher rate is selected precisely because it is *not* wasted.)

---

8. Visual capture

Records have a visual side as much as an audible one — covers, j-cards, booklets, labels, the media itself, and the runout/matrix etchings — and Lacquer captures it and feeds it to enrichment. The architecture here mirrors the audio side exactly, including the two attitudes toward equipment that the audio side holds *together*: **the system makes the best of whatever visual input it is given, and it records exactly what produced each input.** Those are not in tension — they are the two halves of the same stance (§2.6, §5.4). Input-agnosticism is about what you may *feed* the system: a copy-stand camera, a flatbed scanner, or a handful of phone photos taken on the couch — Lacquer does not assume which, and is not built around any one of them. Provenance is about what the system *remembers* about that input: which device, which lens or scanner, what lighting, captured how. Its job is to get *as much real information as possible into the visual master* — even a partly-glared phone snapshot is worth keeping in full — *and* to stamp that master with the chain that made it, so later you can reason honestly about quality ("the glare on these is because they're handheld phone shots; the platen RAWs are clean") and improve deliberately.

That rests on the same master/derivative split as audio: **the input image, in the highest fidelity it arrived in (RAW where available, the original file otherwise), is an immutable master in CAS; every processed image — dewarped, cropped, stitched, colour-normalized — is a regenerable derivative.**

Improve the dewarp model next year and re-run it on the originals; never re-photograph. Better input raises the ceiling on what can be extracted; it never raises the *floor* of what the system accepts. A phone photo simply yields a lower-confidence master that lands more of its fields in the review queue — it is never rejected, and the §8.2 pipeline degrades gracefully toward "keep the original, flag for a human" rather than failing. And like the audio chain, the visual chain is **intrinsic, queryable provenance** (§5.4): every visual master carries the equipment that produced it, so "show me everything shot on the copy-stand" or "re-capture the phone-photo'd sleeves now that I have the rig" are first-class queries, not guesses.

> **Best-case reference rig.** The highest-quality setup below — a tethered copy-stand camera with controlled lighting — is described in full as a *reference* and as the kind of rig Lacquer might one day ship as a turnkey product, **not** as a requirement. Read §8.1 as "here is the ceiling, and here is what each piece of it buys you," then note that every stage of the pipeline also has a path for inputs that have none of it.

8.1 Reference rig (the quality ceiling) and lower-effort inputs

The best-case capture path is a ceiling-mounted, `gphoto2`-tethered camera over a glass platen (copy-stand geometry), pulling RAW directly (no card shuffling). Two diffused lights at ~45° kill glare on glossy laminated sleeves — the single biggest quality factor, because no algorithm recovers a blown-out specular highlight. The platen flattens warped sleeves and lets a gatefold be pressed flat one panel at a time. This rig defines three capture modes:

- **Document mode** — flat-ish items on the platen: sleeves, inserts, j-cards, booklets, flat labels.
- **Object mode** — 3D things on a neutral mat, platen removed: the cassette shell, the record itself, a boxset spine.
- **Matrix mode** — raking-light burst for runout etchings.

Lower-effort inputs map onto the same modes, and each still carries a chain. A flatbed scanner is an excellent document-mode source (flat, evenly lit, high-resolution) and often *better* than a camera for flat items — it just can't do object or raking-light modes. A phone camera covers all three modes at lower and more variable quality: handheld glare and perspective instead of controlled lighting and a fixed platen. The system takes what each provides — it reads EXIF/RAW where present and accepts ordinary JPEGs where not — and, crucially, records the equipment chain for *every* visual input, exactly as the audio side does (§5.4). The copy-stand input names its camera, lens, platen, and lighting; the flatbed names the scanner model and DPI; even a phone/import path records what it can (device and EXIF where the file carries it, an explicit "unknown handheld" where it doesn't). This provenance does double duty: it calibrates

downstream confidence (a controlled platen RAW is trusted further than a handheld snapshot) *and* it is durable, queryable archival truth about how each image was made — the same record that lets you ask "which sleeves still need a proper re-shoot?" or judge why one capture looks better than another. Capture can be live-tethered or simply a folder of images dropped in after the fact — an import is just another input, chain and all.

Triggers, all lazy and all optional depending on the rig: a foot pedal (`/dev/input`), motion-settle auto-fire (frame changes then stabilizes → shoot), the `lq` /web "capture" button, or — for the phone/import path — nothing at all beyond handing the system the files. Each shot gets a provisional role guess (`cover_front` , `gatefold_L/C/R` , `jcard` , `label_A/B` , `matrix_A/B` , `object`) correctable later.

8.2 Processing pipeline (per image or burst)

1. **Isolation.** Document mode: grayscale → blur → Otsu threshold → `findContours` → largest quadrilateral (`approxPolyDP`). Object mode: a segmentation model (`rembg` /U²-Net, or SAM for precise masks) to cut a 3D object from the mat.
2. **Straighten + trim** (one step, both jobs). A **4-point perspective warp** (`getPerspectiveTransform` + `warpPerspective`) maps the detected quad to a clean rectangle — dewarped *and* cropped to the item edges. Rotate-only fallback (`deskew` / `minAreaRect` median angle) when there's no good quad.
3. **Dewarp curvature** (document mode, confidence-gated). For items that won't lie flat — a curled j-card, a label following a cassette's curve: `page_dewarp` (cubic-sheet model, CPU-only) by default; a learned dewarper (YOLOv8 detection + cubic-polynomial restoration, CPU-runnable under ~2 GB) for nasty cases. Low confidence → keep the perspective-corrected version, file a review task.
4. **Auto-merge wide items.** Gatefolds and extra-long j-cards shot as overlapping frames → OpenCV `Stitcher` (ORB/SIFT features, homography, blending). Trigger: same role tag + temporal grouping. Low stitch confidence → keep panels separate, raise a review task. The original input frames (RAW where available) are always retained.
5. **Normalize** (gentle color/exposure), write the processed derivative.
6. **Hand to enrichment** (§10): OCR + vision-LLM extraction with per-field provenance.

Object-type classification (`vinyl_disc` vs `cassette_shell` vs `cover_art` vs `jcard` vs `label`) routes downstream behavior and sets the **canonical aspect ratio** — see §8.4.

8.3 Matrix / runout capture (vinyl)

What the runout is and why it's worth the trouble. Between the end of the music and the label, every record has a smooth ring called the **run-out groove** or **deadwax**. Scratched into that area — by hand at

the mastering/pressing stage — are the **matrix (or "matrix/runout") numbers**: catalog codes, stamper identifiers, mastering-engineer signatures, and pressing-plant marks. These are often the single most precise way to identify *exactly which pressing* a record is (two pressings that look identical on the label can be told apart by their matrix codes), which is why collectors and databases like Discogs record them meticulously. The problem: they're shallow scratches in shiny (usually black) vinyl, so under normal flat lighting they're nearly invisible.

The fix is **raking light** — a single light source placed low and to the side, so it grazes across the surface and throws the scratched characters into relief (the same reason low evening sun makes footprints in sand visible). This is the one capture task that genuinely *depends* on the rig, and it degrades along a clear ladder. With **computer-controlled lighting** (relay/DMX/addressable fixtures the station can switch), the matrix sub-pipeline becomes lazy and far better:

- The station kills the main diffused lights, fires a dedicated **low-angle side light** across the runout, shoots, restores normal lighting — automatically, no human relighting.
- It **sweeps**: several light angles / small platter rotations per burst, because any single rake angle tends to catch some characters legibly and lose others to glare or shadow.
- Combine: **exposure-fusion / focus-stack** the burst (merge the best-lit, sharpest region from each frame), or **best-frame selection** by local contrast in the runout band.
- **LLM-in-the-loop reshoot:** the vision model reads the candidate; if confidence is low ("characters lost to glare"), it requests another shot at a different angle (bounded retries), then falls back to a review task. The model controls the rig to get its own better input.

Without controlled lighting, the system asks rather than fakes. A flatbed scanner can't rake at all; a phone *can* — a desk lamp held low and to the side is exactly raking light, just manual. So where the rig can't sweep automatically, Lacquer turns the matrix into a guided **review task**: it tells the user what to do ("shine a light low across the deadwax and take a few photos at different angles"), accepts whatever frames come back, runs the same best-frame/fusion and LLM-read steps on them, and leaves the task open if the result is still unreadable. The matrix is high-value but never mandatory — an unread runout simply leaves the pressing identified at a coarser grain, flagged for later, never blocking anything.

Matrix strings cross-reference against MusicBrainz/Discogs — a matrix/catalog number is often a precise release identifier, which is the whole point of the effort. When a pressing isn't in those databases, the system can prepare a **submission** back to them (it has the structured tracklist, durations, catalog #, matrix, scans) as a review task — contributing to the commons. Per the local-first law, everything submitted is stored locally first as canonical; the submission is a downstream export, and a local snapshot/mirror of depended-on external records is kept so their disappearance loses us nothing.

8.4 Aspect ratio and object type

Forcing everything square is wrong. Each asset role carries a **canonical aspect**: `cover` = square, `jcard_front` = portrait rectangle, `cassette_shell` = landscape rectangle, etc. The detected object type sets it; embedded tag art respects it. ID3 `APIC` and Vorbis `METADATA_BLOCK_PICTURE` have **no** aspect constraint — the "everything square" pressure comes from player UIs, not the formats — so the correctly-proportioned rectangular j-card is embedded and players letterbox it. The system does **not** crop-to-square.

Object-type detection is not cosmetic: a `vinyl_disc` photo triggers "is the matrix readable here? if not, queue a matrix capture"; a `cassette_shell` triggers "read shell markings for Dolby type and tape brand"; and neither is mistaken for `cover_art`.

---

9. Performance video

An **optional, mute video of the medium playing** — the record spinning, the cassette reels turning, the disc sliding in. It exists purely for your later use (YouTube material, ambience) and is **entirely out of band**: no tags, no audio metadata, nothing in the playback/ID3 path. Stored in CAS like any other asset, immutable, attached to the `copy`.

It is captured with **no audio track** by design — the audio truth is the dedicated capture chain, and a camera-mic'd track would only be an inferior duplicate. Provenance via the equipment chain (`{role: "video_camera", ref: …}`) plus what it shows (`platter` / `reels` / `full_deck`).

9.1 Video doubles as a sensor

Where a camera is present, the video is a genuine **sensor feed** to the capture state machine, not just a keepsake. Detected events sharpen capture decisions:

- **End-of-side** corroboration — tonearm reaching the run-out, lifting, or the record being removed. The audio (lead-out groove noise) is primary; the video resolves ambiguous cases ("was that silence the end, or a quiet passage?" — the tonearm's position says definitively). "Tonearm reached run-out" is also a strong "full intentional play → propose commit" signal for §6.3. *Exception:* an inner mid-side locked groove defeats the *video* end-signal (the arm isn't visibly "done") — but the audio periodicity + seam-pop signals (§7.4) flag it regardless, and the disagreement between "audio says looping" and "arm isn't at the run-out" is itself useful evidence routed to the §7.4 multi-signal proposal.
- **Locked/eccentric groove** — the oscillating-deadwax case (§7.4) is detected as periodic visual tonearm motion, easier to see than to hear.

- **Auto-azimuth lock-on** corroboration on cassette decks that have it (§7.1).

Stations without a camera rely on audio alone; the video is always optional and only ever sharpens things.

