

SAGE — Deterministic Governance

WISP decides who. SAGE decides whether — and proves which logic decided.

DOCUMENT	SF-WP-2026-003 · Series SF-WP (Technology Whitepapers)	STATUS	Published · open for refutation
VERSION	r4 · September 2, 2026	AUTHOR	Arc · The Sovereignty Foundation
SYSTEM	SAGE — System Action Governance Engine	ANCHOR	U.S. Appl. 19/670,101, patent pending · §15
NOTATION	Fixed-point values are 32-bit Q16.16 (16 integer, 16 fractional bits; 65,536 represents 1.0) unless stated; hashes are BLAKE3-256 in the preferred embodiment.		
REV	DATE	CHANGES	
r1–r2	2026-09-02	Initial publication; expanded architecture summary.	
r3	2026-09-02	Document form: front matter, contents, appendices, print/PDF.	
r4	2026-09-02	Spec-grade depth: signal lanes and slot assignments, wire-format offset tables, sealed policy/configuration schemas, the ten-step evaluation procedure, worked Q16.16 coherence example, minting and gauntlet procedures with failure modes, ledger record schema and forensic re-verification, deployment topologies, function catalog.	

Abstract. Most governance systems can tell you what they decided. Almost none can prove how. SAGE is built on a structural refusal: the component that decides what is allowed is never the component that signs the authorization, never the component that enforces dispatch, and never the component that records the audit. Four roles — governance evaluator, cryptographic minter, enforcement gate, append-only audit ledger — operate in separate trust boundaries and communicate only through self-proving artifacts of fixed wire format: a 170-byte evidence pack, a 184-byte decision result, a 350-byte permit (§2, Appendix A). Evidence is organized into eight signal lanes of thirty-two Q16.16 channels (§3). Coherence is evaluated with per-signal allowances, a zero-drift-required mask, and Schmitt-trigger hysteresis, demonstrated here on a worked numerical example (§4). The evaluator is a pure function over evidence, sealed policy, and sealed configuration — no I/O, no clock reads, no randomness — producing bit-identical results across platforms, with every input content-addressed in the output (§5). Approval is the only state a minter will sign (§6–7); enforcement re-verifies everything through a five-stage fail-closed gauntlet (§8); the ledger chains every event and supports forensic re-verification of any historical decision by byte-equality, without the original hardware (§9). A semantic firewall raises

the trust required for approval as signal quality degrades (§10); a four-mode hierarchy spans non-blocking telemetry to sealed-binary execution (§11); topology options run from co-located processes to air-gapped sovereign deployments (§12). The architecture is implemented — six crates, 4,464 lines, 151 passing tests (§14) — and anchored in the filed estate (§15).

CONTENTS

1	Introduction	10	The semantic firewall
2	Architecture overview	11	Operating modes
3	Signal ingestion & evidence packs	12	Deployment topology
4	Coherence evaluation	13	Application: autonomous agents
5	The governance evaluator	14	Implementation status
6	Authorization states	15	Lineage, anchors, related documents
7	The cryptographic minter	A	Wire-format offset tables
8	The enforcement gate	B	Operating-mode matrix
9	The audit ledger	C	Function catalog

SECTION 1

INTRODUCTION

1 Governance that cannot show its work

Ask a deployed access-control system to justify a decision from last March and you will usually receive a log line: what was decided, when, by which rule name. Whether the rule’s implementation that day matches the one now running; whether the same inputs would decide the same way twice; whether the decision path touched a random seed, a wall clock, a cache — unanswerable. Governance without reproducibility is testimony without cross-examination.

Beneath the audit failure sits an architectural one. In nearly every deployed system, the component that *decides* is also the component that *signs*, *enforces*, and *records* — four powers in one trust boundary, so one compromise forges all four at once. And deployed systems tend to operate at a single rigidity level: a system built for high-throughput evaluation with replaceable strategies cannot transition into sealed deterministic execution for forensic audit, while a system built sealed cannot loosen into observation-only operation for tuning. The configurable surface spanning that spectrum — one codebase, from telemetry to lockdown — is the gap this architecture occupies.

For human-scale systems these failures are embarrassing. For systems now delegating actions to autonomous agents, they are disqualifying. This document describes SAGE in the order a request travels it: how evidence is constructed (§3), how coherence is judged (§4), how the decision is computed (§5) and

expressed (§6), how it becomes signed authority (§7), how authority is enforced (§8), how everything is recorded and later proven (§9), how the system behaves under adversarial signal degradation (§10), and how the whole pipeline is deployed (§11–12).

