SAWBILL

WHITE PAPER / 04

EVIDENCE

Prove Without Overclaiming

Evidence, attestations, provenance, and auditability in Sawbill

How Sawbill turns hostile observations into bound, validated, attributed, and verifiable evidence.

READERS

Architects / Security and audit leaders / Quality and compliance leaders / AI platform leaders

Contents

Prove Without Overclaiming

The issue in one minute

Agentic systems produce an abundance of traces: messages, tool output, job statuses, tests, signatures, commits, and provider logs. This abundance can create an illusion of proof. Yet a log may be partial, a signature may cover the wrong bytes, and a `Completed` status may say nothing about the business objective.

Sawbill turns these signals into a controlled evidence chain:

> **observe, bind to an exact claim, verify, validate, admit as evidence, evaluate gates, decide, and then retain an auditable history.**

Each step increases the information's usability without expanding what it can actually support.

Abundant data does not constitute evidence

An artifact is usable only if several questions can be answered:

- which exact bytes were received?
- from which source, and when?
- which claim are those bytes meant to support or refute?
- for which subject, version, and attempt?
- are the format, signature, and trust chain valid?
- is the evidence fresh, revoked, or replayed?
- does the validator have the required qualification and independence?
- which coverage or interpretation limits remain?

Without these bindings, evidence is merely a document adjacent to the subject, not a verifiable reason to make a decision.

The single evidence path

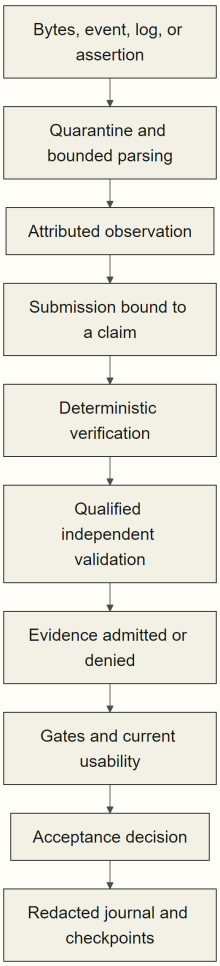

Figure 1 - Sawbill

There is no privileged path for an administrator, webhook, provider, or internal agent. Each begins with a hostile observation.

1. Observe without believing yet

An attributed observation records that an event or bytes were received from a given source, under a given scope and cutoff. It does not yet say that the content is true or that it constitutes evidence.

This distinction is decisive for integrations:

- `Completed` becomes "the provider emitted this status";
- a scanner report becomes "these bytes were received from this tool";
- a signature becomes "a cryptographic assertion to verify";
- human testimony becomes "this person made this claim."

The signal is retained without allowing its producer to choose its evidentiary weight.

Integration turns tool output into controlled input

In an existing chain, Sawbill collects without granting implicit trust. Webhooks, APIs, files, logs, and attestations arrive through adapters that preserve their source and format. They first become observations; using them as evidence is a separate decision.