9.2 Audio/video sync — different timelines, one frozen offset

The ideal video starts *before* the audio program (needle descending, disc inserting); the audio render must start *at the music* (no needle thunk). So the two streams have **different natural start points and are not the same timeline.** The system stores the offset rather than pretending they align:

- The **audio render** is trimmed to `program` (clean). The **video master** keeps the full event, mute. An `av_sync_offset` (e.g. "video 0:14 = audio 0:00") is stored metadata.
- The offset is **proposed once** (by onset cross-correlation: music-onset in the audio vs. motion-onset in the video) and **frozen** — never recomputed per render (principle 2.3; the same safety as trim points).
- A **trimmed/synced video derivative** can be generated on demand (video trimmed to the offset, frame-aligned to the audio, with the real audio master muxed in only when you want a watchable-with-sound file — a deliberate occasional output, never the archival or playback path). The full needle-drop master stays in the asset namespace; the trimmed-synced derivative is what co-resides beside audio (§12.4).

9.3 Universal and effortless, including digital sources

Any station can carry a camera that rolls on the **same trigger that starts a capture** — including a CD rip (disc insertion / rip start). You do nothing extra. The sync model differs for digital:

- For a CD, there is **no analog playback timeline** (data extraction, not playback), so the video is a **detached aesthetic asset** by default (`timeline_model: digital_exact` , no `av_sync_offset`).
- A "watch-along" for a CD (see the disc, hear the music) is still possible as a derivative: lay the ripped audio under the video at a **stored, frozen offset** (e.g. "audio starts at tray-close," proposed automatically, confirmed once). Deterministic, not a per-render guess, not a manual chore.

So CD video stops being a special "production act" — it's just "a station with a camera," with the watch-along a stored-offset derivative.

One caveat so it isn't lost: everything in §9.3 assumes the CD is being *ripped* — a bit-exact digital extraction with `timeline_model: digital_exact` (§6.6, §11.1). But a user may deliberately choose to **play a CD through an analog rig and capture the analog output** instead — for the sound of a particular player's DAC and output stage, or simply because that is the chain they have. That is a fully valid transfer: it is an `analog_continuous` capture like any vinyl or tape transfer (onset detection, trim regions, av-sync

machinery all apply), its equipment chain records the CD player → ADC path honestly (§5.4), and it produces a normal analog-sourced master — *not* the bit-perfect data copy that ripping yields. The two coexist: a CD can have a bit-exact rip master *and* an analog-transfer master of the same disc, as different captures under the same copy (§5.1), preferred between on the merits. The digital-source video model above applies to the rip; the analog-transfer capture instead uses the ordinary analog av-sync model of §9.2.

---

10. Enrichment and the LLM layer

Enrichment is the "make sense of it lazily" engine: it turns raw captures + scans into *proposed* metadata. It runs server-side as workers pulling `task` rows.

> **Background for the non-specialist: what an LLM is, and why one belongs here.** A *large language model* (LLM) is an AI system trained to understand and produce language; a *vision-capable* LLM (or "vision model") can also look at an image and describe or reason about it. The ones relevant here are the kind you may have used through a chat assistant — give it a photo of a record sleeve and a question, and it answers in words. Lacquer uses such a model for exactly the jobs a knowledgeable human helper would do while cataloguing: read the text off a liner-note photo even when it's skewed or faded, recognize a logo, and make an educated guess about an ambiguous case ("a CBS Mastersound classical cassette from 1982, judging by its j-card, was *probably* Dolby C"). Those are pattern-recognition and world-knowledge tasks that hand-written rules do badly and a model does well, which is why one earns its place in an archival pipeline.
>
> Three things keep this safe and honest, and they hold everywhere the LLM is mentioned in this document:
>
> - **It only ever proposes; you confirm.** The model writes to a `proposal` table, never to a master or a confirmed field (§10.1). Its output is a suggestion surfaced for a one-tap yes/no, not an edit. A wrong guess costs you a tap, never your data.
> - **It is entirely optional.** Per the local-first law (§2.4), the LLM is a *convenience, never a dependency*. With no model configured, every capture still ingests, plays, and serves perfectly; the only difference is that the "what is this?" questions sit in the reminder queue waiting for you instead of arriving pre-answered.
> - **It can run locally.** The appliance default is an on-device model, so enrichment works offline with no cloud account and no data leaving the box (§10.2); a remote API is an opt-in upgrade for

> harder cases, not a requirement.
>
> So throughout this document, "the LLM reads the cartridge label" or "the LLM infers the Dolby type" should be read as: *an optional assistant looks at the evidence and suggests an answer, which you can accept or ignore.* Nothing below depends on it being present.

10.1 Pipeline per release

1. **AcoustID / Chromaprint** on each recording → candidate identities (cheap, deterministic, first).
2. **Disc ID / AccurateRip** (CDs) → often an exact release match.
3. **OCR + vision LLM** on scans → proposed fields (artist, title, tracklist, catalog #, label, matrix, year, credits) **with confidence and per-field provenance** (which image each field came from). The LLM also does the Dolby-type/quad-encoding reasoning (§7).
4. **Reconciliation** — merge fingerprint matches + OCR proposals + any user input; matrix/catalog lookups against MusicBrainz/Discogs.
5. **Write to the `proposal` table, never directly to the release.** High-confidence + corroborated proposals (e.g. AcoustID and OCR agree) may auto-apply per a configurable threshold; everything else becomes a review task. The LLM **never overwrites a master or a confirmed field**; it only proposes. Raw model responses are retained so reconciliation can be re-run later with a better model without re-photographing anything.

10.2 Swappable backends: local / remote / none

Enrichment sits behind a single internal interface ("given these images/fingerprints, propose structured metadata with confidence + provenance") with pluggable backends:

- **Local model** — an on-device vision-capable model, sized to the hardware. This is the **appliance default**: enrichment works offline, no API key, no cloud — the only acceptable default for a sealed appliance, and a privacy win. A smaller model leans toward "flag for review" rather than "auto-apply," which is the safe bias anyway.
- **Remote API** — Anthropic/OpenAI/etc. via pasted credentials (a GUI/settings toggle). For frontier-quality extraction on damaged liner notes or obscure pressings.
- **None** — the system still fully functions; enrichment simply stays in the nag queue (local-first law).

Optional **tiering**: the local model does a cheap first pass on everything; the remote backend is invoked only for low-confidence cases or an explicit "really try hard on this one," minimizing cost and cloud dependence while getting frontier quality where it matters.

11. The three generators

The system's extensibility (principle 2.5). Three declarative spec languages, three generic interpreters, one immutable core. Each spec is authorable by hand or by an LLM that understands the schema (the LLM fills a structured schema — reliable — rather than writing imperative engine code — not reliable). All three are reviewed and committed; nothing auto-applies engine behavior.

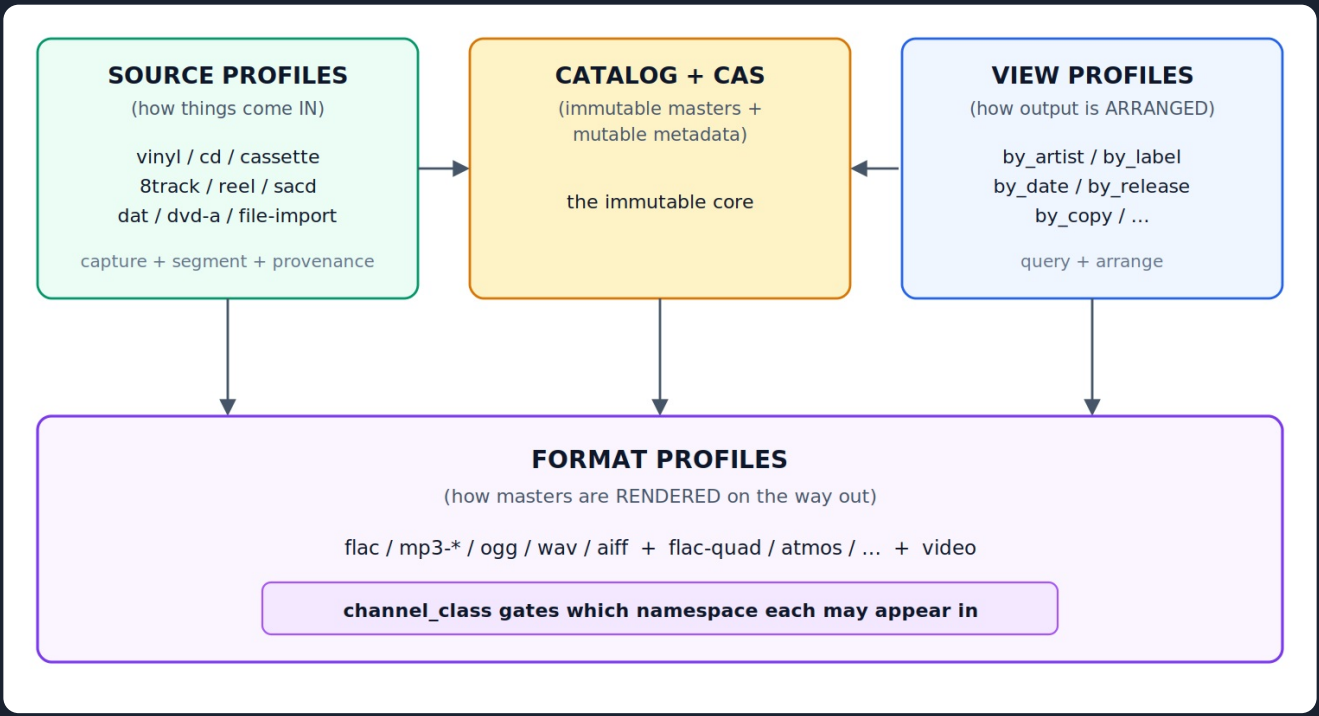


Figure: the three generators. (An editable vector version, `generators.svg`, accompanies this document.)

11.1 Source profiles (input)

How a kind of media is captured and what master it yields. Adding reel-to-reel, DVD-Audio, SACD, DAT, MiniDisc is a spec, not surgery.

- `name` — `vinyl`, `cd`, `cassette`, `8track`, future `reel-to-reel`, `sacd`, `dat`, `multitrack-reel`, …
- `capture_method` — `analog_line_in` | `optical_disc_rip` | `digital_transfer` | `file_import`.
- `timeline_model` — `analog_continuous` (onset detection, trim proposals, av-sync machinery runs) | `digital_exact` (exact TOC boundaries, no onset ambiguity, video detached). This is the flag that cleanly separates CDs from records.
- `master_spec` — channels, target rate (with the §7.5 calibration hook), bit depth, codec. (SACD's

master is DSD — a special case that the format generator then knows to derive PCM from. A `multitrack-reel` master has *N* channels that are session tracks, not a playback layout — see `output_namespace` below and §5.6/§12.5.)

- `output_namespace` — which top-level tree a master from this source surfaces in: `music` (the default stereo world), `multichannel` (consumer surround layouts, §12.3), or `sessions` (multitrack/stem source masters — "stems" being the isolated constituent tracks of a recording, the separate drum, bass, and vocal takes a mix later combines — whose mixdowns are renders that themselves surface in `music` / `multichannel`, §12.5). This is what keeps a board reel out of the listening library and a quad master out of the ancient-client-safe stereo tree.
- `detection` — udev for discs, signal-onset for analog, LLM for physical markings.
- `segmentation` — silence-gap (vinyl), program-change + loop-correct (8-track), TOC (CD).
- `chain_schema` — relevant equipment-chain fields (a reel deck has IPS speed, track format, NAB/CCIR EQ; a DAT has none of that but a recorded sample rate).
- `special_handling` — pointers to format-specific processing (8-track loop-seam, CD-4 demod, Dolby decode).

11.2 Format profiles (output)

How masters render into files. Adding a codec/bit-depth/channel combination is a spec; the transcode engine never hardcodes a format list.