2 Four roles, four boundaries, three artifacts

SAGE’s central design decision is that authority never travels whole. The *governance evaluator* holds state and produces decisions (§5). A *cryptographic minter* — KNOX, in the preferred embodiment — accepts only positive decisions and produces signed permits (§7). An *enforcement gate* — AEGIS, in the preferred embodiment — independently verifies every permit and is structurally fail-closed without one (§8). An *append-only audit ledger* records every decision and every verification result (§9). Each operates in its own trust boundary (Figure 1; realizations in §12). No component trusts another component’s authorization without independent cryptographic verification of the artifact it received; the minter holds no access to sealed policy, the gate holds no access to signing keys, and compromise of any single component is structurally insufficient to forge what the other three produce. The interesting attacks against SAGE require four coordinated failures; the boring ones die at the first boundary they touch.

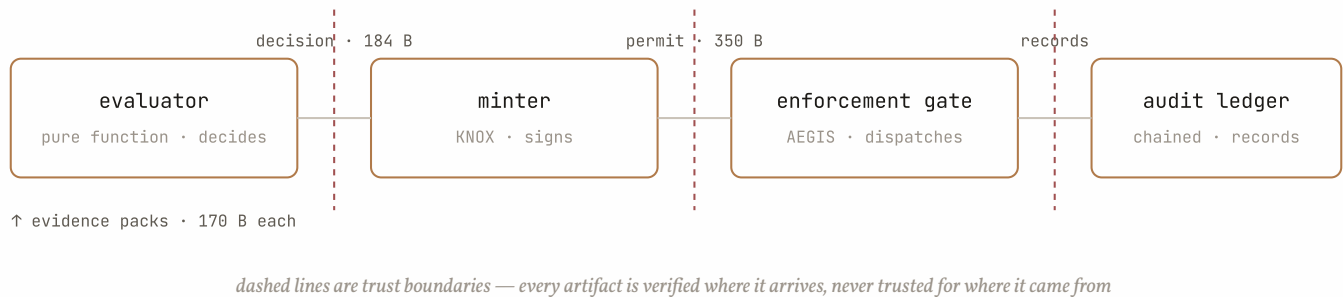


Figure 1 — The pipeline. Decide, sign, enforce, record: four powers that never share a boundary.

What crosses the boundaries is deliberately dumb: fixed-format, fixed-length artifacts that carry their own proof — a 170-byte evidence pack, a 184-byte decision result, a 350-byte self-proving permit, specified byte-by-byte in Appendix A. Fixed formats are a governance property, not a serialization preference: a parser that cannot be surprised cannot be exploited by surprise, and an artifact whose every field has a known offset can be verified by a component that shares no code with the one that produced it. Partial compositions are supported — a standalone evaluator, evaluator-plus-coherence, or the full four-component pipeline — selected per deployment; this document describes the full composition.

3 Signal ingestion and evidence packs

3.1 Lanes

Evidence is organized into *lanes* — eight in the preferred embodiment — each carrying thirty-two fixed-point signal values. Lanes partition evidence by domain, so that policy can weight, mask, or require them independently:

LANE	IDENTIFIER	DOMAIN	SIGNALS CARRIED
0	LANE_FILE	File-system operations	read, write, list, search, and similar
1	LANE_SHELL	Process and shell execution	command execution and similar
2	LANE_CODE_MOD	Code modification	edit, create, apply-diff and similar
3	LANE_SYSTEM	System / catch-all	operations not falling within prior lanes
4	LANE_NARRATIVE	Narrative semantic features	projections from upstream language-model embeddings
5	LANE_TEMPORAL	Temporal features	projections from upstream temporal embeddings
6	LANE_SPATIAL	Spatial features	projections from location and spatial coordinates
7	LANE_MODAL	Modal and role features	projections from sensor and behavioral classification

Table 1 — Lane assignment, preferred embodiment for tool-call governance. Lane count varies per deployment.

Within an evidence-pack sequence, lane identifiers form a strictly ascending sequence with no duplicates. Each lane is hashed independently — a present lane under a LANE_PRESENT tag, an absent lane under a LANE_MISSING tag bound to the current chain head — and the lane hashes combine into an aggregated evidence hash. An absent lane is therefore a *recorded* absence: evidence that is missing cannot be silently missing.

3.2 Signal slots

Within each lane, the first eight slots carry a canonical assignment; slots 8–31 carry deployment-specific extensions:

SLOT	SIGNAL PROPERTY
0	Action reversibility
1	Action blast-radius
2	Path scope
3	Target-resource sensitivity
4	Operation type
5	Action change-size
6	Action rate over sliding temporal window
7	Coherence score from prior coherence evaluation

Table 2 — Per-lane signal channel assignment, slots 0–7.