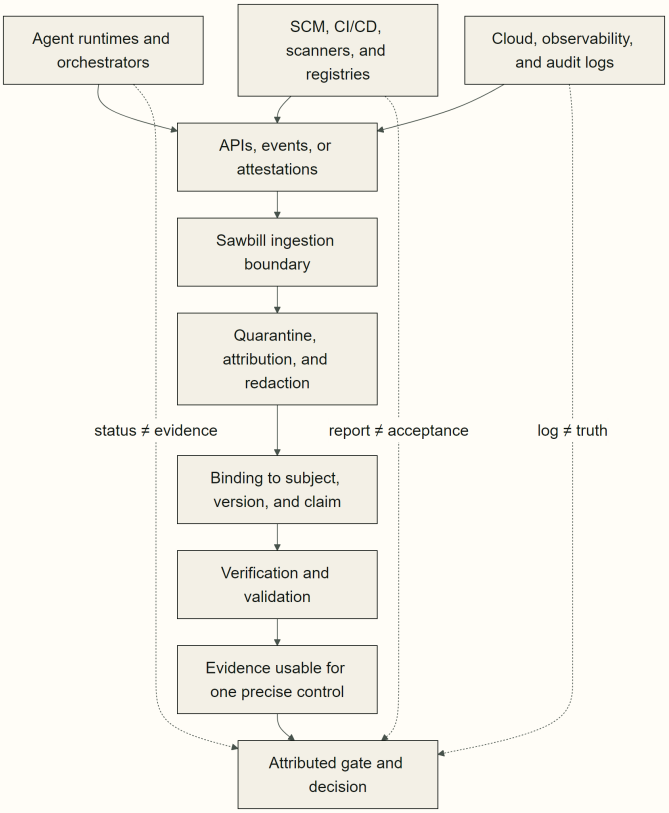

Figure 2 - Sawbill

The boundary therefore moves from retaining traces to the quality of their binding. An external platform may state that a job is complete; it does not choose what the signal proves, for which version, or with what weight in a Sawbill decision.

2. Bind the observation to a precise claim

An evidence submission proposes using an observation for an exact predicate, subject, and purpose. It may support or refute a requirement, but it is not yet admitted evidence.

Depending on the case, the binding includes:

- the identity of the raw observation;
- the subject and its version;
- the requirement or predicate;
- the expected polarity;
- the target, realization, contract, or pack;
- the attempt and execution context;
- the relevant organizational and technical scope;
- the method, tool, and configuration;
- freshness, retention, and declared limits.

The same bytes used for two claims produce two distinct submissions. Content identity may be the same; the evidentiary relationship is not.

3. Give content a cryptographic identity

A digest alone is ambiguous when the format and domain are unknown. Sawbill therefore associates the fingerprint with the content type, format, and algorithm used.

