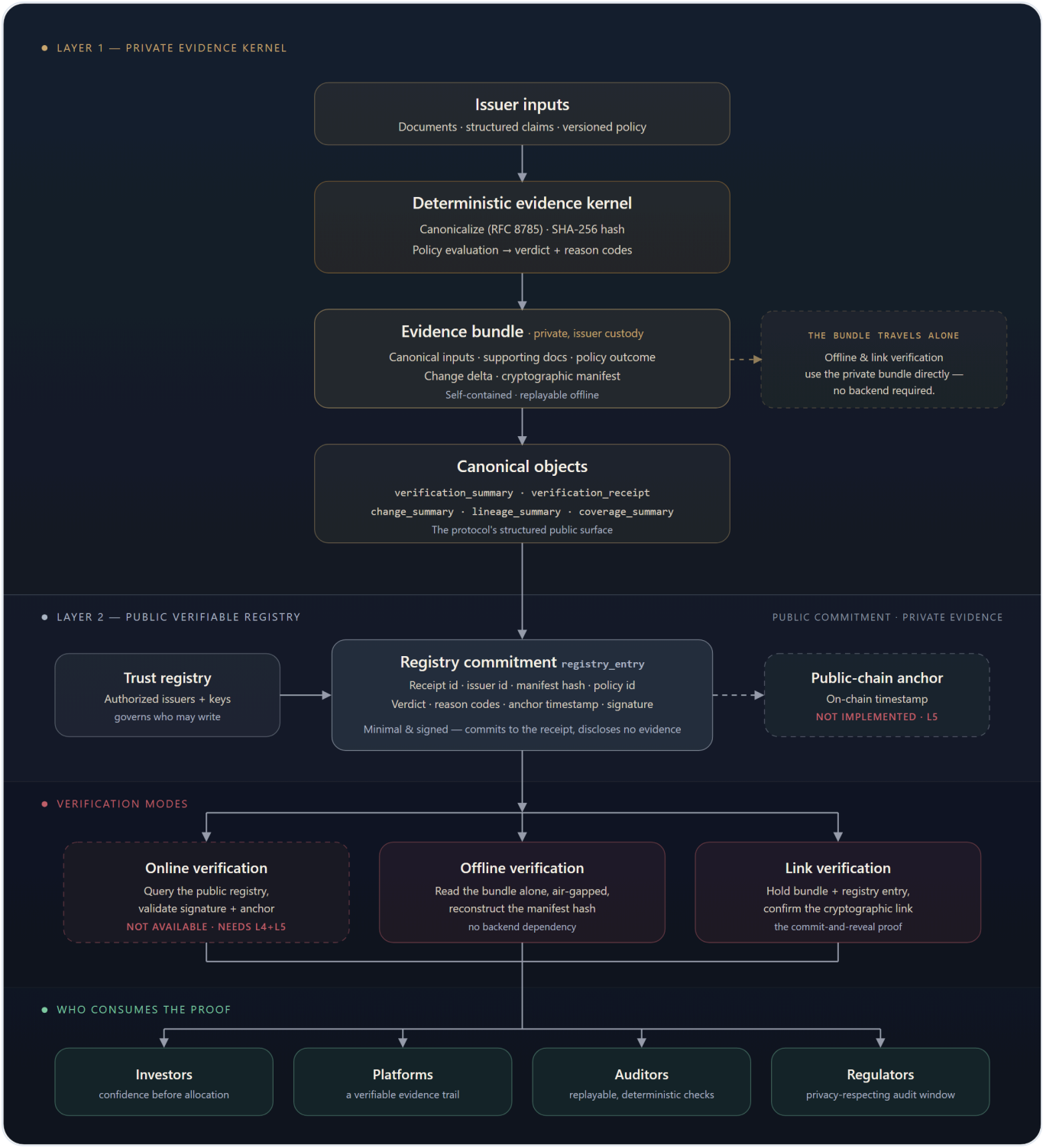
PLATFORM WORKFLOW · LOQUITUR

How the Loquitur platform works, end to end.

PREVIEW | PRIVATE / PUBLIC BOUNDARY | BUSINESS + TECHNICAL

Loquitur turns a value statement — a property valuation or a fund NAV update — into a portable verification receipt that anyone holding the bundle can verify offline, without seeing the underlying evidence. The receipt and its offline verification are implemented and tested; the public registry the design commits them to is not yet operating. This page walks the full flow once, for both business and technical readers.



The diagram reads top to bottom. The upper band is the private evidence kernel under the issuer's control; below the dashed boundary is the public verifiable registry; then the ways anyone can verify a receipt — the diagram marks which are available today; then the audiences that consume the proof.

Every node of the flow, in business and technical terms.

Each stage is described twice on purpose: what it means for a business reader deciding whether to trust the system, and what it means technically for a reader implementing or auditing it.

01 Issuer inputs

BUSINESS

An issuer — a tokenization platform, a fund administrator or an appraiser — starts from the same material they already produce for a valuation or a NAV update: source documents, the financial claims that summarize them, and the rules that define what a sound result looks like.

TECHNICAL

Inputs are typed: documents, structured claims (typed assertions, each with an explicit support basis), and a versioned policy. Scope is intentionally narrow — property and fund-NAV workflows — so the result stays deterministic and reviewable.

02 Deterministic evidence kernel

BUSINESS

This is where a statement ("the value is X") becomes a checkable result. The kernel applies the policy to the evidence and returns a verdict with readable reason codes — not a black-box "approved", but an outcome that explains why it passed.

