MONTREAL.AI RESEARCH ◇ GOALOS CANONICAL SERIES

GoalOS: Proof-Carrying Autonomy

The Canonical Proof-to-Capability Operating System for Autonomous AI Work

Vincent Boucher

President, QUEBEC.AI & MONTREAL.AI

Institutional Research Monograph | Canonical Edition 1.0. Production-grade specification; external empirical validation pending.

Core Thesis

GoalOS converts an objective into a bounded Proof Mission; unsupported claims into Proof Debt; Proof Debt into custom AGI Jobs; jobs into Evidence Dockets; evidence into Chronicle decisions; Chronicle-admitted results into a Validated Skill Graph; and admitted graph state into tamper-evident Merkle commitments. Future missions improve only from scoped, replayable, validated, root-verifiable capability.

AI creates output. GoalOS creates proof.

Abstract

Frontier models make candidate work abundant, but institutions require a governed method for deciding which work deserves trust, reuse, payment, memory, propagation, or rollback authority. This paper presents GoalOS as a proof-to-capability operating system for autonomous AI work. The canonical loop is: Objective → Proof Mission → AGI Jobs → Evidence → Chronicle → Validated Skill → On-chain Graph Root → Future Mission Improvement. Unsupported claims become explicit Proof Debt. An empty-by-design job factory converts that debt into mission-specific proof contracts. Returned ProofBundles populate an Evidence Docket. Chronicle acts as a memory firewall, admitting only capabilities that satisfy proof-level-specific evidence, replay, validation, scope, risk, utility, freshness, and rollback criteria. Admitted capabilities become nodes in a Validated Skill Graph rather than entries in a prompt library. A Merkle-native commitment layer binds typed roots for skills, evidence, ProofBundles, policies, attestations, encrypted capsules, and Chronicle decisions into append-auditable graph roots. Encryption, selective disclosure, zero-knowledge policy proofs, direct-agent authorization, and an explicit local/live boundary protect private intelligence and constrain authority. The central claim is architectural and falsifiable: durable autonomous work requires proof-carrying loops that prevent persuasive but unsupported output from becoming institutional memory. This canonical edition specifies the object model, security boundary, user experience, production artifacts, conformance levels, evaluation program, and first external proof run required to test that claim.

Executive Compression

The problem: output is abundant; justified authority is scarce. **The mechanism:** Proof Debt becomes AGI Jobs; Evidence becomes Chronicle candidates; admitted skills become reusable capability. **The cryptographic boundary:** Merkle roots and ZK receipts commit proof state without exposing plaintext private intelligence. **The empirical burden:** one externally reviewed mission, one admitted skill, one verifiable graph root, and one measured improvement on a second mission.

Keywords: GoalOS; proof loops; autonomous agents; Proof Mission; Proof Debt; AGI Jobs; ProofBundles; Evidence Docket; Chronicle; Validated Skill Graph; Merkle roots; zero-knowledge proofs; ENS; governed capability improvement; proof economy.

Contents

1 Executive Thesis

The first age of generative AI made candidate work abundant. Models can draft reports, code, plans, policies, simulations, critiques, hypotheses, and tool actions at low marginal cost. The scarce resource is no longer output alone. It is justified authority: which result deserves trust, reuse, payment, propagation, memory, or rollback authority.

GoalOS is the operating layer that converts autonomous AI activity into proof-carrying institutional capability. It separates drafting from authority, action from proof, evidence from memory, and private intelligence from public commitments.

Canonical Product Promise

Set the objective. GoalOS runs until proof is done.

The canonical loop is:

Objective → Proof Mission → AGI Jobs → Evidence → Chronicle → Validated Skill → On-chain Graph Root → Future Mission Improvement

Unsupported claims become explicit Proof Debt. Proof Debt becomes custom AGI Jobs. Jobs return ProofBundles. An Evidence Docket makes proof inspectable. Chronicle decides what may become durable memory. Accepted results become Validated Skills. Merkle roots commit the admitted graph state. Future missions may inherit only scoped, fresh, Chronicle-admitted, root-verifiable capability.

Institutional Distinction

A prompt asks a model to answer. A proof loop decides what must be proven before the answer can matter. A model can answer. An agent can act. An institution must prove.