- `name` — the leaf name (`flac` , `mp3-320` , `mp3-64-mono` , `ogg-128` , `wav` , `aiff` , future `flac-quad` , `atmos-ec3` , `mp2-amiga`).
- `container` + `codec` + parameters — the ffmpeg recipe: codec, bitrate/quality, sample rate, **channel layout**, bit depth.
- `channel_class` — `stereo` | `mono` | `multichannel` . **Gates which namespace the profile may appear in:** stereo/mono populate the universal tree; multichannel only appears in `/multichannel/` . This structurally prevents an exotic format (Atmos, quad) from polluting the ancient-client-safe stereo tree.
- `size_model` — `exact` (PCM: computable arithmetic) | `cbr_exact` (CBR-MP3: frame-count arithmetic) | `render_then_record` (everything else: size known only after first render). Feeds §12–13.
- `tag_policy` — `minimal` (compatibility floor) | `rich` , scaled to the profile's likely destination.
- `source_req` — what master this can derive from (an Atmos profile *requires* an object/multichannel master; it can't be conjured from 2 channels — the generator refuses rather than fabricating channels).

- `gapless` — whether the codec/settings preserve gapless.
- `regions` — which labeled regions to include (this is how `-AMBIENT` variants are defined: "include `mechanism_sound` + `lead_in_noise`").

The proof this abstraction is load-bearing: **"12-bit MP2 at 22 kHz for an Amiga" is one config block** — `container=mp2, bit_depth=12, sample_rate=22050, channel_class=stereo, tag_policy=minimal` — and it appears wherever its channel_class permits, served by the same engine, with no code change.

11.3 View profiles (arrangement)

How output is arranged into browsable trees. Adding `by_genre`, `by_decade`, `by_producer`, `by_recording_engineer` (you have the equipment chain!), `by_matrix_family`, `by_completeness` is one spec each.

- `name` — the top-level folder.
- `path_template` — the directory shape as an ordered list of levels, each a field or function: e.g. `[artist_initial_bucket, artist, album, release, format_leaf]`. `by_release` is just a different template over the same data (album level collapsed).
- `filter` — an optional predicate (`by_recently_added` = `ingested > now - 90d`; `by_needs_attention` = `completeness < 1.0`).
- `order` — sort.
- `bucketing` — a per-view function: `[A-Z0-9]` initial for name views, decade/year for `by_date`, none/month for `by_recently_added`. Makes huge directories browsable and helps clients that choke on enormous `readdir`s.
- `leaf_set` — which format leaves appear (usually the universal stereo set; `/multichannel/` views use the surround set). This is where view meets format generator, and it supports **per-mount scoping** (§12.3).
- `aliases` — computed symlinks like `_preferred`.

A view owns *queries that resolve to tracks and how they're arranged*; it does **not** own files. The materialize/trigger policy (§12) decides when bytes exist. A new view inherits all existing trigger behavior for free.

---

12. Render-at-ingest and the virtual filesystem (VFS)

The archive must expose, over many protocols, a tree like:

`/music/{by_artist,by_label,by_date,by_recently_added,by_release,by_copy}/[bucket]/<artist>/<album>/<re`

…where exactly one immutable master exists per recording and every output file is derived. The central engineering tension was: *a filesystem must report a file's size before the file exists, and must satisfy clients that demand exact sizes and random-access seeks, but transcoding is sequential and yields an unknown final size.* After working through every read-time trick, the resolution is simple and decisive.

12.1 Render-at-ingest: the renders are real files

Renders are produced when the master arrives, not when a client reads. When a release is captured and ingested, the system immediately renders the standard profile set (flac, mp3-320, mp3-128, the formats you actually use) to permanent storage, recording each file's exact size. This takes seconds-to-a-minute in the background while you're putting the record away. By the time anything — your old Mac over AFP, iTunes over SMB, a remote peer — browses that folder, **the files are real, correctly sized, instantly served, gapless intact.** No stall, no estimate, no warming-as-anxiety, no race.

This dissolves the entire size/timeout problem class, because there is no longer a gap between "the file should exist" and "the bytes exist." The "VFS" becomes a **view/naming layer over mostly-real files**, plus an occasional on-demand transcoder for the long tail.

Rejected alternatives, for the record (each fails on correctness, not taste):

- **Estimate-and-stream** (advertise an estimated size, transcode during read) — breaks filesystem protocols, which treat size as structural.
- **Over-estimate then truncate** — client-dependent breakage (wrong durations, tail-read failures, hangs); SMB/NFS especially treat the advertised size as real and error on premature EOF.
- **Over-estimate then pad to size** — **destroys gapless for everyone**, including players that otherwise handle it, because the arbitrary trailing silence isn't in the codec's delay/padding metadata. Fatal.
- **In-band "not ready" signals** (a `RELOAD_IN_30s` placeholder file) — absurd for normal clients (iTunes would import a phantom and still race). No filesystem protocol has a clean "not ready, try later" for a file that's supposed to exist; the honest answer is to make the file actually exist first.

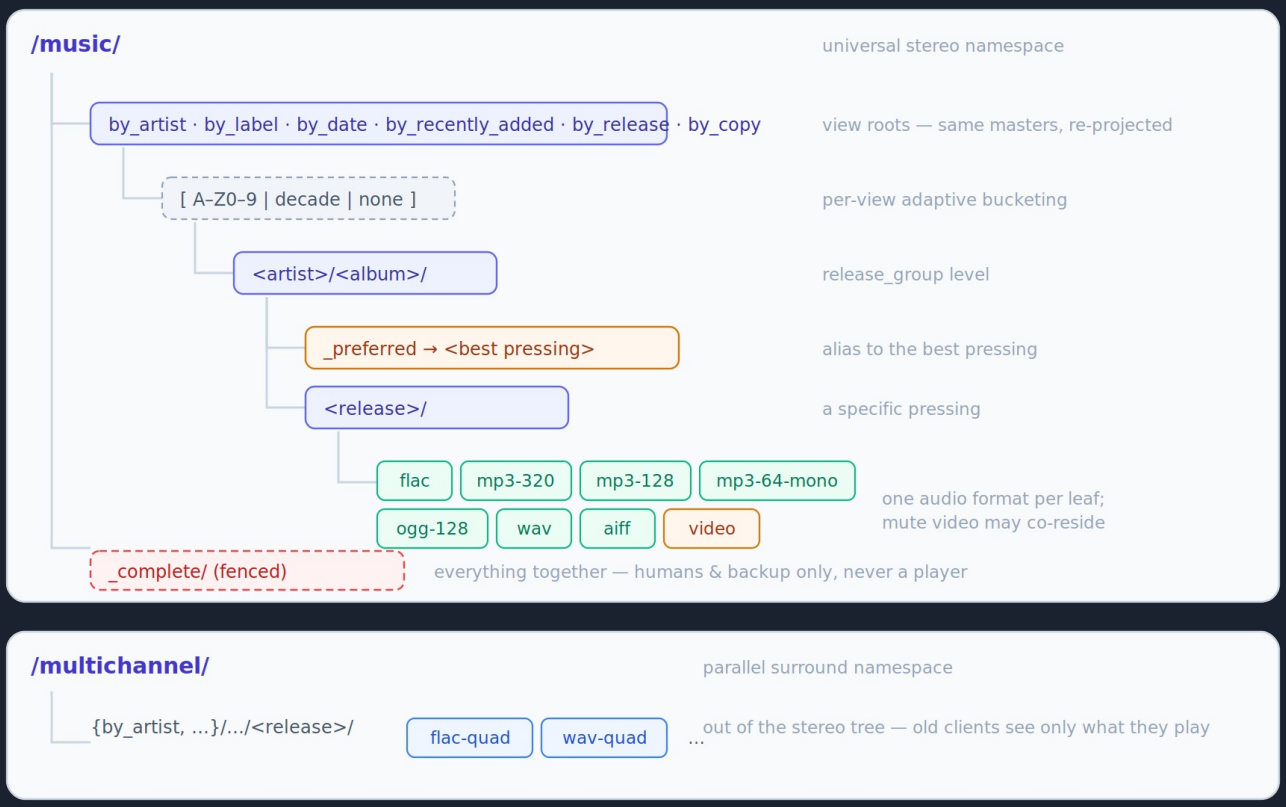
12.2 The rendered-output store and on-demand tail

- Renders live in a store keyed by `(master_hash, profile)` — coherent forever, because the master is immutable, so a cached render is valid until deliberately evicted and re-derives identically.
- **Retention is tiered and mostly "keep":** pin commonly-used profiles permanently (you wanted to

keep re-encodes anyway, and disk is cheap relative to the master pool); LRU-evict the rest under a budget; mark rarely-used profiles (the Amiga MP2, a one-off Opus) **on-demand-only** to save space — those have no timeout pressure because no one points an old Mac at them. First touch of an on-demand profile renders it (seconds) and it becomes permanent too.

- **Render-from-master only**, never from another lossy rendition (so `mp3-64-mono` comes off the 88.2/24 FLAC), enforced because every profile reads the master blob by hash. In-flight dedup: two clients requesting the same `(master, profile)` attach to one ffmpeg job.
- **Gapless** is a release-level flag selecting gapless-aware encode settings (LAME delay/padding metadata for MP3; inherent for FLAC/Ogg/Opus) *and* triggering whole-folder render so an album never gaps at a track boundary. FLAC decodes many times faster than realtime, so feeding the render engine is cheap.

12.3 The view tree



- `by_artist` expands `artist/album/release/format/`, with a `_preferred` **alias** at album level so a lazy client grabs `_preferred/flac/…` and gets the best pressing without choosing. `by_release` collapses the album level (each pressing is a top-level entry under the artist). `by_copy` drills one deeper (release → individual physical copies) for A/B-ing.
- `[bucket]` keeps directories browsable and helps clients that choke on huge `readdir`s.

- `_complete/` is the “everything together” view — every format + video + scans in one place — but it is **deliberately fenced** (leading underscore / separate mount) so a naive music player never wanders in and duplicate-imports the same track in five formats. It is for humans and backup tools, never pointed at a player. (A peer-level `all_data` leaf was rejected precisely because it reintroduces the duplicate-import problem the per-format leaves exist to prevent.)
- `/multichannel/` is a parallel namespace. The `channel_class` gate (§11.2) keeps surround formats out of the universal stereo tree, so an ancient client mounting `/music/by_artist/` sees only stereo files it can play, no dupes, nothing unplayable. Quad fold-downs appear (labeled) in the stereo tree; discrete/decoded versions live here.
- **Per-mount leaf-set scoping:** each consumer can be shown exactly the formats it likes — the Amiga’s mount shows `{mp2-amiga, mp3-128}`, the hi-fi streamer’s shows `{flac, wav}`, the default shows the universal set. Just a `leaf_set` filter on the view/mount.

12.4 Companion files and variant naming

A leaf contains **exactly one audio format** (never two audio encodings of the same track — that’s the duplicate-import sin), but may carry non-audio companions that can’t cause a naive audio client to duplicate-import a track. A `flac` leaf therefore looks like:

```
…/flac/
   Marvin Gaye - Let's Get It On (1973) US MoFi [matrix ABC-123] - recording 1.flac
   Marvin Gaye - Let's Get It On (1973) US MoFi [matrix ABC-123] - recording 1.mp4
   folder.jpg
```

The three entries play distinct roles:

- `… - recording 1.flac` — the audio track itself, the one file an audio client imports.
- `… - recording 1.mp4` — the **mute performance video**, co-resident beside the audio (same basename) so it travels with the track but is never mistaken for one.
- `folder.jpg` — the **cover art**, in the universally-understood `folder.jpg` location that essentially every player and file browser already recognizes.
- **Mute video co-resides safely** — an audio client imports the `.flac` as the track and ignores/file-as-video the `.mp4` (mute → not music → not a duplicate track). The safety depends on the video being mute; **mute is enforced** on any video in an audio leaf. The co-resident video is the trimmed/synced derivative (§9.2) so any accidental together-play lines up; the full needle-drop master lives in the asset namespace.
- **Variant suffix convention:** the default clean render has no suffix; variants carry a label — `<basename> -<VARIANT>.<ext>` (`-AMBIENT`, `-QUADFOLD`, `-FLAT`, `-LOCKEDGROOVE-PRESERVED`).