Evidence-pack construction is itself a pure function of its inputs — the evaluation timestamp is supplied as a parameter rather than read from a clock at construction time, preserving determinism from the first byte of the pipeline to the last.

DOMAIN EXAMPLE — AGENT TOOL CALLS

A tool-call governance deployment maps each request’s category to its lane (file, shell, code-modification, system), extracts the eight core properties into slots 0–7, populates the extension slots, and emits the 170-byte pack to the pipeline. An agent deleting ten thousand records and an agent reading one differ *in the evidence* — blast-radius, change-size, reversibility — before judgment begins.

4 Coherence with hysteresis: evaluation that does not flap

Coherence evaluation detects drift between current and prior evidence while absorbing the noise that makes naive drift detection useless. It operates per-signal, in Q16.16 fixed-point with 64-bit overflow-safe accumulators, over four sealed configuration structures:

STRUCTURE	SIZE	CONTENTS
Coherence configuration master	105 B	wire version, coherence threshold (Q16.16), lane masks, sealed-config hash references
Zero-drift-required mask	256 B	8 lanes $\times$ 32 signals $\times$ 1 bit — per-signal designation
Per-signal drift weights	1,024 B	8 $\times$ 32 $\times$ Q16.16 — accumulation weights
Per-signal drift allowances	1,024 B	8 $\times$ 32 $\times$ Q16.16 — hysteresis-band absorption thresholds

Table 3 — Coherence configuration structures, preferred embodiment.

4.1 The algorithm

1. **Raw drift.** For each enabled lane and signal, compute the absolute difference between the current and prior signal values.
2. **Allowance subtraction.** Saturating-subtract the per-signal allowance: drift at or below the allowance is absorbed entirely; only the excess survives as *effective drift*. Allowances are sized — in some embodiments from an estimated projection residual variance — to absorb a configurable percentile of projection-boundary oscillation: when a high-dimensional embedding is projected into fixed-point governance channels, values near a projection boundary oscillate discretely without the underlying state moving. The allowance distinguishes a signal that *moved* from a signal that is merely *ringing*.
3. **Zero-drift check.** If a signal is designated zero-drift-required and its effective drift is nonzero, evaluation terminates immediately with `REQUIRED_DRIFT_EXCEEDED`, identifying the offending lane and signal. No accumulation, no threshold comparison — the aggregate can never average away the one signal that must not move.
4. **Weighted accumulation.** Surviving effective drifts multiply by per-signal weights and accumulate (saturating arithmetic throughout) into a coherence score.
5. **Threshold comparison.** The score compares against the configured coherence threshold: above fails with `COHERENCE_THRESHOLD_EXCEEDED`; at or below passes, and the score itself feeds slot 7 of the next evidence cycle.

4.2 Worked numerical example

Three signals; `Q16_ONE` = 65,536 represents 1.0; coherence threshold 0.5 (32,768):

SIGNAL	ALLOWANCE	WEIGHT	ZERO-DRIFT-REQUIRED
Location	0.10 (6,554)	0.5 (32,768)	No
Biometric	0.05 (3,277)	0.3 (19,661)	Yes
Intent	0.20 (13,107)	0.2 (13,107)	No

Table 4 — Example configuration.

Passing scenario — projection noise absorbed. The subject travels (location moves hard) while biometric and intent jitter within their allowances:

SIGNAL	PRIOR	CURRENT	RAW DRIFT	EFFECTIVE DRIFT	WEIGHTED
Location	0.95	0.40	0.55	0.45	0.225 (14,745)
Biometric	0.98	0.97	0.01	0 (saturated)	0 · z-d-r pass
Intent	0.70	0.60	0.10	0 (saturated)	0

Table 5 — Score 14,745 ($0.225 \leq \text{threshold } 32,768 \rightarrow$ **PASS**. The one-centimeter biometric jitter cost nothing; the genuine relocation was weighed and found acceptable.

Failing scenario — the unforgiving lane. A credential-stuffing pattern: everything drifts, including the biometric:

SIGNAL	PRIOR	CURRENT	RAW DRIFT	EFFECTIVE DRIFT	OUTCOME
Location	0.95	0.10	0.85	0.75	n/a
Biometric	0.98	0.65	0.33	0.28	FAIL — zero-drift-required
Intent	0.70	0.30	0.40	0.20	not reached

Table 6 — Evaluation terminates at step 3 with *REQUIRED_DRIFT_EXCEEDED*; accumulation and threshold comparison never run.