2 Reader Map

Reader	Primary question	Sections to prioritize
Executive / buyer	What is the product and why does it matter?	Executive Thesis; Canonical Loop; Production and Evaluation
Operator	How do I run a proof mission?	Proof Mission; Proof Debt; AGI Jobs; Evidence Docket
Reviewer / validator	What earns memory?	Chronicle; Validated Skill Graph; Acceptance Tests
Protocol engineer	What is committed and who may write it?	Merkle Forest; ZK Boundary; Direct-Agent Authorization
Governance / security	What prevents authority leakage?	Security Model; AEP Alignment; Claim Boundary
Researcher	What would falsify the architecture?	Evaluation Program; Proof Run 001; Baselines and Metrics

Table 1: *Reader map for the canonical edition.*

3 Evidence Status and Claim Boundary

The present system is best described as a **production-grade specification and release-candidate implementation package**. It includes formal objects, browser-local command surfaces, reference smart contracts, GitHub Action deployment packs, research papers, runbooks, schemas, and validation harnesses. It is not yet an externally validated production network.

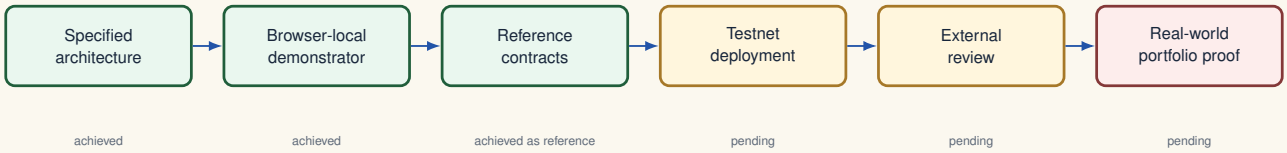


Figure 1: *Evidence maturity. The architecture and implementation package are substantial; stronger production and empirical claims remain gated by testnet deployment, independent review, and real-task evidence.*

Layer	Current evidence	Status
Doctrine and formal model	Mission OS, PROOF_LOOPS, Chronicle law, VSG object model, AEP alignment	Specified
Product demonstrator	Browser-local objective-to-root interfaces and artifact export	Implemented
Reference protocol	Solidity registries, Merkle construction, direct-agent authorization model	Reference-ready
Live network deployment	Reviewed testnet deployment, verified contract address, operational key policy	Pending
Independent review	External reviewer replay and signed verdict on a real mission	Pending
Empirical compounding	Measured improvement on a second mission under controlled baselines	Unproven

Table 2: *Evidence-status matrix. The architecture is substantial; empirical authority remains gated.*

The paper does not claim achieved AGI or ASI, empirical state of the art, legal or security certification, guaranteed economic return, live mainnet settlement, or autonomous production authority. Those claims would require real tasks, baselines, replay, external validation, safety and cost ledgers, delayed outcomes, and independent review.

4 CANONICAL CONTRIBUTIONS

This edition consolidates the architecture into twelve contributions.

1. **Proof-to-capability operating loop.** A lifecycle from objective to future-mission improvement.
2. **Proof Mission.** A bounded authority object with immutable proof level, scope, risk, validators, and done condition.
3. **Proof Debt.** A method for turning unsupported claims into explicit work requirements.
4. **Custom AGI Job factory.** An empty-by-design factory that creates only mission-specific proof contracts.
5. **ProofBundles and Evidence Dockets.** A structured proof room tying work to provenance, replay, validation, cost, risk, and claim boundaries.
6. **Chronicle memory firewall.** A policy gate that prevents unsupported output from contaminating future missions.

7. **Validated Skill Graph.** A governed graph of scoped, evidence-backed, versioned, rollbackable reusable capability.
8. **Worth-keeping policy.** A retention rule requiring expected reuse and value to exceed maintenance, drift, and governance cost.
9. **Merkle-native graph state.** Typed roots for skills, evidence, ProofBundles, policies, attestations, encrypted capsules, and Chronicle decisions.
10. **Private/public proof boundary.** Encryption, commitments, selective disclosure, and zero-knowledge policy proofs instead of plaintext private intelligence.
11. **Direct-agent authority.** A writer model based on direct `*.agent.agi.eth` control or revocable delegation.
12. **Falsifiable production program.** Conformance levels, security tests, acceptance tests, and Proof Run 001.