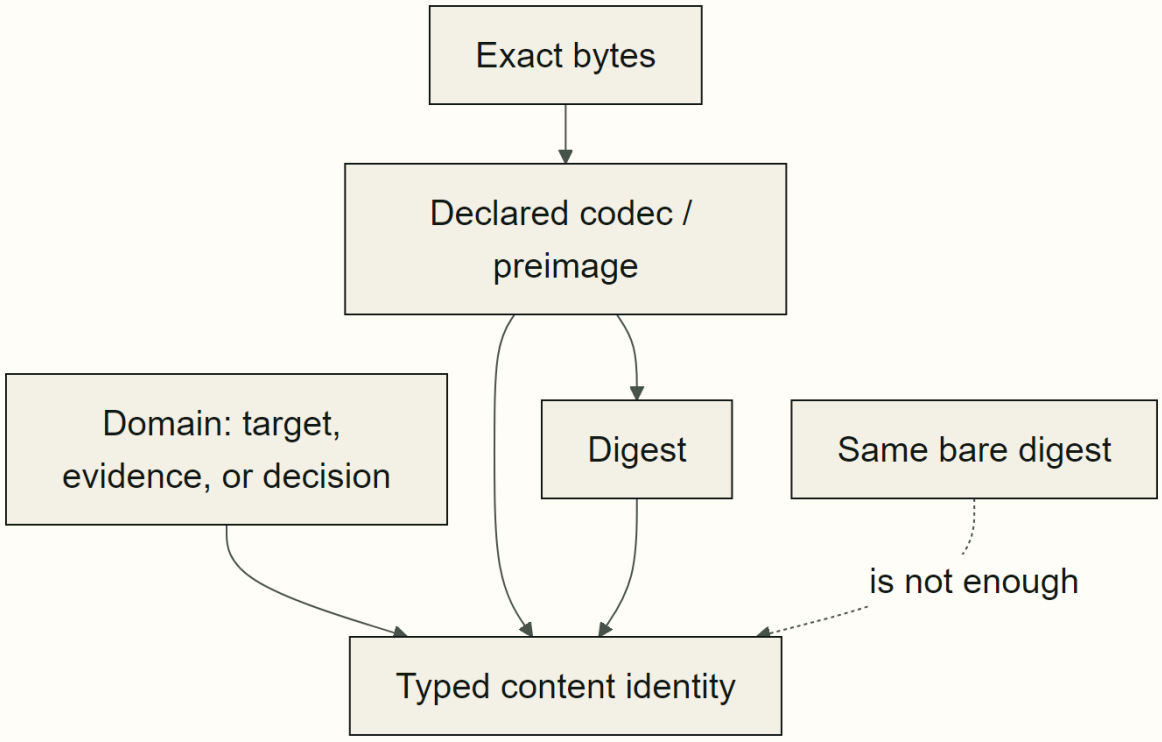

Figure 3 - Sawbill

For Sawbill JSON records, a canonical profile can rely on the JSON Canonicalization Scheme, RFC 8785. For an external artifact, Sawbill preserves the native encoding specified by its standard.

This rule prevents substitution across domains: the fingerprint of a target does not silently become that of evidence, and reserialization changes the cryptographic identity of the bytes involved.

4. Authenticate a statement with DSSE and in-toto

Sawbill uses software supply chain standards to transport authenticated attestations:

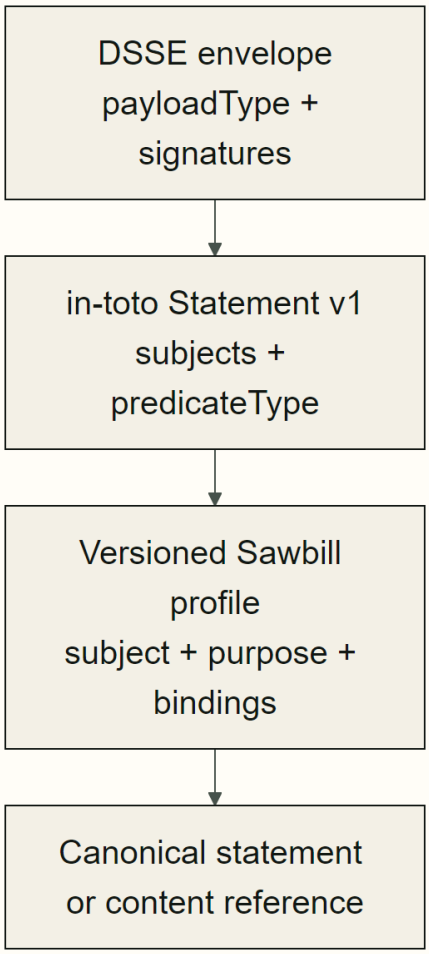

Figure 4 - Sawbill

DSSE authenticates the payload type and bytes using a defined signature construction. The in-toto Attestation Framework binds a predicate to one or more subjects identified by digest. Sawbill applies a versioned governance profile to carry the subject, purpose, and bindings needed for the decision.

Responsibilities remain clearly separated:

- DSSE provides the signature envelope;
- in-toto provides the attestation structure;
- the Sawbill profile defines the fields needed for a record type;
- Sawbill policy decides whether the signer, subject, and use are admitted.

A fresh envelope does not repair an invalid native artifact. A malformed external report or expired signature remains invalid even if Sawbill later wraps and signs it.

5. A signature does not say the claim is true

Cryptography establishes precise facts: bytes were signed with a key matching the verified profile, in a given trust context. It does not prove that the signed content is complete, understood, applicable, or true.

Cryptographic fact	Establishes	Does not establish
Valid digest	Exact identity of bytes under a codec	Origin, truth, or authority

Cryptographic fact	Establishes	Does not establish
Valid signature	Authenticity of covered bytes under a key	Signer's right or payload truth
Trusted credential	Acceptance of the credential for a use	Business permission
Timestamp	Existence of data before a time under the TSA model	Semantic accuracy of the content
Inclusion proof	Presence of a leaf in a checkpoint	Truth, completeness, or acceptance

Sawbill therefore separates cryptographic verification from semantic validation.