4.3 The Schmitt trigger

Downstream of the score, a Schmitt-trigger hysteresis state machine evaluates coherence against two thresholds — a strict threshold to *gain* coherent status, a relaxed threshold to *lose* it — holding prior state inside the dead zone between them. A score wandering in the dead zone changes nothing; only a hard crossing changes state. The same conditioning discipline appears throughout the Foundation’s temporal

work — the Kronos paper derives it for time — and its effect here is blunt: multi-signal governance that does not flap under noise, and cannot be walked across a threshold by an adversary supplying jitter.

5 A decision as a pure function

5.1 The contract

The evaluator’s purity is a five-part structural property: (i) output determined exclusively by input arguments; (ii) no input/output operations during evaluation; (iii) no clock reads — the evaluation timestamp is a parameter; (iv) no side effects beyond the returned result; (v) no source of randomness. Identical inputs therefore produce a bit-identical 184-byte decision result across executions, hardware platforms, and sessions — forensic-grade reproducibility, verified by byte equality (§9.3). Arithmetic is Q16.I6 with 64-bit accumulators in the preferred embodiment; other Q-formats are contemplated per deployment.

5.2 Sealed policy and sealed configuration

The rules the evaluator applies arrive as two content-addressed, immutable objects. *Sealing* means the object’s hash is computed at sealing time and used thereafter as its identifier; no field modification is permitted afterward — an update is a new object with a new hash. Every decision records which policy and which configuration judged it, by hash.

SEALED POLICY FIELD	TYPE	PURPOSE
Subject-to-role mapping	ordered (glob, role) list	first-match-wins role resolution against subject identifiers
Role profiles	role → profile map	per-role parameter sets
Lane masks	u8 bitmask per role	which evidence lanes a role may submit
Weight-section identifiers	string per role	pointer into sealed-configuration weight sections
Time-to-live caps	u32 seconds per role	maximum permit validity per role
Policy hash	32-byte BLAKE3	content address of the signed bytes

Table 7 — Sealed policy schema (preferred embodiment). Extensions include per-role effect-type permission grids, temporal-window constraints, environmental gating, and escalation routing.

SEALED CONFIGURATION FIELD	TYPE	PURPOSE
Per-role per-lane signal weights	Q16.16 grid (8×32×N)	trust-score weighting
Per-role global lane weights	Q16.16 array (8×N)	per-lane multiplier
Allow / escalate thresholds	Q16.16 per role	the two decision boundaries of §5.3 step 8
Time-to-live caps	u32 seconds per role	per-role permit validity
Hysteresis allowances · zero-drift mask · coherence threshold	1,024 B · 256 B · Q16.16	the §4 structures
Configuration hash	32-byte BLAKE3	content address of the signed bytes

Table 8 — Sealed configuration schema (preferred embodiment).

5.3 The evaluation procedure

One evaluation runs ten ordered steps:

1. **Chain-head validation** — the current ledger head is a required input; an evaluator cannot run detached from the audit chain.
2. **Lane-structure validation** — ascending lane order, no duplicates, and every submitted lane permitted by the subject’s role lane-mask.
3. **Per-lane hashing** — present lanes hashed under the present tag; absent lanes hashed under the missing tag bound to the chain head (§3.1).
4. **Evidence aggregation** — lane hashes combine into the aggregated evidence hash that the decision will carry.
5. **Role resolution** — first-match glob resolution against the sealed policy; the role selects its weight section in the sealed configuration.
6. **Coherence evaluation** — the §4 algorithm, if enabled; failure returns an immediate Deny carrying REASON_COHERENCE_FAILED and all input hashes.
7. **Trust-score computation** — per-signal weighted accumulation per lane, then per-lane global weighting, in a 64-bit saturating accumulator.
8. **Threshold application** — score $\geq$ allow-threshold $\rightarrow$ Approve; $\geq$ escalate-threshold $\rightarrow$ Escalate; below $\rightarrow$ Deny.
9. **TTL derivation** — the permit lifetime cap is the minimum of the global and role-derived caps.
10. **Result construction** — the 184-byte decision result (Appendix A): decision, reason mask, TTL, and the content hashes of evidence, policy, and configuration.

A sealed policy maps practitioner identifiers to clinical roles — attending, resident, nurse, pharmacist — and a sealed configuration carries per-role weights tuned to clinical-context signals. Each record-access decision emits a 184-byte result binding the authorization to the practitioner evidence, the clinical policy in effect, and the configuration in effect — at decision time, by hash.