5 System Doctrine

GoalOS rests on six laws.

Law 1: Output Is Not Authority

Fluency, confidence, novelty, or model pedigree cannot grant memory, settlement, or future-mission influence. Output remains candidate work until it survives the required proof gates.

Law 2: Work Must Produce Proof

An AGI Job is not a todo item. It is a proof contract with a source claim, why-needed rationale, worker, validator, sentinel, deliverables, acceptance tests, blocked claims, ProofBundle return path, and rollback condition.

Law 3: Evidence Must Become Inspectable

A pile of files is not an Evidence Docket. The docket must allow a reviewer to reconstruct what was attempted, what evidence exists, what failed, what remains blocked, and what may be reused.

Law 4: Memory Is Gated

No Chronicle entry means no future-mission influence. Unsupported claims may remain visible as Proof Debt, but they may not propagate through memory.

Law 5: Private Intelligence Is Protected

Public chains coordinate proof, not plaintext private intelligence. Sensitive methods, traces, reviewer notes, customer data, and proprietary context remain encrypted, off-chain, or selectively disclosed.

Law 6: Improvement Is Earned

Exploration may guide allocation, but outcome authority is mechanical. Evidence, replay, baselines, validation, risk, persistence, scope, and rollback govern promotion.

6 CANONICAL PROOF-TO-CAPABILITY LOOP

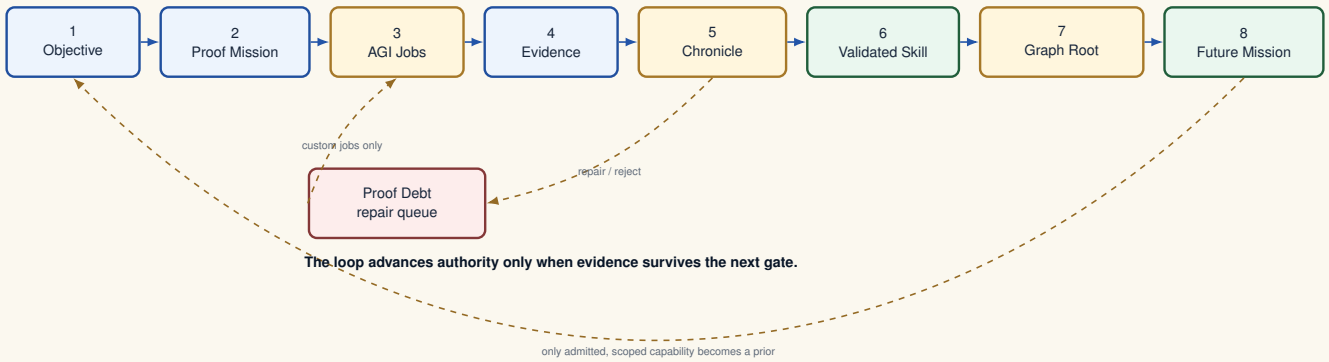


Figure 2: *Canonical proof-to-capability loop. Work becomes reusable capability only after evidence passes Chronicle and the admitted skill is bound to an auditable graph-root state.*

The user-facing loop is compact:

Objective → Proof Mission → AGI Jobs → Evidence → Chronicle → Validated Skill → On-chain Graph Root → Future Mission Improvement.

The implementation loop is explicit:

```
Objective → Mission Contract → Proof Debt → Custom AGI Jobs → ProofBundles → Evidence
Docket → Verifier Mesh → Chronicle Gate → Validated Skill Graph → Merkle Graph Root →
Future Mission Brief.
```

6.1 Formal state model

Let a run at time t have state

$$S_t = (O, M, C, D, J, P, E, V, K, G, R, F, B), \tag{1}$$

where O is the objective; M the Mission Contract; C the claim set; D Proof Debt; J the AGI Job queue; P returned ProofBundles; E the Evidence Docket; V verifier state; K Chronicle state; G the Validated Skill Graph; R the graph-root record; F the future-mission prior; and B the claim, privacy, settlement, and scope boundary.