6. Verify mechanically, validate semantically

A deterministic verification has a closed predicate: recalculate a digest, validate a signature, compare bytes, or evaluate a schema.

Semantic validation examines a claim without a complete oracle: adequacy of an architecture, realization of an intent, quality of an explanation, or applicability of a requirement. It must remain:

- attributed to its validator;
- bound to the sources and counter-evidence examined;
- limited by a method and scope;
- contestable and reevaluable;
- independent along the dimensions required by the profile.

A second call to the same model is not necessarily independent validation. Sawbill can examine principal, organization, model, tool, credential, runtime, and control domain to establish the level of independence actually demonstrated.

7. Admit evidence for one use

After verification and validation, Sawbill may admit evidence usable for a precise predicate. Its state is not merely positive or negative:

- admitted;
- rejected;
- obsolete;
- revoked;
- replaced;
- unknown;
- blocked.

Historical evidence is never rewritten. A policy change, revocation, new counter-evidence, or new version adds an invalidation or reevaluation record.

This property prevents the past from changing silently while requiring consumers to check current usability.

8. Gates and acceptance remain two decisions

A gate evaluates a complete vector of controls. If a mandatory control is missing, evidence has expired, or its polarity is unknown, the gate cannot become favorable.

Acceptance then occurs as a governance decision: the competent authority accepts or rejects an exact subject based on exact gates and evidence.

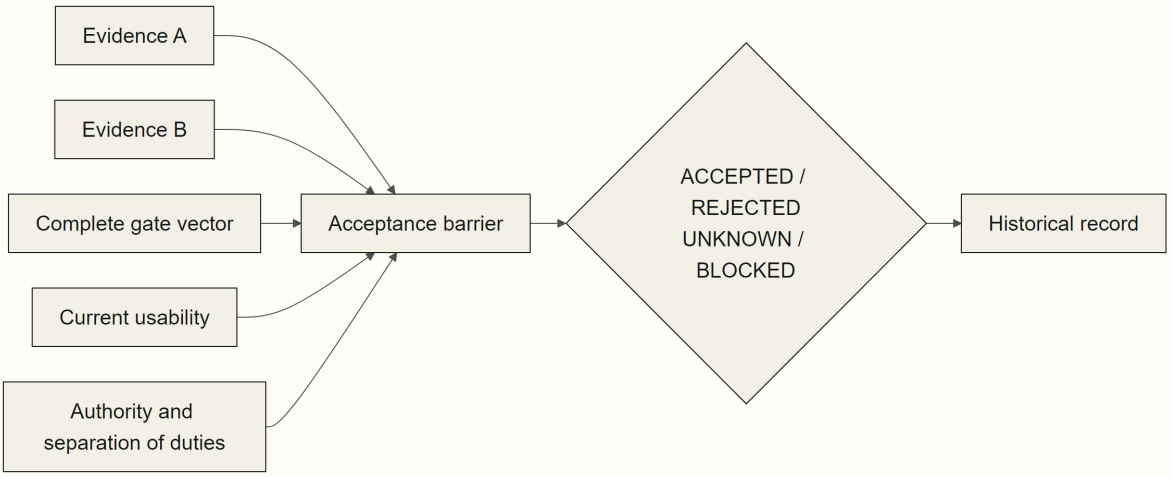

Figure 5 - Sawbill

An ACCEPTED decision is not permission to merge or deploy. The external effect still traverses its own authority, admission, and control path.

9. An append-only ledger with bounded guarantees

The Sawbill ledger retains redacted observations and their causal relationships. It does not become a second decision engine. A ledger entry creates neither evidence, a gate, nor acceptance.