6 Four states, default deny

Approve · Deny · Escalate · Delay

The pipeline's output is a finite enumeration, not a score. *Approve* permits downstream execution — and is the only state the minter will sign. *Deny* prevents it. *Escalate* defers the decision to external review: a person, structurally, with the deferral itself recorded — the direct descendant of the May 2025 whitepaper's position that no system should operate without human oversight where meaningful consequences exist. *Delay* imposes a temporal cooldown resolving on a subsequent evaluation cycle — the state for evidence that is insufficient *now*. The default is Deny, enforced structurally rather than by convention: absent an affirmative result there is nothing to sign, nothing to verify, nothing that executes. Downstream components face a bounded set of four cases; the open-ended judgment lives inside the evaluator, where it can be replayed.

7 The minter: authority as a signed artifact

The minter — KNOX in the preferred embodiment — converts an Approve decision into a 350-byte self-proving permit: a 286-byte signed body under a 64-byte Ed25519 signature (post-quantum signature families are contemplated as alternatives). Minting runs nine ordered steps: validate that the signing key is role-bound to minting authority and active; validate the decision result's structural integrity; **refuse any decision that is not Approve**; acquire the next strictly-monotonic replay counter for the subject; derive the expiry from the decision's TTL; compute the effect hash over the canonical effect specification and the audit-pointer hash; pack the signed body — subject, counter, validity window, chain head, decision hash, effect hash, evidence hash, policy hash, configuration hash, audit pointer; sign; emit.

The minter's ignorance is a design feature: it performs no governance evaluation, interprets nothing beyond structural integrity and the decision value, and holds no access to sealed policy or configuration — it sees them only as hashes. Its signing-key material is structurally inaccessible to every other compo-

ment; in the preferred embodiment it is encrypted at rest under a Argon2id-derived key with AES-256-GCM, decrypted only into the minter’s runtime memory, and zeroized on shutdown.

DOMAIN EXAMPLE – FEDERATED MINTING

In a multi-organizational deployment, a service provider operates the minter under its own key infrastructure; consumer organizations honor the resulting permits at their own enforcement gates without trusting the provider’s governance evaluation beyond what the permit’s cryptographic content proves on its face.

8 The gate’s gauntlet

A permit arriving at the enforcement gate — AEGIS in the preferred embodiment — has already been signed by the minter, and the gate does not care. It re-verifies everything, in order, failing closed at every stage, each stage encoding a distinct refusal with a named failure:

STAGE	VERIFICATION	REFUSES	FAILURE
1	Signature over the signed body, against the key ring entry for the permit’s key identifier (revoked keys refuse)	permits not signed by active, role-bound minting authority	PermitSignatureInvalid · KeyRevoked
2	Replay counter strictly greater than the largest seen for this subject and key	a permit spending twice	ReplayDetected
3	Current time within issued-at / expires-at	expired and not-yet-valid permits	PermitExpired · PermitNotYetValid
4	Hash of the received effect specification equals the permit’s effect hash	authority harvested from one operation and spent on another	EffectHashMismatch
5	The permit’s chain head is a valid head of the audit ledger’s chain	permits whose audit lineage does not check out	ChainHeadMismatch

Table 9 — The five-stage gauntlet. Only after all five does the operation dispatch — and the dispatch itself is recorded (§9).

Deployments extend the gauntlet per threat model: proof-of-work verification for anti-denial-of-service gating, verifiable-delay-function proof of sequential work, environmental-context verification by cosine comparison against a baseline fingerprint, and biometric-continuity verification by fuzzy hash comparison. The gate holds no signing keys — only public verification keys in its own ring — and no access to sealed policy or configuration; it verifies cryptographic content, nothing else. Its posture without

a valid permit is not “deny by policy” but fail-closed by structure: there is no dispatch path that bypasses the gauntlet.

9 The ledger: provenance you can re-execute

9.1 The chain

Every stage transition in the pipeline — evaluation, minting, verification, dispatch outcome, mode transition, key rotation — appends a record to a ledger whose only supported operation is append: no modify, no delete, no reorder. Each record carries its sequence number, timestamp parameter, event type, the hash of its upstream artifact, its canonical payload, and two chain fields — the prior record’s hash and its own, computed as:

$$h_n = H(h_{n-1} \parallel \text{record}_n)$$

with H collision-resistant (BLAKE3-256 preferred). Appends commit through atomic writes before returning. Modifying any historical record breaks the linkage of every subsequent record — tampering is not prevented; it is made self-announcing.

9.2 Persistence modes

MODE	BEHAVIOR	SUITABLE FOR
Local	append-only file on the local file system	single-host; offline edge
Emit	records buffered to a downstream aggregator	distributed deployments with central audit
Both	simultaneous local append and downstream emit	defense in depth — neither channel sufficient alone

Table 10 — Persistence configuration. Tiered, replicated, and hardware-attested persistence are contemplated extensions.