Variants are region-selection format profiles (§11.2). **Mute-video variants** (`-AMBIENT.mp4`) co-reside beside clean audio; **audio variants** (`-AMBIENT.flac`) would be a second audio file of the "same" track and so must live in their own variant-scoped leaf, never loose beside the default audio.

12.5 The `/sessions/` namespace — multitrack and stem masters

§5.6 establishes that a multitrack session reel is a master whose "channels" are *session tracks* (drums, bass, vocal) rather than a playback layout — the **stems** introduced in §11.1, the isolated constituent recordings a mix later combines. Such a master needs a home that is neither the universal stereo tree (it is not a stereo recording) nor `/multichannel/` (its channels are not a surround layout — routing a drum stem to your left-rear speaker is meaningless). So Lacquer adds a third top-level namespace, parallel to `/music/` and `/multichannel/` :

```
/sessions/
  {by_artist, by_release, …}/
    <artist>/<recording>/
      <session-master>/                ← the multitrack master (stems as channels/files)
        stem-01.flac  stem-02.flac  …   ← labeled, never reordered (see locked rule below)
      mixes/                           ← renders of the session master
        <basename> -STEREO-MIXDOWN.flac
        <basename> -ROUGH-MONO.flac
```

The defining properties, each chosen to keep `/sessions/` from corrupting anything else:

- **It is a *source* namespace, not a listening one.** Like `_complete/` (§12.3) it is **fenced** — never pointed at a music player, never auto-imported. A stem set is raw production material; a player wandering in would import twelve "tracks" that are really one song's disassembled parts. The fence is the same mechanism (leading-context separation / separate mount) that protects `_complete/` .
- **A mixdown is a render, and it surfaces in the ordinary trees.** The whole point of the master/render line doing real work here (§5.6): a stereo or surround *mix* derived from the session master is a render of it, and it appears where any render appears — a stereo mixdown in `/music/` , a surround mixdown in `/multichannel/` . The session master stays in `/sessions/` ; its mixes live out in the listening world, correctly. A mix that took real work carries `precious` (§5.5).
- **Lazy cataloging, exactly like everything else.** A bare pile of unlabeled stems is a complete, valid, fully-ingested session master at low completeness (§5.3), tracked by the reminder engine (§14). You are never *required* to say which stem is the kick drum; you may, whenever you like, or never.
- **Stem labels are proposed, never asserted — and they may *only annotate*.** This is the one rule in `/sessions/` that is **non-negotiable and stricter than the usual propose-once-freeze**, because the cost of a confident-but-wrong stem label (silently calling the snare the kick in a historical session) is high and not obviously catchable later. The locked rule:

> **Track order and identity are immutable facts of the master.** Stem 5 is stem 5 forever. A proposed label *annotates* a stem — it never reorders, renumbers, merges, or splits the channels of the master. And stem labeling is the one place where the human-confirmation step is **mandatory, not skippable on high confidence**: the model may suggest "stem 3 sounds like a bass DI," but ambiguity is a *required* review flag (§14), never a silent auto-apply. The LLM proposes a label and a confidence; the human confirms; only then is the annotation frozen. The audio channels themselves are never touched.

- **It is explicitly not a DAW.** `/sessions/` preserves and serves multitrack masters and their mixes; it does not edit, mix, or process audio interactively. Producing a new mix is running a render recipe (a format profile, §11.2) over the session master, not opening a mixing console. If you want to mix, you pull the stems into your own tools — and if you re-ingest the result, it lands as a new `precious` render (or, if it is itself a fresh capture of a physical object, a new master). The boundary keeps `/sessions/` a preservation namespace, not an application.

This is why the source-generation work and the namespace work are one idea: §5.6 says *what* a multitrack master is (a master at the deepest generation, channels = session tracks), and §12.5 says *where it lives and how its mixes surface* (a fenced source tree; mixdowns are renders in the normal trees). Together they let the rarest production artifacts — the session reels behind famous records — be preserved as first-class masters without distorting the listening library by one file.

---

13. Protocols and client compatibility

The system serves "100% NixOS servers, many client types" — from a modern surround streamer to a 2005-era iTunes on 256 MB of RAM, an old Mac, an Amiga, a flip phone. Because of render-at-ingest, every protocol is serving real, exactly-sized files, so the historically hard protocols are no longer hard.

13.1 Filesystem-class vs stream-class

The distinction that predicts behavior is *"is the client treating this as a filesystem or as a stream?"*

- **Filesystem-class — exact size required: SMB/CIFS, AFP, NFS, FTP, mounted WebDAV.** These stat the file, believe the size, cache and read-ahead against it. A wrong size causes corruption, wrong durations, tail-read failures, or hangs — not graceful degradation. SMB is the strictest (OS-level redirector caching, unforgiving timeouts you don't control) and AFP matters because old Macs need it. **All served from real rendered files (render-at-ingest), so size is always exact and present.** The

FUSE mount that backs these defaults to exact behavior.

- **Stream-class — tolerant: HTTP (chunked transfer = no Content-Length, read until end), most DAAP/UPnP/DLNA streaming.** These have machinery for unknown-length streams. HTTP is the easy path and goes **direct from the render engine**, bypassing FUSE entirely (it doesn't need the filesystem abstraction). A per-client/renderer quirk table promotes any stream client to exact-file serving when it turns out to demand a size (e.g. a DLNA box wanting a seekbar).

Because render-at-ingest makes the bytes real before access, even strict SMB/AFP "just work": pick a folder, add files, go. The protocol rule is therefore mostly about *which front-end serves which class*, not about size tricks.

13.2 Tag compatibility floor

Two requirements tension: correct geometry / rich provenance vs. "must work on ancient stuff." Resolved with an explicit floor:

- **Universal fields** (artist, title, album, track, year, genre) in **ID3v2.3** (the maximally-compatible version) + **ID3v1** fallback; baseline (non-progressive) JPEG embedded art at sane dimensions (~500 px, not 4000 px) at the correct aspect ratio (no crop-to-square).
- **Rich structured data** (the equipment chain) in `TXXX:LACQUER_CHAIN` + human-readable `COMMENT` (e.g. "Transferred on Nakamichi Dragon, flat, software Dolby B decode" — or whatever deck/chain was actually used) — ancient players ignore frames they don't understand (graceful degradation).
- **Per-profile tag policy:** the lossy profiles likely bound for old devices get the `minimal` floor; `flac` /high-bitrate profiles carry `rich` tags. The full truth always lives in the catalog regardless of any tag format's limits.
- **Administrative metadata is never in tags** (principle 2.7): federation provenance, fetch dates, completeness, task state stay catalog-only.

---

14. The reminder engine

Incompleteness is a first-class valid state, and the reminder engine is what makes laziness pay off — it is a core subsystem, not an afterthought. Completeness (§5.3) is recomputed on every edit; outstanding work becomes `task` rows with priority, age, and surfacing rules.

Task types include: `confirm_identity` (fingerprint found candidates), `add_artist_title` (bare dump), `review_autosplit` (low-confidence track split), `photograph_matrix`, `verify_ocr`, `confirm_dolby_c`,

`needs_manual_pass` (Dolby S / SQ / CD-4), `scan_cartridge` (8-track order), `triage_provisional` (abandoned-looking captures), `review_stitch` (low-confidence panel merge).

Two surfacing classes:

- **Passive backlog** — "17 releases need attention, whenever you like." Surfaced in the `lq` dashboard and an optional daily digest (email / ntfy / Matrix). Digestible, never nagging.
- **Active "act-now" prompts** — fired immediately at capture time when completeness *critically depends on a co-located physical action you can only easily do right now*. The canonical case: "capture that 8-track cartridge label **now** — I need it to get the track order right, and you're holding it." Also: "this vinyl capture has no matrix photo and you've got the record out — capture the runout now?" Delivered loudly and immediately on the station you're at (review screen, phone push), not buried in the digest. **They degrade gracefully:** ignored, they downgrade to a normal passive task with context preserved, rather than vanishing or hounding you. They are recommendations, never gates — the capture is already safely ingested.

The "review station" is just `stationd` / `lq` in a `review` role showing the highest-priority task on a screen near your listening chair.

---

15. Deployment, appliance, and federation

Lacquer scales from one box to a federated network. The application design (everything above) is unchanged across all of these; deployment is a separate layer.

15.1 Clan as the machine fabric

For multi-machine deployments, Lacquer is authored as a **Clan** `clanService` — Clan's inventory model (a higher-level service definition from which per-machine config is *derived by role*) maps exactly onto the topology. Roles:

- `server` — runs `lacquerd` (CAS, catalog, render engine, view/VFS, protocol front-ends, enrichment workers, reminder engine). Usually one machine.
- `station` — runs `stationd` with per-machine **input** settings (§6.1): the list of inputs, each with type, device, chain, calibration.
- `review` — the `lq` UI near the listening chair.

```
inventory.instances.lacquer = {
  module = { name = "lacquer"; input = "lacquer-flake"; };
```

```
  roles.server.machines."archive-box".settings = {
    archivePath = "/archive";
    profiles = [ "flac" "mp3-320" "mp3-128" "mp3-64-mono" "ogg-128" "wav" "aiff" ];
    views = [ "by_artist" "by_label" "by_date" "by_recently_added" "by_release" "by_copy" ];
    protocols = { smb.enable = true; afp.enable = true; http.enable = true; };
    renderAtIngest = true;
  };
  roles.station.machines."studio-box-01".settings.inputs = { /* §6.1 */ };
  roles.station.tags.portable = {};
};
```

What Clan provides for free:

- **Secrets via `vars`** — generated station↔server identity (the SSH-CA pattern certifies host keys across machines), DB credentials, enrichment API keys, federation keys. Declarative, deployed encrypted to `/run/secrets/`, never hand-managed.
- **Networking mesh + internal DNS** — Clan manages a VPN mesh so the roaming portable station reaches the server from anywhere, resolvable as `archive.clan` regardless of location/NAT.
- **Fleet deploy + reproducibility** — `nixos-rebuild` across all machines from one command, all capture-tool versions pinned (no "it worked until I updated something" for capture rigs).
- **Backup orchestration** — Clan's borgbackup service (client/server roles) backs up the masters (the only irreplaceable data).

The **portable non-NixOS station** (a friend's laptop) sits *outside* Clan — it's the static-musl/container `stationd` build, authenticating with a vars-issued token, reaching the server over the same mesh.

Caveat: Clan's inventory/service API is explicitly unstable, and its networking/Yggdrasil services are young/experimental. This is a moving dependency (see tally). The response is *not* to defer Clan to a later phase — quite the opposite. Because Clan supplies the foundational seams (declarative secrets, the identity/CA pattern, the mesh, fleet deploy, backup orchestration) that everything multi-machine and federated later leans on, it is far easier to **design these in from the start** than to retrofit them once a single-box system has hardened around their absence. Even the Phase 1 single box is built as a one-machine Clan so the secret-management, identity, and deploy patterns are in place from day one; the instability is managed by **pinning** Clan and treating its surface as a tracked dependency, not by postponing it. Designed in early, the second machine and the first peer are configuration, not a rewrite.