To detect modifications and provide inclusion proofs, the ledger may use a Merkle structure and signed checkpoints:

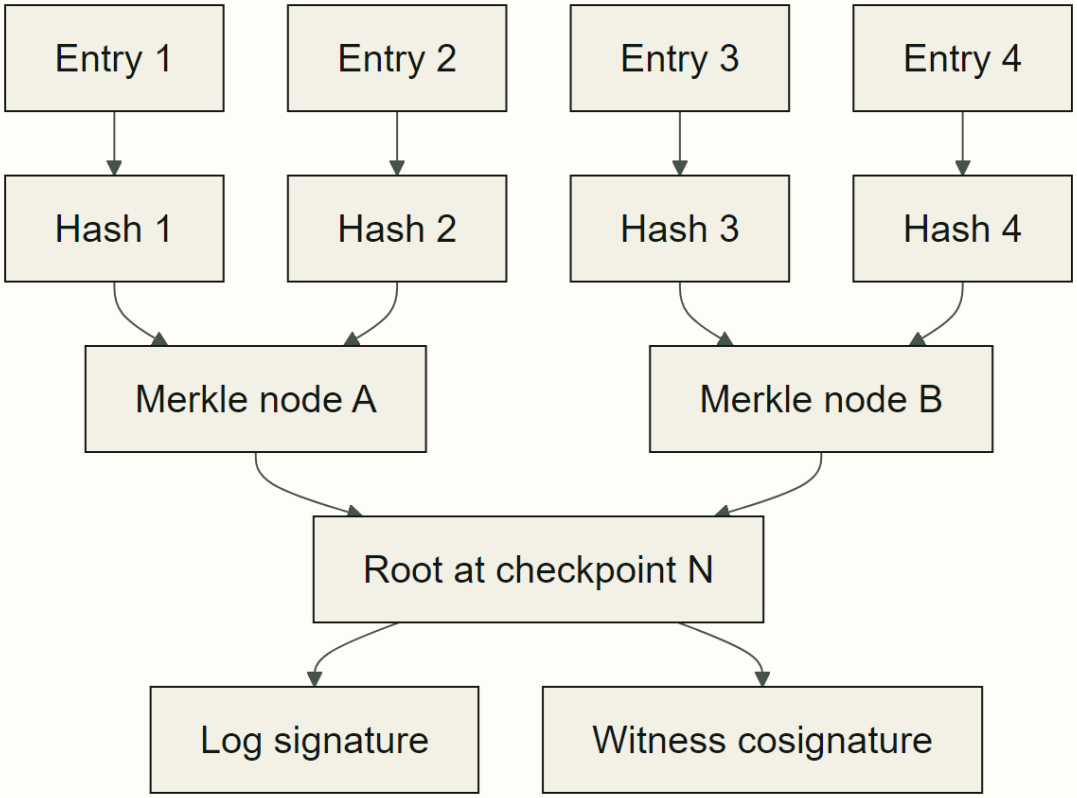

Figure 6 - Sawbill

RFC 9162 describes Certificate Transparency v2 and auditable append-only logs. The C2SP specifications tlog-tiles and tlog-witness provide open formats and protocols for checkpoints, proofs, and witnesses. Sawbill bounds their use by an identified origin, declared coverage, and explicit key policy.

10. What transparency actually proves

Ledger fact	Can support	Does not prove
Entry retained at a given sequence	Bounded retention and order	Truth of the content
Inclusion in a checkpoint	Presence of a leaf in that tree	Capture of every event in the world
Consistency between checkpoints	Coherent extension of a prefix for an origin	Global non-equivocation across all views
Cosigned checkpoint	A named witness saw the checkpoint	Independence when witness and log share control
Exported checkpoint	Possibility of later comparison	Continuity before the exported anchor

A local hash-chained journal can detect internal modifications. Without an external checkpoint, it does not by itself prove that the end of history was not deleted. Sawbill exposes that difference instead of using "immutable" as an absolute guarantee.

11. Freshness, revocation, and anti-replay

A historically valid assertion may no longer be usable. Sawbill binds verification to:

- a time or cutoff;
- a trust snapshot;
- a revocation state;
- an anti-replay context;
- a subject, attempt, and purpose;
- a freshness period defined by the profile.