9.3 Forensic re-verification

The ledger’s deepest property is what it enables months later. Because the evaluator is pure (§5.1) and every decision records its inputs by content hash, any party holding the recorded decision, the recorded evidence packs, and the sealed objects those hashes identify can re-execute the decision:

```

function reverify_decision(recorded_decision, recorded_evidence,
                           sealed_policy, sealed_config) → bool:
    require(sealed_policy.hash == recorded_decision.policy_hash)
    require(sealed_config.hash == recorded_decision.config_hash)
    recomputed = sage_evaluate(recorded_evidence, sealed_policy, sealed_config,
                                recorded_decision.subject_id,
                                recorded_decision.chain_head,
                                recorded_decision.evaluation_timestamp)
    return recomputed.bytes == recorded_decision.bytes    // byte equality

```

No original hardware, no original execution environment, no context beyond the recorded inputs and the sealed objects. The verdict is byte equality against a tamper-evident chain — cryptographic-grade proof that the recorded authorization was correctly computed under the rules in effect at decision time.

DOMAIN EXAMPLE — THE AUDIT, MONTHS LATER

A healthcare deployment is asked to prove a clinical authorization from last quarter. The audit query retrieves the recorded decision, the practitioner evidence pack, and the sealed policy and configuration identified by the decision's hashes. Re-execution produces a byte-identical result: the authorization was computed correctly, under those rules, on that evidence — provably, not testimonially.

10 The worse your signals, the more trust you need

An adversary's cheapest move against a multi-signal evaluator is not to forge good signals — it is to degrade them: jam a channel, starve a sensor, inject noise until the evaluator is judging on fumes, then slip through on whatever remains. The semantic firewall inverts the economics. Each input signal carries a trust coefficient; the coefficients aggregate into a signal-trust quality metric for the evidence as a whole; and the trust score required for an Approve decision is *elevated as a function of degraded quality* (Figure 2).

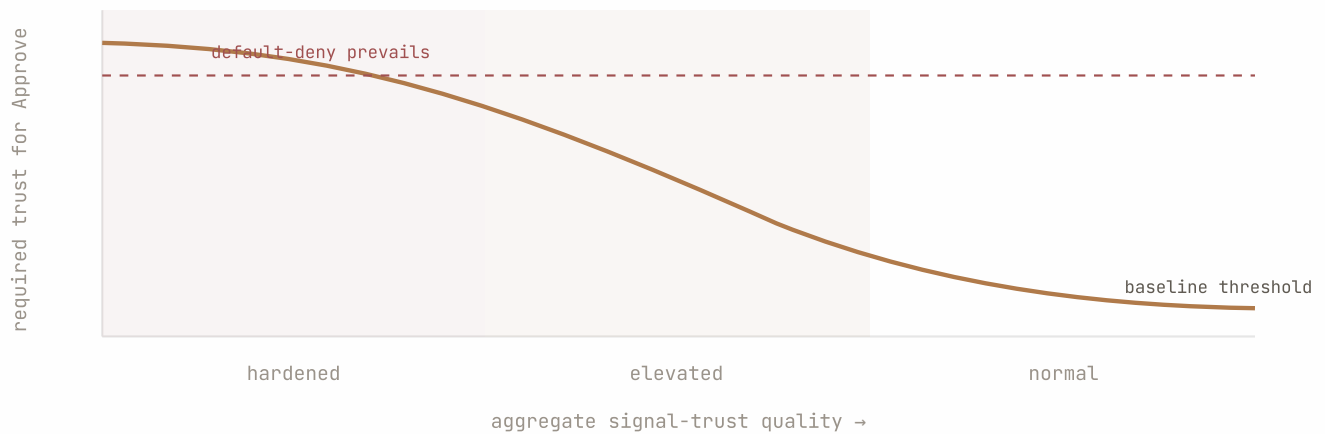


Figure 2 — Threshold elevation. Degraded evidence raises the bar; sufficiently degraded evidence closes the door.

Three regions fall out of the inverse relationship: a *normal* region where the baseline threshold applies; an *elevated* region where approval demands more trust than usual precisely because the evidence is worse than usual; and a *hardened* region where the elevated threshold sits high enough that structural default-deny prevails for substantially all inbound operations. Degrading the evaluator’s senses does not blind it — it hardens it. The attack that pays against a naive threshold produces, here, a system that has already stopped saying yes.

11 Observe $\subset$ Float $\subset$ Deterministic $\subset$ Locked

Operating posture is selected at deployment from a graduated hierarchy in which each mode imposes a strict superset of the prior mode’s constraints — and mode transitions are themselves governance events, recorded in the ledger. The constraint matrix is Appendix B.