A transition is

$$S_{t+1} = \tau(S_t, u_t, \pi_\ell, \rho_t), \tag{2}$$

where u_t is an operator or system action, π_ℓ is the selected proof-level policy, and ρ_t is a review, replay, validation, risk, or external-outcome signal.

A run is complete when

$$\mathrm{Done}(S_t) = M \wedge D_{\mathrm{resolved}} \wedge E_{\mathrm{reviewable}} \wedge K_{\mathrm{decision}} \wedge B_{\mathrm{intact}}. \tag{3}$$

Completion does not require every claim to be accepted. Repair, rejection, quarantine, or retirement can be correct decisions when the reason, evidence, and rollback state are preserved.

7 AUTHORITY STACK

Figure 3: *Authority stack. GoalOS separates desire, bounded work, proof production, evidence review, memory admission, and public commitment.*

Each layer narrows authority:

- The **Objective Layer** states the decision and desired outcome.
- The **Mission Layer** freezes scope, proof level, validators, constraints, and done conditions.
- The **Proof Layer** turns unsupported claims into custom jobs.
- The **Evidence Layer** packages returned work as inspectable proof.
- The **Memory Layer** admits only bounded reusable capability.
- The **Commitment Layer** makes admitted graph state compact, tamper-evident, and externally auditable.

No lower layer may silently bypass the gate above it. A model may propose. A worker may execute. A validator may attest. Chronicle decides memory. A registry may commit only the admitted state.

8 PROOF MISSION AND MISSION CONTRACT

A Proof Mission is the bounded institution created from the user objective. It is the first authority-bearing object: it specifies what decision the work supports, what evidence is required, what remains forbidden, who may validate, and what counts as done.

Listing 1: *Mission Contract skeleton.*

```
{
  "mission_id": "mission-001",
  "objective": "Convert AI vendor claims into a buyer-ready Evidence Docket.",
  "decision_to_support": "Which claims are trusted, blocked, or jobified?",
  "proof_level": "L3_EXTERNAL_REVIEW",
  "risk_class": "medium",
  "success_criteria": [
    "Proof Debt is explicit",
    "Jobs are custom to the mission",
    "Evidence is reviewer-ready",
    "Chronicle decision is recorded"
  ],
  "failure_criteria": [
    "Unsupported claim enters Chronicle",
    "Private intelligence is exposed as plaintext",
    "Local mode crosses the live settlement boundary"
  ],
  "required_validators": ["human-reviewer", "agi-node-validator"],
  "blocked_claims": ["certified", "guaranteed ROI", "production authorized"],
  "rollback_obligations": ["retire skill if evidence or scope fails"],
  "done_condition": "human-review-ready governed decision state"
}
```

The proof level should remain immutable within a run unless a signed revision event explicitly updates the mission. This prevents the evidence burden from changing after results are visible.

9 PROOF DEBT

Proof Debt is the distance between what a system wants to claim and what it can support.

$$D_{\text{proof}} = \sum_i \text{Impact}(c_i) \cdot \text{Unsupported}(c_i) \cdot \text{Irreversibility}(c_i) \cdot \text{ReuseRisk}(c_i). \tag{4}$$

Every material assertion should be classified as one of:

supported | needs evidence | needs replay | needs reviewer | needs boundary | blocked | out of scope

Proof Debt is not failure. It is the job factory's fuel. A system that cannot state what proof is missing is still chatting rather than operating.

10 CUSTOM AGI JOB FACTORY

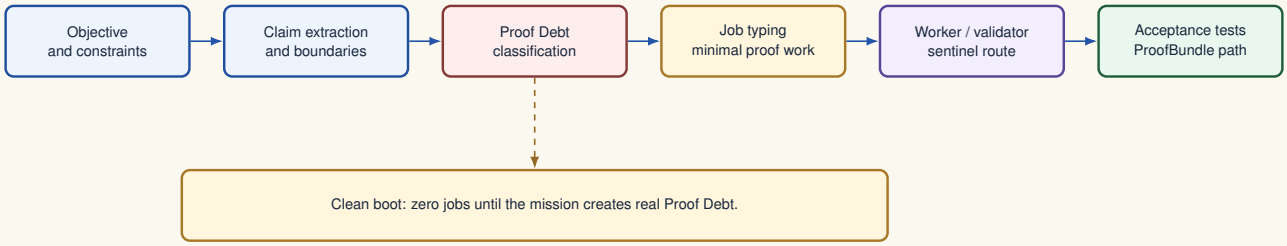


Figure 4: *Custom AGI Job factory. The factory begins empty and creates only the proof contracts required by the current mission and proof level.*

The Workbench must start empty. No canned jobs, fake graph, or preloaded authority may be shown before a mission creates real Proof Debt.