Protected reuse binds the assertion, signer, and purpose to a determined context. Changing attempt or resource requires a new context; copying the same signature is not enough.

To bind data to time, a profile can use the Time-Stamp Protocol in RFC 3161, updated by RFC 5816. The timestamp remains bounded proof of existence, not a verdict on the content.

12. Confidentiality and key custody

Useful evidence must not become a durable leak. Sawbill classifies and redacts data before persistence, indexing, or ledger entry. Secrets, private keys, bearer tokens, and raw personal data are not added to an attestation "to make auditing easier."

Keys remain behind signing or custody ports. Depending on the profile, Sawbill can integrate with standard interfaces such as PKCS #11 v3.2 or KMIP v2.1. Envelope encryption separates the data key from the wrapping key and binds the ciphertext to its scope through authenticated data.

Encryption proves neither authority, residency, nor destruction of every copy. Those claims require their own observations and evidence.

Scenario: reconstruct acceptance of the data export

For the export capability, an auditor wants to know why the version was accepted.

Sawbill can reconstruct:

1. the exact identities of target and realization;
2. observations received from tests, scanners, and environments;
3. the precise claim supported by each submission;
4. cryptographic verification results;
5. validations and their demonstrated independence;
6. evidence that was admitted, rejected, or became obsolete;
7. the complete gate vector used;
8. the authority and cutoff of the acceptance decision;
9. ledger entries and their causal relationships;
10. receipts and observations from the subsequent external mutation.

If the residency configuration was inaccessible, the audit shows `UNKNOWN`. If evidence was revoked after acceptance, history preserves the initial verdict and the later invalidation. Neither is erased.

Enterprise value

Auditability reconstructed, not narrated

The audit starts from content identities, relationships, and attributed decisions rather than a chronology of screenshots.

Portable evidence

DSSE, in-toto, SLSA, and transparency formats make verification by independent tools and parties easier without locking the entire chain into a proprietary format.

Visible limits

The distinction among signature, validation, evidence, gate, and acceptance reduces overbroad conclusions and false confidence in conformance.

History compatible with change

Rotation, revocation, correction, and reevaluation add events. They do not rewrite the past or silently preserve trust that has become obsolete.

Official references

Source	External status	Contribution used	Limit retained by Sawbill
RFC 8785 - JSON Canonicalization Scheme	Informational RFC	Canonical JSON representation for hashing and signature	Codec and domain remain explicitly profiled
DSSE v1	Maintained open specification	Signature envelope and preimage with authenticated type	DSSE provides neither key management nor trust policy
in-toto Attestation Framework v1.2	Current attestation framework version	Binding between authenticated subjects and predicates	The Sawbill predicate carries product-specific governance semantics
SLSA v1.2	Published specification version	Provenance and software build and release chain levels	Build provenance does not replace deliverable acceptance
RFC 3161 - Time-Stamp Protocol	IETF Proposed Standard, updated by RFC 5816	Proof that data existed before a time	Timestamping does not prove content truth
RFC 9162 - Certificate Transparency v2	Experimental RFC, obsoletes RFC 6962	Auditable Merkle log constructions	The Sawbill profile bounds origin, coverage, and guarantee scope
C2SP tlog-tiles v0.1.0	Versioned C2SP specification	Static distribution of checkpoints, entries, and Merkle tiles	Log transport does not decide evidence
C2SP tlog-witness v1.0.0	Versioned C2SP specification	Checkpoint cosigning after consistency verification	Independence depends on actual witness control
PKCS #11 v3.2	OASIS Standard, June 3, 2026	Interface to tokens and cryptographic modules	The custody interface does not define Sawbill policies
KMIP v2.1	OASIS Standard, December 14, 2020	Key management interoperability	Key management does not become an authority decision

These mechanisms are adopted within their own scope. Sawbill does not reinvent cryptographic primitives or turn a tool's popularity into an automatic guarantee.