15.2 The appliance: plug-and-go, fully FOSS

The business goal is a sealed hardware product an archivist or music lover plugs in and uses without touching a config. This is reconciled with the deeply-configurable system via a **two-tier configuration model** — power underneath, zero of it exposed by default:

- **Sealed hardware profile per SKU.** Because you build the hardware, you know its exact bill-of-

materials. Inputs, devices, chains, and **factory calibration** (CD-4 rate, drive offset) are pre-baked into a hardware-profile module keyed to the SKU. A "Lacquer One (Vinyl Edition)" image already knows its cartridge → ADC at the calibrated rate. The buyer never sees an input config. This is precisely what NixOS does well: a reproducible, identical image.

- **First-boot GUI is the only setup surface a normal user sees.** On first power-on the appliance serves a guided web UI on its own hotspot (like any NAS/router): name it, join WiFi, optionally paste an LLM key, optionally pair peers. Answers write to local state (Clan vars / settings store), **never to a hand-edited config file.**
- **Boot-time input auto-detection** backstops the profile: probe what ADC/drive/camera actually enumerated and map to the profile; a unit with no camera attached simply hides the live-capture controls while still accepting flatbed scans and imported image files for visual capture.
- **The FOSS/hacker tier is the same system with the lid off** — the power user drops to the inventory, writes input specs, authors generators, runs a multi-machine fleet. **One codebase, two presentations:** sealed-appliance and open-fleet. The LLM-assisted "add a source/format by conversation" capability is the advanced expansion path (plugging in a reel-to-reel the appliance didn't ship with), never something a normal user needs.

15.3 Federation: a preservation network of archivists

This is where the deeper mission of §1 becomes concrete. Everything else in this document makes *one* archive excellent; federation is what turns many excellent archives into a **permanent, redundant, equipment-diverse preservation network** — the mechanism by which the rarest recordings actually stay alive. The convenience framing ("park your collection at a friend's house, back each other up, browse each other's shelves") is true and is the everyday experience, but it is the surface of something larger: *distribution is how fragile things survive*, and Lacquer is built so that the act of being a good archivist among friends **is** the act of preserving the commons, with no extra effort and no central authority that has to persist.

Two professional archivists should be able to back each other up and browse each other's archives trivially — "old-school P2P file-sharing, for archivists." Clan's Yggdrasil service solves the hardest part (transport between independent parties) almost entirely: it auto-selects the best path (direct internet, mesh, or Tor hidden service) with failover, traversing NAT/CGNAT without port-forwarding, and gives each machine a stable address reachable as `<host>.<domain>`. (Anonymity is not needed here; Yggdrasil provides reachability + encryption, and the Tor path is available for the rare peer who wants it.)

Clan provides transport; Lacquer provides the application layer:

- **Identity & trust** — each archive has a stable cryptographic identity (from its Clan/Yggdrasil key material). Federation is **explicit and mutual**: you add a peer by identity (exchanged out-of-band, like adding a friend), both sides consent. A trusted-peer model, not a public DHT.
- **Granular, revocable grants** — per peer you choose: *browse* catalog, *stream* renders, *pull* masters, *be a backup target*, *push* submissions.
- **Federate the catalog, fetch bytes on demand** — maps onto the master/catalog/render split. A peer browses your **catalog** (small, syncable metadata, with scan thumbnails); when they want to hear something, the **render streams on demand over the mesh** (the §12 VFS, consumer = remote peer). A peer can mount your archive as a remote view in their own namespace (`/peers/librarianA/by_artist/…`).
- **Backup = CAS replication** — "back up to a peer" is replicating immutable, hash-addressed blobs (+ a catalog snapshot): enumerate hashes, ship what's missing, verify by hash on arrival. Resumable, dedup'd (across the whole network — same master = same hash for everyone), integrity-guaranteed by construction.
- **"Parked, readable, immutable" default** — a backup peer can **read and play** what's parked with them (a friend parks his collection at your house and you both enjoy it), and CAS guarantees they **can't alter** it. Optional client-side encryption is a per-relationship toggle for the rare case that wants store-only-no-read; the default is open and friendly.
- **Local-first extends to peers** — anything pulled from a peer is copied locally with **administrative** provenance ("source: peer X, hash Y, fetched Z"), which stays catalog-only and **never enters tags or onward-shared files** (principle 2.7). If a peer vanishes, what you pulled is yours. Peers enrich; they never become a live dependency.
- **Appliance federation** — "pair with a peer" is a QR-code/identity-exchange flow in the first-boot GUI plus a grant menu. Two archivist friends scan each other's codes, pick grants, and are federated backups with browse access — no NAT config, no Nix, no networking knowledge.

15.4 What the network actually preserves

The mechanics above are general-purpose backup-and-browse; what makes them *preservation* is the way they compose. These properties are emergent — nobody has to run a preservation program; the network preserves because of how it is built.

- **Lots of copies keep stuff safe.** A master that exists in one place is one accident from gone. Once it is mirrored to even a few trusted peers — automatically, as hash-verified blobs — its survival no longer depends on any single drive, house, person, company, or funding source continuing to exist. The rarer

the recording, the more this matters, and the rare stuff is exactly what no commercial service will ever bother to keep.

- **Many masters of one cut, and the best one wins.** For a recording rare enough to matter, several archivists may each transfer their own physical copy on their own equipment. The network holds all of these as distinct, provenance-stamped masters of the same release (the `release → copy → capture` hierarchy already models this, §5). Nothing is overwritten and nothing is lost: a community can compare independently-sourced transfers, keep the cleanest as the preferred render while retaining the rest, and re-derive better versions later as tools improve — always reaching for the maximally best version possible, which is the whole point. A worn copy transferred on modest gear and a mint copy on a reference rig can coexist and inform each other.
- **Integrity you can prove.** Because every blob everywhere is content-addressed, the network can continuously verify that what it holds is bit-for-bit what was captured. Decay and tampering are detectable wherever a copy lives, not just at home. This is preservation you can audit, not merely trust.
- **Collections outlive their objects and their owners.** This is the part that is personal as much as technical. The physical hoard is a burden as much as a treasure — you cannot reasonably will a thousand pounds of vinyl and paper to children who may not want it, and a streaming subscription is not yours to leave to anyone. But the *masters* — the actual recordings, captured at the best fidelity your equipment allowed, with full provenance — can be inherited, gifted, or simply kept alive by the peers who already hold copies. When an archivist dies or steps away, their life's work does not evaporate; it persists in the network and remains playable and verifiable. A collection becomes a thing that can be handed down even when the shelf cannot.

Put together, this is the largest-scale answer to the question the whole system exists to serve: *how do we keep the rarest recordings in the world alive, honest, and at their best possible quality, indefinitely?* The answer is a living commons of archivists — each capturing effortlessly on their own equipment, each holding pieces of each other's collections, no center to fail — preserving, circulating, and collaboratively improving the record of recorded sound.

Another way to say all of this: the network lets a community **be its own Library of Congress.** A preservation institution is, stripped to its functions, a few specific things — redundant custody (many copies, many places), provenance and chain of custody, a strict separation of pristine preservation masters from the access copies derived for daily use, and a catalog that makes the holdings findable. Every one of those has a direct analogue here: CAS replication across federated peers (custody), the equipment-chain provenance model (§5.4) and hash-verified fixity (chain of custody), the master/render split (§2.1, §12), and the catalog plus the public registry (§15.5) as finding aids. The system is, in effect, a functional

decomposition of what such an institution does — spread across many small independent holders rather than concentrated in one funded building. The bootstrap is correspondingly humble: a founding cohort of even ten or twenty serious collectors who agree on who backs up whom is already a working preservation institution on day one, and the network grows from there. One honest distinction remains, and it is why §15.5 insists discovery must *precede* strenuous backup rather than replace it: a chartered institution also carries a mandate to exist in perpetuity, with staff and budget, that a federation of enthusiasts does not. The network earns the comparison only once material is genuinely replicated — a holding that has merely been *announced* but never mirrored is known, not yet safe. “Be your own Library of Congress” is a description of the mature, well-backed-up network, and a goal the machinery is built to make reachable.

15.5 Finding the others — the public discovery plane

Everything above (§15.3–15.4) is a **closed, trusted-peer mesh**: you add a peer out-of-band, both sides consent, and that is deliberately *not* a public network. That model is right for where data actually moves — but on its own it has a contradiction with the mission. The preservation goal is *global*, yet the only way to gain a peer is to **already know them**. The network can grow only along pre-existing social ties, so the archivist in Lisbon holding the last copy of something and the archivist in Osaka who would give anything to help preserve it can never discover each other. Worse, the rarest material is rare precisely because it is **invisible**. Take a concrete archetype, which the rest of this section will return to: a musician dubs a few dozen copies of their own cassette by hand and sells them from an open guitar case on a street corner shortly before dying. No label, no catalogue entry, no database knows it exists. One copy survives, held by a single person who happened to buy it that day — undocumented, unmentioned anywhere, one house-fire from total erasure, with nobody else even aware it existed to mourn it. Call it the **guitar-case cassette**: it is the limiting case the discovery plane has to serve, because if the system can make *that* findable, it can make anything findable.

So Lacquer adds a **second, completely separate plane** — optional, and off by default. The two planes differ in kind, not degree:

- **The private plane** (§15.3) is *transactional*: it is where files actually move, and it stays exactly as closed, consent-gated, and trust-based as described above.
- **The public plane** is *informational only*: a hosted, file-free, persistent **registry of what is being preserved**. It carries no audio, no masters, no transfers — and it deliberately **knows nothing about who shares what with whom** (that all happens down on the private plane, off the record). It exists to do one thing the mesh structurally cannot: let strangers who have never met *discover that something — or someone — exists.*

Why hosted, and why an index rather than gossip. Spreading broadcasts by gossip across the existing mesh cannot serve the invisible-rarity case, because gossip only reaches people you are already connected to — and the whole problem is that the holder of the guitar-case cassette is connected to *no one*. The thing must be findable by someone who does not know the holder, does not know the artist, and is not even searching for it specifically. That requires a persistent, queryable, public-by-default index that outlives any single connection — which only a hosted service provides. This is the most natural home for one of the optional hosted services (§19): *we run the bulletin board.* Like every external service, it obeys the local-first law (§2.4) — it is a convenience for discovery, never where anything is stored or depended upon.

Discovery is of people and provenance, not a shopping catalogue. Matching is on the *production* axis — eras, regions, genres, formats, equipment, rescue activity — the Soulseek ethos of community-around-contribution, not the Last.fm axis of shared consumption. You find *the kind of archivist who works the corner you care about*; the conversation about specific holdings happens privately, after you choose to peer. How much to reveal is the individual's call, set per broadcast: from a vague profile ("1970s West African vinyl, serious rig") to a specific, deliberate flag ("I found the last known copy of ___"). Because the material was *legally bought and owned*, announcing that you hold and preserved it is a statement about your own property — not a statement that you are sharing it; the index is structurally incapable of being a distribution directory.

The safety model is the speed asymmetry, by design — not policing. The two planes are tuned to opposite speeds on purpose:

- *Knowledge spreads freely and fast.* The index is open and frictionless to browse, because the survival of a recording should be **knowable** the instant its holder chooses to say so.
- *Access is earned slowly and by hand.* Actually peering — the only path by which bytes move — is deliberately high-friction and relationship-driven: you make real friends, over time, out-of-band. This is intentional and load-bearing.