Clean-Boot Invariant

0 AGI Jobs **0 ProofBundles** **0 Validated Skills** **0 Graph Roots** **Chronicle closed**

An AGI Job is an executable proof contract:

Listing 2: *AGI Job proof contract.*

```
{
  "job_id": "agi-job-reviewer-verdict-001",
  "source_proof_debt": "proof-debt-001",
  "claim": "The workflow is buyer-ready.",
  "why_needed": "The claim cannot influence Chronicle until reviewed.",
  "worker": "Human Reviewer or Reviewer Agent",
  "validator": "AGI Node Validator + Human Review",
  "sentinel": "Boundary Sentinel",
  "deliverables": ["reviewer-verdict.json", "notes.md"],
  "acceptance_tests": [
    "supported and unsupported claims identified",
    "blocked claims preserved",
    "verdict artifact signed or hashed",
    "rollback condition recorded"
  ],
  "proofbundle_return_path": "proofbundles/agi-job-reviewer-verdict-001/",
  "rollback_condition": "Chronicle remains closed if verdict is incomplete"
}
```

If a job does not state what evidence would satisfy it, it is not a proof job. It is a wish.

11 AGIJOBMANAGER HANDOFF BOUNDARY

GoalOS may prepare AGIJobManager-style job specifications and handoff packets, but local mode must not imply live posting, escrow, validator voting, dispute resolution, or settlement.

Function	Local GoalOS	External live boundary
Job definition	Generate job specification, acceptance tests, blocked claims, validator route	Submit live transaction
Identity	Record intended agent / validator names	Wallet or authorized agent signs
Funds	Record proposed budget and risk	Escrow or stake assets
Validation	Prepare rubric and evidence requirements	Validators attest, dispute, or finalize
Settlement	Prepare completion packet	Contract releases or withholds value
Chronicle	Prepare candidate memory entry	Institutional or protocol policy admits final state

Table 3: *Local/live separation. A handoff packet is a proposal artifact, not a transaction.*

Boundary Rule

Local mode may generate, inspect, hash, package, and export. It may not silently connect a wallet, move funds, publish private intelligence, or represent a simulated handoff as live settlement.

12 PROOFBUNDLES AND EVIDENCE DOCKETS

Completed AGI Jobs return ProofBundles: content-addressed packages containing artifacts, provenance, replay information, validator decisions, acceptance-test results, cost and latency records, and rollback references.

Listing 3: *ProofBundle skeleton.*

```
{
  "proofbundle_id": "proofbundle-001",
  "job_id": "agi-job-reviewer-verdict-001",
  "artifact_hashes": ["sha256:..."],
  "source_provenance": ["source://..."],
  "replay_manifest_hash": "sha256:...",
  "worker_signature": "0x...",
  "validator_verdict": "supported_with_limits",
  "acceptance_tests": {"passed": 4, "failed": 0},
  "cost_ledger_hash": "sha256:...",
  "risk_ledger_hash": "sha256:...",
  "rollback_reference": "rollback://mission-001"
}
```

The Evidence Docket is the institutional proof room. It ties claims to sources, ProofBundles, replay paths, validator verdicts, risk, cost, blocked claims, public/private boundaries, and decision state.

A reviewer should be able to answer:

- What was attempted and under which mission contract?
- Which claims are supported, limited, blocked, or unresolved?
- What evidence and provenance exist?
- What was replayed or independently checked?
- Who validated the result and against which rubric?
- What risk, cost, latency, and rollback conditions remain?

- What may enter Chronicle, and under what scope?