Observe computes and records decisions while the gate passes all operations through without denial — the tuning mode: baseline establishment, signal-distribution collection, false-positive evaluation of candidate sealed configurations, coherence-behavior measurement, training-data construction; a glass-cockpit posture in which governance outputs are visible while execution continues uninterrupted. **Float** enforces per-decision, permits floating-point pre-processing ahead of the fixed-point governance computation, and allows free hot-swap — development and low-stakes inference. **Deterministic** requires fixed-point arithmetic throughout, blocks floating-point pipeline stages, and retains governance-validated hot-swap — the auditable-replay mode for regulated environments. **Locked** adds mandatory verification of a binary integrity seal over the evaluator’s own executable image, mandatory ledger record-

ing for every decision, and a structural block on hot-swap — sovereign and high-assurance deployments, where even the evaluator’s binary must prove it is the binary that was sealed.

The hierarchy exists because real systems need to span it: the same deployment tunes in Observe, graduates to Float, hardens to Deterministic for its audit season, and seals to Locked for sovereign operation — one codebase, one policy language, four postures. In candor, and per the Foundation’s published register of known deltas: the reference implementation currently defaults to Float, and the AGI Standard’s conformance cells require Deterministic — the delta is disclosed and its closure tracked in the open.

12 Realizing the boundaries

The four trust boundaries are architectural; how they are realized is a deployment decision:

TOPOLOGY	REALIZATION	SUITABLE CONTEXT
Co-located process separation	distinct OS processes, cryptographically authenticated IPC	single host; embedded
Container separation	distinct containers, authenticated sockets	cloud governance services
Virtual-machine separation	distinct VMs, authenticated protocols	high isolation
Cross-host separation	distinct hosts across regions	distributed deployments
Cross-cloud separation	evaluator, minter, gate, and ledger on distinct cloud providers	federated multi-organizational governance
Hardware-attested separation	distinct trusted execution environments (SGX/TDX, SEV-SNP, TrustZone, confidential VMs, TPM-rooted enclaves)	high assurance; confidential computing

Table 11 — Topology options. Hybrids are contemplated.

Because the inter-component artifacts are self-proving, federated arrangements need no shared operational trust: each organization operates its components inside its own organizational, jurisdictional, or operational context, honoring only what the cryptography proves. At the far end of the spectrum, air-gapped sovereign deployments transfer permits and audit events between boundaries by physical media — the pipeline’s guarantees survive the absence of a network.

13 Governing the autonomous: tool calls as governed effects

The newest actor in any system is an AI agent asking to execute a tool call, and SAGE's answer is deliberate boredom: intercept the call before dispatch, normalize it into the lanes and slots of §3, and send it down the same pipeline as everything else. The four states then mean what they always mean; Escalate routes the genuinely ambiguous call to a person; and the ledger holds a tamper-evident record of every tool call any agent was ever permitted or refused. The method is agent-agnostic and target-agnostic — no modification to any of the four components across distinct agent implementations and tool targets.

The same domain-independence holds beyond agents. The evaluator, the arithmetic, the minter, the gate, and the ledger are domain-agnostic primitives; specialization lives entirely in the sealed policy and sealed configuration (§5.2). The same governance binary, in the same mode, may govern agent tool calls in one deployment, network packet authorization in another, robotic motion commands in a third, and clinical decision support in a fourth — distinguished only by configuration. One governance codebase, many regulated domains: this is what makes the discipline standardizable, and it is the architectural bridge to the Foundation's AGI Standard and the wider proof-gated discipline demonstrated in the reduction-to-practice report.

14 Reduction to practice

SAGE is implemented and operational in the Foundation's reference stack: six crates, 4,464 lines, **151 passing tests**. A pure-function mathematics library with zero dependencies carries the evaluation primitives (Appendix C); the trust engine and geometric firewall implement the coherence discipline behind pluggable comparison modules; the governance chain, decision ledger, and binary seal implement the provenance layer — the seal being the compile-time binary hash that Locked mode verifies; a bridge adapts identity-layer types through the temporal layer; and thirty-nine stage wrappers integrate the pipeline into the operating system's staged execution. The implementation receives live tool-call requests, computes deterministic evaluations, mints signed self-proving permits, gates execution, and chains every decision. The conformance corpus's scaffolded cells exercise the chain and mode disciplines publicly, and per-requirement test coverage is cited in the evidence footers of the AGI Standard.

15 Lineage, anchors, and related documents

The line runs unbroken from the May 2025 SAGE whitepaper — human oversight where consequences live, preserved in the Foundation archive as this document’s ancestor — through the filed architecture, to the running system, to this publication: written first, filed second, built throughout, published after. The escalation state is the 2025 principle wearing 2026 structure.