That asymmetry — *fast to know, slow to get* — is what makes the system **worse at piracy than existing tools while being better at preservation than anything that exists.** A pirate wants fast acquisition; acquisition here is slow and gated behind earned human trust. A preservationist wants permanence of knowledge and the patient building of a trust network among serious people; that is made the path of least resistance. The design selects for the users it wants and against the ones it does not — structurally, by what it makes easy versus hard, rather than by rules to be enforced.

Safety scaffolding is built in and partly mandatory: aliases required, **no real names, no faces/selfies** (rare-collection value attracts real-world predators; the registry should never become a targeting map of identifiable people and their addresses), per-broadcast disclosure granularity, and direct messages that the

user can switch off entirely — broadcast-only is a valid mode. The entire public plane is opt-in; an archivist who never touches it loses nothing.

What this uniquely preserves. §15.4 keeps *copies* alive; this keeps *the knowledge that something ever existed* alive — the most fragile link in the whole chain, and until now the one nothing in the design protected. Masters can be re-derived and copies re-replicated, but a recording no one knows survived is gone the moment its single holder is. An institution broadcasting "we hold and digitized this," even with access fully restricted, is doing exactly this at scale: making survival *known* so a researcher decades later learns the recording exists at all. The public plane becomes a distributed, community-run **registry of what survives** — a different and arguably more historically important artifact than the backup mesh.

This does not replace backing up — it precedes it. Being *visible* on the public plane is the beginning, not the end. The whole point of finding the others is to then make friends, peer on the private plane, and **back each other up strenuously** (§15.3–15.4). Discovery that never graduates into replication has made a recording *known* but not *safe*. The mission needs both: the registry so the guitar-case cassette becomes visible, and the trusted mesh so that once visible, it never dies.

15.6 The service archivist — contributing capability, not custody

Everything in §15.3–15.5 assumes peers trade in *masters*: you hold copies, I hold copies, we back each other up. But a healthy preservation network also needs people who contribute something other than holdings — and the model should represent them as first-class without bending the trust rules. Call them **service archivists**: peers who contribute **capability or labor** rather than (or in addition to) masters. Two distinct kinds, because they attach to the network differently:

- **Custody / infrastructure services** — the peer offers *durable resources*: an always-on LTO tape library or big ZFS pool that acts as a backup target, a relay that improves reachability (§19), a beefy machine that maintains a search index of the public plane (§15.5). What they provide is storage, uptime, or bandwidth. In network terms this is mostly an unusually reliable §15.3 peer, distinguished by the *grants* pointed at it (`be a backup target`) and by an honest record that this peer is offering custody capacity rather than its own collection.
- **Transformation services** — the peer offers *skilled processing*: running a manual Dolby decode through DDi (§7.1), an analog-domain de-hiss or click-repair, a quad decode through the one proprietary tool that does it (§7.3), a careful restoration. What they provide is the thing the §17 dependency-risk register keeps flagging as "no FOSS path, manual pass." Their output is, in exactly the §5.5 sense, a `precious` **render** of a master they were granted access to — sent back to the owner, re-ingested with provenance recording who produced it and how (`dolby_decode: "software-DDi-`

`manual"`, plus the producing peer's identity in administrative metadata, §2.7).

How a service archivist attaches uses machinery that already exists — there is no new permission system. The §15.3 **granular, revocable grants** are exactly the right tool: a transformation service is a peer you grant `pull masters` (or `stream renders`) for the specific items you want worked on, and who is granted `push submissions` so the finished `precious` render comes back to you. A custody service is a peer you grant `be a backup target`. Consent stays mutual and explicit; nothing about "this peer offers a service" loosens the speed-asymmetry (§15.5) — you still choose them by hand, slowly, like any peer.

The one genuinely new thing is *honesty about the relationship*, and it has a deliberate boundary: **the system records and verifies the relationship; it never enforces the human obligation.** Lacquer can record that peer X agreed to de-hiss these twelve masters, can verify by hash that what came back derives from the masters you sent (the parent-master hash is intrinsic to the render), and can mark the result `precious` and credited. What it does **not** do is adjudicate whether X "really" did a good job, chase X for late work, or encode payment or contracts. Those are human matters between archivists. The software's job ends at *making the relationship legible and the artifacts verifiable* — who was asked, what came back, whether it checks out — and the social layer does the rest. This keeps the service archivist firmly inside the local-first, trust-based model: a service archivist is a peer with a particular grant profile and an honestly-recorded role, not a new kind of authority and not a contract engine.

15.7 The discovery substrate — an ownerless, reconstitutable log

§15.5 introduced the public discovery plane as "a hosted bulletin board." That framing is the right *user experience*, but a hosted box that everyone depends on is in tension with the local-first law (§2.4): if discovery lives on one server, the network has a center that can fail, be seized, or simply stop being paid for. This subsection resolves that tension. It is the most forward-looking material in the document — squarely **Phase 5+**, not a dependency of anything earlier — and it is recorded in full because the shape is settled even though the build is far off.

The resolution is to make the discovery plane obey the same survival principle as the masters: **many independent copies, no center.** Concretely, the substrate underneath the bulletin board is a **replicated, signed, append-only log** with these properties:

- **Ownerless and reconstitutable.** Every participant holds the whole log, so it survives the loss of any host the way a master survives the loss of any one drive (§15.4). A hosted bulletin board (§15.5) becomes merely a *convenient view* over this log — nice to have, never the source of truth. If it vanishes, the log is rebuildable from any participant.
- **Metadata only — so no cryptocurrency is needed.** The log carries existence proofs, schemas,

service offerings, discovery profiles, and contact handles (below) — small, bounded records. It carries **no audio and no masters** (those move only on the private plane, §15.3). It is worth being precise about what this log is and isn't, because it has a few **blockchain-like properties** — it is append-only, cryptographically signed, and replicated across every participant with no central owner — that make the comparison natural. But it is blockchain-*like* only in those structural senses; it deliberately has **none of the cryptocurrency machinery** that people usually associate with the word. Because it stores only bounded metadata and rations no scarce resource, it needs no coin, no token, no validators, no mining, and no continuous global consensus — it is simply a gossiped, signed, append-only log. Keeping it that way is a deliberate, load-bearing choice: the moment a design like this introduces a tradeable token, it inherits speculation, incentives that pull against preservation, and a governance surface nobody wants.

- **Externally anchored for non-repudiation.** Periodically the log's head (a hash) is published to some independent permanent record, so that "this existence proof was on the log by date D" is provable after the fact without trusting any single participant's clock. This gives backdate-proof priority ("I had captured this rarity by 2027") without any central timestamping authority.

A **standing discipline** comes attached, and it must be in the document as a real cost, not a footnote: because the log is fully replicated and append-only, **it must stay small** (strictly metadata — hashes, aliases, pointers; never anything that grows unbounded per user) and **you cannot truly delete** from it. Append-only interacts with the alias rules (below) and with anyone who later wants out; "query the whole log" needs local indexing to feel like search. These are the right costs for ownerlessness, but they are real and are noted as such.

Anti-abuse, sized to the three actual attacks. An ownerless append-only log faces three distinct threats, and they do *not* all want the same medicine — naming the mismatch is what keeps the design from over-engineering:

1. **Malformed / invalid entries** (bad signatures, wrong structure). Handled *for free* by the substrate itself: a signed log with a strict schema rejects these deterministically on validation. Every participant checks the rules and drops anything that fails. No special mechanism needed.
2. **Spam / flooding** (millions of junk entries bloating the log everyone must replicate). This is the one that most threatens the "everyone holds the whole thing" property, because the cost lands on every honest participant's disk. The defense is **proof-of-contribution**: to write to the registry you must demonstrate, via a hash challenge, that you have actually done real preservation work (you hold/produced masters with given hashes) — a Sybil/spam gate *denominated in genuine archival effort* rather than in money or raw compute. "Valid contribution" is asserted by versioned, local

software, so the definition can evolve.

3. **Sybil** (one actor wearing thousands of alias-masks). Mitigated by the same proof-of-contribution wall: masks are cheap, but masks *that can prove preservation work* are not, because the work is the scarce thing.

Plural schemas, not one constitution. A quick definition, because the word is doing real work here and has been implicit until now. A **schema** is just an agreed *shape for metadata* — which fields a catalogue record has and what they mean (artist, title, matrix number, source generation, equipment chain, and so on). Nearly everything this document has described — the `release_group → release → copy → capture` hierarchy (§5.1), the `dolby_type` and `source_generation` fields, the equipment-chain structure (§5.4) — is in effect *a* schema: one particular, opinionated way of describing recordings. The important thing the discovery substrate makes explicit is that this schema was **always only one choice among many, never a mandated standard.** The log therefore carries a *registry of schemas*, not a single blessed format: anyone can publish a schema; anyone can adopt, ignore, or fork one. This is the **"be your own Library of Congress" idea** (§15.4) made literal and consensus-free: a participant can adopt the Library of Congress's actual published cataloguing schema and use it, ownerless and offline, for free — or run an idiosyncratic private schema that no one else uses, which remains entirely first-class. There is no central authority that blesses the "correct" schema; interoperability is opt-in by shared adoption, exactly as it is for real-world cataloguing standards. (Lacquer ships a sensible default schema — the one this document has been describing — so a user who never thinks about any of this still gets good, interoperable metadata out of the box.)

A multi-entry-type substrate. Putting it together, the log carries several entry types, each with its own visibility and governance, all Phase 5+ and none a dependency of the core:

- **Existence proofs** — public: "a recording with these characteristics exists and is preserved" (§15.5's invisible-rarity case).
- **Schema entries** — public, adopt-at-will (above).
- **Service offerings** — member-visible, contribution-gated: the §15.6 service archivists advertising custody or transformation capability. (Deliberately *not* shouted across a fully public board — capability ads and "I own rare expensive gear" have a different, more sensitive exposure profile than "find the others by provenance interest.")
- **Discovery / profile entries** — alias-governed (§15.5's safety scaffolding: aliases required, no real names, no faces).
- **Contact handles** — see below.

The hosted bulletin board is only a *view*; the data is the log. This is the property that makes the whole public plane robust, and it is worth stating plainly because it is easy to miss. The hosted "bulletin board" (§15.5 — the website strangers browse) does not *hold* the discovery data. It is just a convenient, searchable **front-end rendered over the log**, and the log itself is fully replicated across every participating daemon. So the relationship is the same one seen with resilient public indexes elsewhere: when a site like The Pirate Bay loses a domain (`.org` , then `.se` , and so on), the catalogue does not die with it — because the underlying data lives in many hands, anyone can stand up a fresh front-end at a new address and repopulate it completely from a copy they already hold. The bulletin board here works exactly that way: if whoever runs it disappears, is blocked, or simply walks away, **any participant can spin up a replacement from their own copy of the log**, and nothing is lost — not one existence proof, schema, or contact handle. The hosted service is therefore a pure convenience (faster, prettier search) and never a point of control or a single point of failure; the survival of the discovery metadata rests on replication, not on any one operator staying online.