TECHNICAL

The kernel canonicalizes inputs (JCS / RFC 8785) and hashes them with SHA-256, then evaluates the policy to emit a status (pass / warning / review / fail — the public registry entry restates the same fact as a verdict, PASS / WARN / REVIEW / FAIL) plus standardized, versioned reason codes. Determinism is architectural: identical inputs always produce the same hash and the same verdict.

03 Evidence bundle — private

BUSINESS

All of that work is packaged into one portable file that stays under the issuer's control. Nothing confidential leaves their environment unless they choose to share it, yet the result is fully checkable later.

TECHNICAL

The bundle is a self-contained package — canonical inputs, supporting documents, policy outcome, change delta and a cryptographic manifest. It is replayable offline: it can be re-verified on an air-gapped machine with no backend dependency.

04 Canonical objects

BUSINESS

From the bundle the platform derives a small set of standard summaries that answer the questions a counterparty actually asks: what passed, what changed, what the history is, and what is covered versus missing.

TECHNICAL

Seven canonical objects form the structured surface. Five describe the review — verification_summary, verification_receipt, change_summary, lineage_summary and coverage_summary — and two govern what happens next: registry_entry, the minimal public commitment, and trust_registry, which says whose signature counts. The receipt is the compact, hashable artifact the registry entry later commits to.

05 Registry commitment — public

BUSINESS

The issuer publishes a minimal public record proving a verification happened — by whom, under which policy, with which result — without revealing the underlying evidence. Presence in the registry becomes a reputation signal that compounds with every commitment.

TECHNICAL

A registry_entry is a minimal, signed projection of the receipt: receipt id, issuer id, manifest hash, policy id, verdict, reason codes, signature, and an optional anchor field that is presently null. The design specifies a trust registry of authorized issuers and their public keys to govern who may write, and a content-hash identifier so that anchoring stays chain-agnostic if it is added. Authorization checking against a trust registry is implemented and works offline. Anchoring is not implemented, no chain has been selected, and nothing is anchored anywhere.

06 Verification — online, offline, link

BUSINESS

Anyone can independently check a claim in the way that fits their situation: checking a downloaded file with no internet at all, or proving that a private file and a public record are one and the same. Querying a public record is the third way, and it is not built yet.

TECHNICAL

Three checks today, two more on the roadmap. Structure and integrity: read the bundle alone and reconstruct the manifest hash, no network required. Link: hold both the private bundle and the public entry and confirm the cryptographic link — the commit-and-reveal check. Authorization: check the issuer's signature against a trust registry, which distinguishes revoked, suspended, expired, not-yet-effective and out-of-scope keys, so an authority failure is never reported as an integrity failure. All three run offline. Inclusion proof and anchor finality are not built.

07 Who consumes the proof

BUSINESS

The output is a single proof that different audiences can rely on without having to trust each other: investors gain confidence before allocating, platforms gain a verifiable evidence trail to differentiate, auditors get replayable checks, and regulators get a privacy-respecting audit window.

TECHNICAL

Because verification is deterministic, every consumer reaches the same conclusion from the same artifact — no shared backend, no privileged access, and no silent overwrites: every change is a new snapshot with explicit supersession.

ARCHITECTURE

Two layers: private evidence, public commitment.

The platform is deliberately split in two. Layer 1 is a private evidence kernel that runs inside the issuer's environment and produces the portable bundle. Layer 2 is a public verifiable registry that holds only minimal, signed commitments to those bundles.

The operating default is "public commitment, private evidence": issuers commit publicly to the existence and integrity of a receipt, while the evidence itself stays under their custody and is disclosed only on demand, by their choice, to authorized recipients.

This separation is what lets a business share proof with a counterparty without exposing a data room, and lets an engineer verify that proof with nothing more than a file and a public record.

WHY A REGISTRY

A registry, not just a verification tool.

A verification tool answers "is this valid?" for whoever runs it. A registry answers a harder question that no single tool does: can the existence and integrity of a receipt be proven publicly, later, by anyone, without disclosing the evidence behind it?

A registry that is public, append-only and cross-issuer would compound in value as receipts accumulate — a track record that cannot be back-dated and does not move if a competitor builds the same software. None is operating yet; this is the argument for building one, not a description of one. That durability is the point: trust is earned from adoption, not declared.

BOUNDARIES

What the platform is and is not.

Loquitur governs the evidence behind a value and proves the verification happened. It does not calculate the price, does not issue tokens, does not transport data on-chain in real time, does not perform KYC/AML, and does not take custody of the evidence.

The correct framing is consistent across the whole flow: the platform makes an external mark or NAV update replayable and independently verifiable, and provides public proof of that without forcing disclosure of confidential evidence.

DESIGN PRINCIPLES

Five rules the flow never breaks.

Determinism

Same inputs, same hash, same verdict — every time, on any machine.

Explain the verification

A PASS always carries reason codes, support posture and policy identity. If it cannot be explained, it is not verified.

No silent overwrites

Every change produces a new snapshot with explicit supersession and a structured delta. History is extended, never rewritten.

The bundle travels alone

If a bundle needs our backend to be verified, the design has failed. Offline, reproducible verification is a protocol property.

Public, neutral, durable

The design requires that a committed entry persists, and that the operator can neither censor nor retroactively edit. No registry is operating, so this is a constraint on what may be built, not a representation about current conduct.