RELATED DOCUMENTS: the AGI Standard (the requirements this architecture satisfies) · the conformance corpus (the cells that test it) · the Proof-Gated Execution report (the wider discipline) · the Kronos paper (the temporal substrate) · the RITA whitepaper (the modular evaluation layer).

DISCLOSURE ANCHORS

This whitepaper describes architecture disclosed in U.S. Patent Application 19/670,101 (SAGE), **patent pending**, with written-description support across the Foundation’s filed family, including the sovOS omnibus application 64/004,108, patent pending. The identity substrate upstream is the subject of granted U.S. Patent 12,688,267 B1, whose claims recite the four-state decision discipline. Published after filing and scoped to the filed disclosures, per the Foundation’s disclosure-anchoring practice. Comments and refutations are invited.

A Wire-format offset tables

Byte-precise layouts, preferred embodiment. Other embodiments vary lane counts, signal counts, Q-formats, and metadata fields under different fixed lengths.

OFFSET	LEN	FIELD	TYPE
0	1	Wire-format version	u8
1	1	Lane identifier	u8 (0..7)
2	8	Evaluation timestamp (parameter — not read from clock)	u64 LE
10	32	Source identifier	bytes
42	128	Signal array — $32 \times \text{Q16.16}$	i32 LE $\times 32$

Table A-1 — Evidence pack, 170 bytes.

OFFSET	LEN	FIELD	TYPE
0	1	Wire-format version	u8
1	32	Subject identifier	bytes
33	32	Chain-head hash	bytes
65	1	Authorization decision	u8 enum: Deny=0 Approve=1 Escalate=2 Delay=3
66	8	Reason mask	u64 LE bitfield
74	2	Error code	u16 LE
76	4	Time-to-live (seconds)	u32 LE
80	32	Aggregated evidence hash	bytes
112	32	Policy hash	bytes
144	32	Configuration hash	bytes
176	8	Evaluation timestamp	u64 LE

Table A-2 — Decision result, 184 bytes.

OFFSET	LEN	FIELD	TYPE
0	1	Wire-format version	u8
1	1	Signature-algorithm identifier	u8 (Ed25519=1)
2	4	Key identifier	u32 LE
6	32	Subject identifier	bytes
38	8	Replay counter (monotonic per subject)	u64 LE
46	8	Issued-at	u64 LE
54	8	Expires-at	u64 LE
62	32	Chain-head hash	bytes
94	32	Decision-result hash	bytes
126	32	Effect hash	bytes
158	32	Aggregated evidence hash	bytes
190	32	Policy hash	bytes
222	32	Configuration hash	bytes
254	32	Audit-pointer hash	bytes
286	64	Signature over bytes 0–285	bytes

Table A-3 — Self-proving permit, 350 bytes (286-byte signed body + 64-byte signature).

B Operating-mode matrix

MODE	ENFORCEMENT	FLOATING-POINT	MODULE HOT-SWAP	BINARY SEAL	AUDIT RECORDING
Observe	Non-blocking pass-through	Permitted	Permitted	Not required	Optional
Float	Per-decision	Permitted (pre-processing only)	Permitted	Not required	Optional
Deterministic	Per-decision	Blocked (governance computation)	Permitted (governance-validated)	Optional	Preferred
Locked	Per-decision	Blocked	Structurally blocked	Mandatory	Mandatory

Table B-1 — Each successive mode imposes a strict superset of the prior mode’s constraints; intermediate modes are contemplated between adjacent rows. Mode transitions are ledger-recorded governance events.

C Function catalog by domain

The preferred-embodiment functions populating the modules of §3–§9, disclosed individually and in combination; deployments select subsets per requirements.

DOMAIN	PURPOSE	PREFERRED-EMBODIMENT FUNCTIONS
Vector	geometric primitives	cosine_similarity · dot_product · l2_norm · l2_distance · cosine6 · dot6 · mag6 · subtract6
Decay	time-weighted operations	exponential_decay · half_life_decay ($\lambda = \ln 2 / h$) · fused_trust
Differential geometry	trajectory-shape operations	curvature_2d · torsion_2d · curvature_1d · torsion_1d
Normalization	range and statistics	z_score_normalize · min_max · unit_normalize · welford_update
Spectral · quantization · comparison · stages	frequency features; fixed-point projection; pluggable comparison modules; governance and engine stage wrappers	per deployment — see the signature registry accompanying the omnibus filing

Table C-1 — The catalog’s domains. The temporal functions are the Kronos family, treated formally in SF-RP-2026-005.