Contact handles, and the binding channel that falls out of them. A discovery substrate without a way to *reach* the people it surfaces is half a bridge — you can learn the LTO operator or the holder of the rare cassette exists, and then what, email a stranger? So each participant may optionally publish a **self-certifying, hash-based contact handle** (a Tox/toxcore ID is the concrete instance — Tox being a peer-to-peer, account-free messaging protocol whose "address" is itself a public key, so it needs no central server, phone number, or email) as another log entry. Its properties fit the substrate exactly:

- **Consent-before-disclosure.** A Tox-style handle is a public-key identifier with no account, phone number, or email; reaching out is a request the other side can simply never accept, and *a "no thanks" leaks nothing* — the ignored request reveals nothing personal about either party. Contact becomes machine-readable and machine-connectable without exposing a person.
- **Permanence is acceptable here precisely because there are no hard identities.** The unlinkability guarantee lives at the *front* (alias-only, no real-world data ever entered) rather than the *back* (deletion, which append-only forbids anyway). A burned handle sitting in the log attached to a pseudonym, with the current pointer moved on, is findable-in-principle but practically inert. The escape hatch is to *rotate* the handle, or in the limit abandon the alias.
- **Rotation as hygiene, with a sparse-log guard rail.** Handles can rotate (Tox supports changing the anti-spam bytes of an ID). The guard rail the small-log discipline demands: rotate the *public* handle infrequently (real anti-harvest benefit, minimal bloat), while telling *existing* peers the new handle **privately, peer-to-peer, off the log entirely** (one daemon messaging another, "I'm switching, here's my new handle"). Frequent automatic on-log rotation would both bloat the log and, ironically, leak an activity-timing fingerprint of the alias — so rotation cadence and whether each rotation touches the

log are deliberate, not automatic.

- Most participants will never even *see* these IDs: their own daemon may hold one and is likely engineered to discard the rest, so the practical exposure is minimal.

And then the realization that ties the whole federation story together: **the contact channel doubles as the automatic daemon-to-daemon binding channel.** Everywhere peering was discussed (§15.3 backup, federation) there was always a manual seam — two archivists "somehow exchange identity out-of-band" and configure each other. But once a self-certifying handle is on the log and both daemons can read it, the daemons have *a mutual channel before any human configures anything* — and that channel is exactly what is needed to negotiate a peering relationship automatically. Find on the log → connect over the contact channel → the two daemons exchange keys, discover each other's capabilities ("I can offer LTO backup," "I render these formats"), and set up the transport binding — all automatically. This is genuinely elegant rather than merely convenient, because the *social* handshake and the *technical* handshake travel the same path: the §15.3 "like adding a friend" metaphor becomes literally true at the protocol level.

The boundary that keeps this from becoming dangerous is sharp and non-negotiable: **the channel auto-configures the ability to connect; it never auto-authorizes what flows.** Daemons may freely and automatically negotiate transport, keys, and capability discovery (the plumbing). But the decision to actually replicate someone's masters or grant them access **stays an explicit human grant** (§15.3), exactly as before. Otherwise "find me on the log → auto-connect → auto-bind" would slide into "a stranger's daemon is now configured to pull my collection," which is the precise opposite of the trusted-peer model. So: **fast to connect, still slow to grant** — the same speed-asymmetry as §15.5, now operating one layer down. The auto-binding removes the tedious technical friction (IPs, NAT, keys, config) while preserving the deliberate slowness of real trust.

There is a secondary payoff worth recording, because it answers an open §19 question with the same primitive: this NAT-traversing, self-certifying, Tor-capable contact channel is *also* how a constrained client reaches **its own daemon** across the network. The phone-as-remote-control (§19) binds to the home daemon over the very same machinery used to bind two peers. So one primitive serves three needs at once — human-to-human introduction, daemon-to-daemon peering negotiation, and own-device-to-own-daemon reachability — which is the mark of a mechanism sitting at the right level. (Anonymity is not required for any of this; the channel provides reachability and encryption, and the Tor path is available for the rare peer who wants it. Small control messages over Tor by default is a reasonable posture given how little they carry.)

---

16. Phasing

The design above is the destination. It was allowed to run ahead deliberately, to confirm the foundations would not need replacing — and they won't, because every advanced feature folds into the same core primitives. That means you can ship a small, honest v1 and grow into the rest **without architectural rewrites.**

The discipline that makes this safe: **build Phase 1 on the real primitives** (CAS, the release_group → release → copy → capture hierarchy, the three generators, propose-once-freeze, region-aware renders), even though Phase 1 uses only a fraction of their power. The foundation is cheap to build right and ruinous to retrofit. **Clan is part of that foundation, not a later add-on:** even the single box is authored as a one-machine Clan from day one (§15.1), so declarative secrets, the identity/CA pattern, and reproducible deploy are designed in from the start — they are exactly the seams that are painful to retrofit later. What *is* deferred is the multi-machine *use* of Clan (fixed fleets, the roaming station, the federation transport) and federation itself — the most ambitious pieces, riding the most unstable parts of Clan's surface (its inventory API, experimental Yggdrasil). The phases de-risk in a sensible order: those fragile external bets are concentrated in later phases, while the early product rides on the stable, designed-in Clan core and nothing fragile.

Phase 1 — "MPD in reverse," single box, the core loop

The viable starting point and the business proof-of-concept.

- CAS master store + Postgres catalog + the full release_group → release → copy → capture hierarchy (built properly even if lightly used).
- Two capture inputs done *well*: **CD ripping** (whipper + AccurateRip/CTDB verification) and **vinyl** (always-listening RAM-buffer capture, provisional/commit state machine, music-in/labeled regions, equipment-chain provenance).
- **Render-at-ingest** to the universal format set + the **view/VFS layer** over real files, served over **SMB and HTTP** — the "pick a folder, add files, go" experience, end to end.
- The **reminder/completeness engine** — incompleteness as first-class state is the soul of the laziness pitch and the day-one differentiator from "a ripper with a database."
- **One minimal control protocol as the single control surface.** In the MPD tradition, the daemon exposes a plain line-oriented socket (one command per line, simple terminated responses — driveable by hand with `nc`, no HTTP framing). `lq` is built as a *client of this protocol*, not as a privileged path into the daemon — so by construction anything `lq` can do, another client can do, which is what

makes a future phone app (Phase 3+) a thin client rather than a re-architecture. The protocol exposes *state transitions over the existing state machines* (provisional → commit, propose → confirm → freeze), not raw catalog CRUD: browse the hierarchy and the reminder/proposal queue, approve/reject/edit proposals, trigger and watch captures, and converse with the LLM. Two deliberate constraints keep it minimal: (1) LLM chat **streams over this same socket** (the daemon emits tokens until a turn-terminator, the same shape as MPD's server-pushes-when-ready `idle`) — no second protocol and no HTTP in the control path; (2) the daemon owns the LLM backend (§10), so a client converses *through* the daemon and never holds model credentials, preserving the local-first/offline guarantee. Media playback is explicitly **not** on this protocol — a client that wants to *play* audio is an ordinary stream-class consumer of the existing front-ends (§13), so the one place HTTP remains is serving real media bytes, where exact sizes and range requests genuinely matter.

- Single NixOS box, **authored as a one-machine Clan from the start** (§15.1): declarative secrets, the station ↔ server identity/CA pattern, and reproducible deploy are in place from day one even with only one machine, because they are the seams that are ruinous to retrofit. Clan is *pinned* and treated as a tracked dependency; what is deferred is the multi-machine *fleet* use of it, not the scaffolding.

This alone is a novel, useful product: *the archiving appliance that captures effortlessly, keeps a pristine master, serves every device, and tells you what's left to do.*

Phase 2 — Breadth of capture + sense-making

- Cassette (flat-master + Dolby strategy + dual-deck provenance), 8-track (loop correction), visual capture + LLM enrichment with the **local-model default**.
- This is where the "it reads the liner notes and figures it out" magic lands — high marketing value — sitting on Phase 1's foundation unchanged.

Phase 3 — The appliance product

- Hardware profiles, first-boot GUI, factory calibration, sealed-but-FOSS presentation. Turns the working software into the sellable plug-and-go thing. A packaging/UX phase, not a re-architecture.
- **Optional mobile app — a thin client of the Phase 1 control protocol, adding no new daemon surface.** This is "the screen": where an active prompt like "capture the cartridge label now" appears, where you approve/reject the LLM's metadata proposals with a tap, and where you can converse with the LLM. Because `lq` already proved the protocol is the one control surface, the app is purely a new client — and it is a *control* client, not a media client: playing audio on the phone is the existing stream-class path (§13), so the app never reimplements streaming. It earns its place only once there is an appliance to point it at, which is why it lands here rather than in Phase 1.

Phase 4 — Multi-machine via Clan

- Fixed fleets, the portable station, the machine→inputs→chain model deployed across nodes. The single-box Clan from Phase 1 is **extended to the multi-machine fleet** here — the patterns are already in place, so this is adding roles and nodes rather than introducing Clan for the first time. (You are still riding a moving API, so the more adventurous mesh/Yggdrasil pieces are timed for when they have settled enough to depend on.)

Phase 5 — Federation

- Trusted-peer parking, catalog federation, CAS replication, browse/stream over the Yggdrasil mesh, appliance QR pairing, and the optional **public discovery plane** (§15.5 — the file-free registry that lets archivists find each other). The most ambitious and the last — and, because of CAS and the catalog/render split, mostly *assembly* of existing primitives rather than new invention. Note the deliberate inversion: this is the deepest part of the mission (§1, §15.4–15.5 — the preservation commons) yet it ships last, precisely because a distribution network is only worth building once each individual archive is excellent and its masters are worth distributing. You earn the network by getting the single box right first.
- This phase also houses the most forward-looking design in the document, to be approached incrementally and only once the rest is solid: the **service-archivist model** (§15.6 — peers contributing custody or transformation rather than masters, attached via ordinary §15.3 grants) and the **ownerless discovery substrate** (§15.7 — the replicated, signed, append-only log, with no cryptocurrency, with proof-of-contribution anti-abuse, plural schemas, contact handles, and the daemon-to-daemon binding channel that the contact handle turns out to provide). These ride the most fragile external ground (experimental Yggdrasil, and substrate work that is partly ours to build), so they are deliberately last; nothing earlier depends on them.

---

17. Dependency-risk register

A standing list of things needing a suitable (ideally FOSS) workflow replacement, or that we must build ourselves, or that are fragile external dependencies. Per the local-first law, each capture-format item below is backed by a preserved master so a future solution can be applied retroactively.

1. **SQ matrix decoder (vinyl)** — no clean FOSS; proprietary (Stereo Lab) or manual. Captured 2ch master preserved; `needs_manual_pass`. *Build-our-own target.*
2. **CD-4 / Quadradisc demodulator (vinyl)** — proprietary (Stereo Lab) only; the hardest quad format.

