SERENITYVAULT PROTOCOL · SERIES 8

A verifiable pipeline: classify, decide, prove, audit

A technical overview of the protocol, its components, its non-negotiable governance rules, and the exact state of its implementation.

Technical white paper · Series 8 · Current stage: ALPHA

ALPHA This document describes a test-only software implementation (ALPHA). No production cryptography, real keys, or customer data are involved. The protocol has not undergone hardware, field, legal, institutional, or regulatory validation, nor any independent external audit.

Purpose of the protocol

SerenityVault Protocol defines a four-stage pipeline — classify, decide, prove, audit — in which each stage is owned by a distinct component, and no component can bypass the one before or after it.

The goal is not to make a decision "infallible," but to guarantee that every sensitive decision produces, at the moment it is made, a trail that can be verified independently of the implementation that generated it. That last point — independent verifiability — is what distinguishes a governance architecture from a mere logging system.

Protocol layers

TCAC — sealed, immutable classification

Every action entering the system is first classified against a fixed five-category schema (ST, SN, CR, CS, INT). The schema is versioned: any change produces a new version and a new cryptographic hash, so no past classification can be silently reinterpreted later.

ADÈLE — deterministic decision engine

For every classified action, ADÈLE renders one of three testable outcomes: `ALLOW`, `DENY`, or `ESCALATE`. No critical action can bypass the engine. It produces a structured justification and includes a contradiction scanner that detects inconsistencies across decisions. When ambiguous, denial (`DENY`) is the default.

METATRON — sealed evidence format

Every piece of evidence tied to a decision is an object sealed with a SHA3-512 hash. No evidence can bypass METATRON: any alteration, however small, invalidates the seal in a detectable way.

Chained journal and Merkle root

Decisions and evidence accumulate in an append-only, hash-chained journal. The protocol computes a deterministic, domain-separated SHA3-512 Merkle root over journal snapshots. Altering any past entry is detectable by recomputing the chain and the root.

SAK v2 — multi-profile reports with redaction

The SAK report can be generated under three distinct profiles — auditor, tribunal, regulator — each with its own redaction rules for sensitive information, drawn from the same source journal.

Independent Rust verifier

A second implementation, written in Rust and independent of the reference Python implementation, verifies evidence and journals. It is tested against a set of parity fixtures to guarantee exact agreement with the reference implementation — allowing a third party to verify evidence without trusting a single implementation. A precompiled 64-bit Windows executable is provided for offline verification without installing a toolchain.

SVP-X API — Evidence / Audit

A sandboxed API, described by an OpenAPI 3.1 contract, exposes evidence and audit verification. It separates verifier and auditor permissions, limits request size, and exposes no Tier 1 routes or operations.

Némésis — supervised forensic compilation

Némésis operates strictly at Tier 2, under supervision. Its function is to compile facts that have already been sealed, under specification `ACT-AUD-02`; it has no direct access to Tier 1 and renders no judgment on innocence, guilt, or truth. Its implementation cannot go beyond a stub until the machine-readable proof layer (A3) and SAK v2 (A4) are fully functional — a deliberate dependency, not an oversight.

Non-negotiable governance principles

01	No real secrets — keys, IP addresses, credentials, customer data, vault contents — ever enter the system; only test placeholders are used.
02	Every critical action must produce a testable decision path: `ALLOW`, `DENY`, or `ESCALATE`.
03	No action may bypass the ADÈLE rule engine.
04	No evidence may bypass METATRON.
05	No Tier 2 component accesses Tier 1 directly.
06	Némésis is Tier 2, strictly forensic; it never judges.

07 Every system output explicitly distinguishes its maturity stage: ALPHA, Beta, Release Candidate, or Production-Proven.

08 Any change to the TCAC schema creates a new version and a new hash.

Verification discipline

The software roadmap progressed under growing test coverage with no reported regression at any stage: 14 Python tests at the initial vertical slice, 56 Python tests by delivery covering PR-001 through PR-012, complemented by a Rust test suite and seven parity fixture sets confirming agreement between the two implementations. The Rust toolchain was introduced only after explicit authorization, consistent with deliberate change governance rather than silent drift. The full suite runs offline, with no keys, secrets, or network access.

Roadmap status (PR-001 through PR-012)

STEP	SCOPE	STATUS
PR-001	Repository skeleton	Complete (ALPHA)
PR-002	Sealed, immutable TCAC parser	Complete for the test fixture
PR-003	Minimal ADÈLE rule engine	Complete
PR-004	METATRON evidence format	Complete for test integrity hashing
PR-005	Append-only journal	Complete
PR-006	Merkle root and simple SAK report	Complete
PR-007	Independent Rust verifier	Complete
PR-008	SVP-X Evidence/Audit sandbox API	Complete
PR-009	ADÈLE explainability and contradiction scanner	Complete
PR-010	SAK v2 profiles and redaction	Complete
PR-011	Némésis specification and supervised mock	Complete
PR-012	Reproducible Longueuil demonstration	Complete

Explicit limits of the current state

- No hardware, field, legal, institutional, or regulatory validation has been conducted.

- No independent external security audit has been performed on this codebase.
- Signatures remain test placeholders; no production cryptography is used.
- Némésis does not go beyond a supervised mock stage.
- This ALPHA software delivery is neither a certification nor a validation of real-world deployment.

A commitment on language

The protocol commits to never using the terms `irrefutable`, `absolute proof`, `unhackable`, or `Production-Proven` to describe its present state. This commitment is one of the project's non-negotiable rules, alongside the Tier 1 / Tier 2 separation.

Contribute or evaluate the protocol

The specification, governance principles, and detailed progress status are documented to support independent technical review. For architecture questions or to propose an external review, contact the SerenityVault team.

`info@serenityvault.org · protocole.org`

SerenityVault Protocol, Series 8 — Technical document. Describes a software architecture at ALPHA stage; it is neither a final specification nor a security certification.