`needs_manual_pass`. *Build-our-own target.*

3. **QS matrix decoder (vinyl)** — FOSS exists (`quarkquad/qs`) but JUCE-coupled, not headless. *Package as a headless Nix CLI worker — the most tractable item, a quick win.*
4. **Dolby B/C software decode (cassette)** — partial OSS (B solid, C uncertain); DDi Codec is the quality benchmark but proprietary/Store-DRM'd/GUI-only/no-Linux. *Validate + Nix-package the OSS decoder; longer-term, a high-quality FOSS B/C/S decoder.*
5. **Dolby S software decode (cassette)** — exists nowhere, FOSS or proprietary. Capture flat + flag indefinitely. *Unsolved everywhere.*
6. **Stereo Lab headless-controllability** — evaluate whether it can be driven file-to-file under Nix as an interim proprietary component (low expectation, same Wine/Store wall as DDi). If not, SQ/CD-4 stay manual. *Investigation.*
7. `netatalk` **/ AFP** — the path for genuinely old Macs. *Actively maintained, not a bitrot risk: netatalk 4 is under active development and restores the full classic AppleTalk/AFP protocol suite, so this is a healthy dependency rather than an aging one. Track for version currency only.*
8. **Clan inventory/service API** — explicitly unstable; the whole fleet-management layer rides it. *Designed in from Phase 1 (single box as a one-machine Clan) and* `clan-core` *pinned, since these seams are ruinous to retrofit; expect to chase breaking changes. The multi-machine fleet use lands in Phase 4.*
9. **Clan networking + Yggdrasil** — young/experimental; provides reachability + encryption but **not anonymity** (Tor path needed for that). The federation transport rides moving ground. *Deferred to Phase 5.*
10. **Ownerless discovery substrate (§15.7)** — the replicated, signed, append-only log is partly ours to design and build; the proof-of-contribution anti-abuse wall and the external-anchoring scheme have no off-the-shelf FOSS implementation we can simply adopt. *Build-our-own target; Phase 5+. Kept honest by the "metadata-only, no cryptocurrency, must stay small" discipline so it never grows into a consensus system.*
11. **Contact-handle layer (Tox/toxcore, §15.7)** — toxcore is mature, but the exact semantics of *rotating* an advertised ID while preserving existing friend reachability (nospam-bytes vs. public-key) need verification, as does running it as the daemon-to-daemon binding channel. *Open research point, flagged not blocking; Phase 5.*
12. **Manual/transformation decode dependencies are also a federation opportunity** — the SQ/CD-4/Dolby gaps (items 1–6) are exactly what a **service archivist** (§15.6) can absorb until FOSS exists: the manual pass becomes a `precious` render produced by a trusted peer rather than a permanent local

blocker. *Not a new dependency — a mitigation path for the existing ones.*

---

18. Storage and hardware sizing

Masters dominate; renders are a modest addition; the transcode/render store is disposable. Concrete figures (FLAC ≈ 50–70% of uncompressed; per ~45-min album):

Master format	FLAC / 45-min album	Albums per 1 TB
44.1/16 (CD master)	~270 MB	~3,700
96/24 (vinyl/tape)	~900 MB	~1,100
192/24 (CD-4 only)	~1.8 GB	~550

Planning rule: ~1 TB per ~1,000 albums of 96/24 FLAC; roughly half that for CD-sourced material. A few-thousand-album mixed collection lands at **~2–4 TB of masters**. (Indiscriminately forcing 192 kHz on *every* source would balloon this to ~6–9 TB while merely storing larger copies of sources that hold nothing that high — hence source-appropriate rates, §7.6, which still spend the bits where a source and chain genuinely carry them.) Add the rendered-output store (tens to low-hundreds of GB for a handful of profiles) and the whole thing fits a simple NAS.

Redundancy goes on the masters (the only irreplaceable data): ZFS mirror or RAIDZ + offsite backup (Clan borgbackup, or a federated peer). The rendered-output store can live on cheaper, non-redundant storage since it regenerates from masters.

Compute: FLAC decodes many times faster than realtime; lossy encode (LAME/Vorbis) is several times faster than realtime per core — so render-at-ingest for a 12-track album completes in well under a minute on a few cores, and is done in the background before you ever browse. The local enrichment model is the main compute sizing factor for the appliance SKU: a cheaper unit ships a smaller model (biased to “flag for review”); a pro unit ships a larger one and/or supports a remote-backend escalation.

---

19. Open questions — revisit

These are explicitly **undecided** and are recorded here so they are not lost. Nothing in this section is a design commitment; each is a thread to work through before it earns a place in the architecture above. A

resolved decision already taken in discussion: **the phone is a remote control for the user's own daemon — a screen, an approval surface, an LLM chat window, and an input device (camera + keyboard for phone-originated visual capture, §8) — that holds nothing durable and reaches only its own daemon; the daemon is the federation peer, never the phone.** The questions below are what remain.

- **Optional paid services.** Three *separable* ideas that share one mission-safe seam: they sell help, never custody, and must each obey the local-first law (§2.4) — a convenience, never a dependency, and *a* copy, never *the* copy.
- *Rendezvous / relay* — sells **reachability**. Helps peers (especially constrained mobile clients reaching their own daemon) find each other and, where NAT/platform limits forbid a direct connection, relays bytes blindly. End-to-end encrypted, so the relay sees only ciphertext; bypassed the instant a direct path exists. Holds nothing at rest.
- *Encrypted backup peer* — sells **durability**. A professionally-run, always-on peer that accepts client-side-encrypted, content-addressed blobs, stores them durably, and serves them back — provably unable to read or tamper (hash-verified, encrypted before leaving the owner). Must remain redundancy the owner *adds*, never the single home they surrender to.
- *Public discovery index* — sells **findability**. The hosted, file-free "bulletin board" of the public plane (§15.5): a persistent, queryable registry archivists publish into so strangers can discover what (and who) is being preserved. Carries no audio and tracks no transfers; obeys §2.4 (a discovery convenience, never where anything is stored). The hard part — "global search wants an index, not gossip" — is now addressed by the ownerless, replicated, signed log of §15.7, of which a hosted bulletin board is merely a convenient *view*; what remains genuinely open is the operational question of who, if anyone, runs that convenience view and on what terms, given that the substrate beneath it must survive the view's disappearance.
- *Home-daemon hosting* — still undefined; noted only so it is not forgotten.
- **Key management for encrypted backup.** "We cannot read your data" and "we cannot help you recover it" are the same sentence. Acceptable for the technical-archivist audience; a footgun for the plug-and-go appliance audience (§1), which needs a recovery story — escrow, social recovery across trusted peers, or a printed recovery code. Likely two different product faces by audience.
- **Offline behaviour for the phone remote control.** Degraded-but-useful offline mode (cached views, queued actions that reconcile when the daemon is reachable again) versus a dumb terminal that is useless without a live connection. Leaning toward the former; the open part is how queued actions reconcile on reconnect.

Appendix: the shape of the whole thing

The system is, at its core, **three declarative generators (source, format, view) feeding and draining one immutable, content-addressed archive**, with a patient librarian (the reminder engine) tracking everything not yet known, and a deployment story that scales from one sealed appliance to a federated network of archivists parking their collections at each other's houses. That last part is not a footnote: the same primitives that make a single archive excellent — immutable hash-addressed masters, honest equipment provenance, the copy/capture hierarchy — are exactly what let many archives become a **permanent, redundant, equipment-diverse commons** in which the rarest recordings survive *because* they are held, verified, and re-mastered to their best possible quality by many independent hands, no center required. Keeping that music alive, honest, and inheritable is the end the whole machine is built to serve.

The few concepts that carry the most weight are worth restating in one place, because nearly everything else is a consequence of them:

- **One word, one meaning: "master."** A master is an untouched capture of a physical source object — never a processed file. There can be *several* masters of one recording at different **source generations** (§5.6), and the further-upstream one is preferred on the merits. Anything derived is a **render**; a render that was expensive to make is still a render, marked `precious` so it is backed up like a master (§5.5), and the render a player should default to is marked `canonical` (§5.5) — so the flat Dolby tape stays the master while its decode becomes the thing you press play on.
- **Everything ambiguous is decided once.** Trim points, sync offsets, Dolby type, release identity, which render is canonical, which copy is preferred, a stem's label — each is proposed, confirmed once, frozen, and reused deterministically (§2.3). Renders never re-guess.
- **Incompleteness is a valid state, not an error.** A bare untagged dump is a first-class archive entry; the reminder engine (§14) patiently tracks what is still unknown without ever blocking capture or nagging.
- **The local box owes nothing to the network, and the network owes nothing to a center.** Local-first (§2.4) makes every external service a convenience; and when the network does arrive, even its discovery plane is an ownerless log with no cryptocurrency (§15.7), fully replicated across every participant, so any hosted view of it is disposable and can be **rebuilt from scratch by anyone holding a copy of the log** if it disappears. Survival is by replication — of masters across trusted peers, and of the discovery metadata across every participant.

Almost every hard requirement encountered during design — multi-deck cassette provenance, software

Dolby decoding, quad in all its incompatible flavors, 8-track loop correction, locked grooves, the needle-drop sync problem, masters at production-chain generations, multitrack session reels, serving an Amiga and an old Mac and a modern streamer from one master, plug-and-go hardware, service archivists who contribute labor instead of holdings, and a discovery network with no center to fail — folded into the same small set of primitives without a structural rewrite. That convergence is the main evidence the foundation is right. Build that foundation properly first (Phase 1), and the rest is earned expansion.

Revision history

Version	Date	Summary
0.3.1	2026-06-04	Coherence pass; no architectural or factual changes. **§11.1 / §12.5:** “stem” is now glossed at its first appearance (§11.1) rather than only at §12.5, and the §12.5 definition reworded as a recall, so the term is never used before it is explained. **§15.7:** changed “on-chain rotation” to “on-log rotation” so the wording no longer half-contradicts the section’s own point that the discovery substrate is blockchain-*like* in structure but is not a chain.
0.3	2026-06-04	Clarity and correctness pass; no architectural changes. **§5.5:** reframed the master-vs-`precious` distinction around whether the software can regenerate an artifact unattended (deterministic pipeline vs. manual work). **§6.5:** added a guard so records that legitimately carry the same content on both sides keep both captures distinct (propose-not-merge, visual corroboration). **§7.1:** clarified that the relevant Dolby variants are the *cassette* NR family (B/C/S), with Dolby A noted as a separate professional master-tape system that can appear upstream; corrected Dolby-reading wording (j-card / shell, not “sleeve”/”pressing”) in §7 and §10; added a research note on Dolby C and promo/demo cassettes. **§7.3:** made explicit that hand-run proprietary quad decodes are `precious` like manual Dolby decodes. **§8.3:** retitled “Matrix / runout capture (vinyl)”. **§9.3:** documented capturing a CD through an analog rig (an `analog_continuous` transfer coexisting with the bit-exact rip). **§12 / §2.4:** harmonized the “VFS” acronym, defined at first use. **§12.4:** reformatted the companion-file example so its annotations no longer overflow. **§15.1 / §16 / §17:** Clan is now designed in from the start (the single box is a one-machine Clan), with only the multi-machine fleet use deferred. **§15.5:** introduced the “guitar-case cassette” as a named running example. **§15.7:** removed the zero-knowledge-proofs digression; reframed the discovery log as having blockchain-like structure but no cryptocurrency; defined “schema” and clarified that the schema described throughout the document is the default among many; made explicit that the hosted bulletin board is a rebuildable view over the fully-replicated log. **§17:** corrected the netatalk entry (netatalk 4 is actively maintained).
0.2	2026-06-04	Committed the foundational master/render refinements and the forward-looking federation layers that had been settled in design discussion but not yet written down. **§2.2, §5.5, §5.6:** “master” fixed to mean one untouched capture; introduced *source generations* (session reel → production master → lacquer → stamper → pressing) with `_preferred` made lineage-aware, and the `precious` (back-up-like-a-master) and `canonical` (default-playable) render flags; §7.1 Dolby text reconciled to the new

		vocabulary. **§6.7 (new) + §2.6:** non-intrusive capture-tap principle and LLM-as-calibration-coach. **§11.1, §12.5 (new):** the fenced `/sessions/` namespace for multitrack/stem masters, the locked stem-labeling rule, and the `multitrack-reel` source profile with an `output_namespace` field. **§15.6 (new):** the service-archivist model (custody vs. transformation peers via §15.3 grants). **§15.7 (new):** the ownerless, append-only discovery substrate with no cryptocurrency — proof-of-contribution anti-abuse, plural schemas, contact handles, and the daemon-to-daemon binding channel. **§16/§17/§19 + Appendix:** phasing, dependency-risk register, and open questions updated to wire the new material in.
0.1	2026-06-04	Initial complete draft of the system design: concept and mission, governing principles, architecture, CAS master store, catalog data model, the station/capture model, format-specific capture quirks (cassette/Dolby, 8-track, quad, locked grooves), visual capture, performance video, the LLM enrichment layer, the three declarative generators, render-at-ingest and the VFS, protocols and client compatibility, the reminder engine, deployment/appliance/federation (including the public discovery plane and the “be your own Library of Congress” framing), phasing, dependency-risk register, storage sizing, and open questions.