Listing 4: *Evidence Docket skeleton.*

```
{
  "docket_id": "evidence-docket-001",
  "mission_id": "mission-001",
  "proof_level": "L3_EXTERNAL_REVIEW",
  "claims_matrix": [],
  "proof_debt_resolved": [],
  "proofbundles": [],
  "reviewer_verdicts": [],
  "replay_manifest": {},
  "risk_ledger": [],
  "cost_ledger": [],
  "blocked_claims": [],
  "chronicle_recommendation": "hold | admit | repair | reject",
  "decision_state": "reviewer-ready"
}
```

Docket Law

The point is not to store more files. The point is to make trust inspectable.

13 CHRONICLE: THE MEMORY FIREWALL

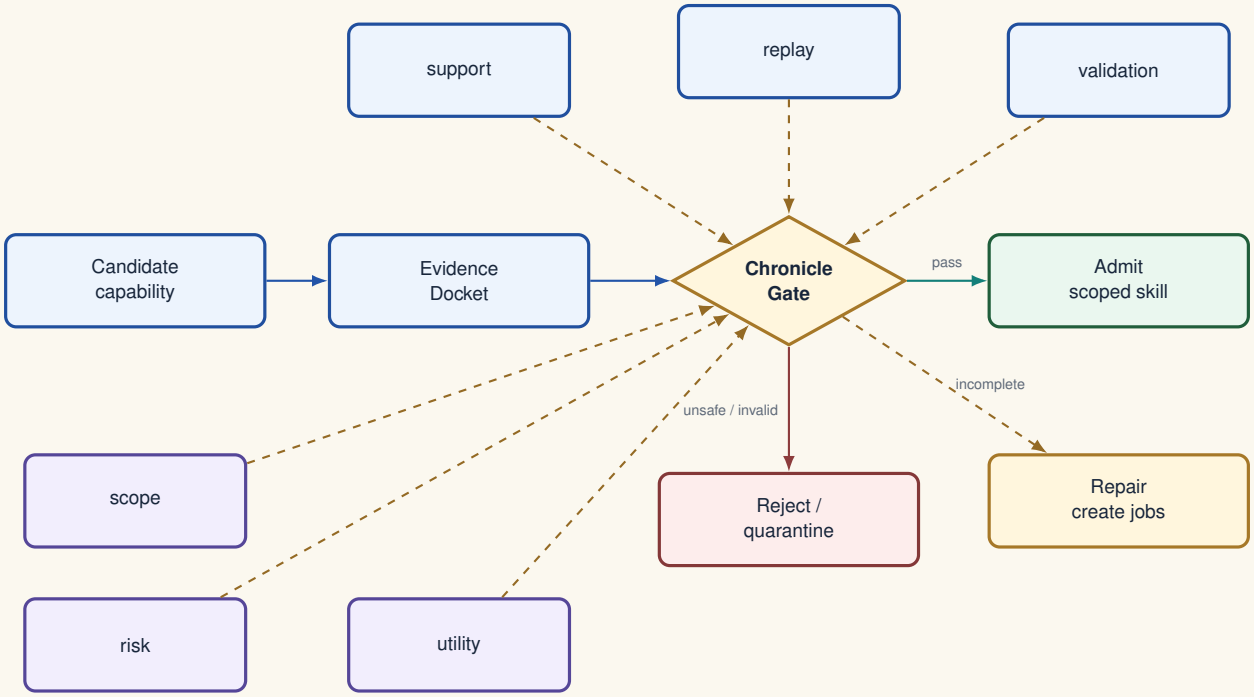


Figure 5: *Chronicle admission. Evidence becomes authority only through a configured policy gate; failed candidates produce repair work or preserved rejection state, not memory.*

Nothing enters memory because it sounds useful. A candidate capability enters Chronicle only if it satisfies the configured proof level.

Let c be a candidate skill and ℓ the proof level. Define

$$\text{Admit}(c,\ell) = \mathbf{1}\left[E(c) \geq \theta_E^\ell \wedge R(c) \leq \theta_R^\ell \wedge V(c) \geq \theta_V^\ell \wedge P(c) = 1 \wedge B(c) = 1 \wedge U(c) \geq \theta_U^\ell\right], \tag{5}$$

where E is evidence strength, R risk, V validation coverage, P replay readiness, B boundary compliance, and U worth-keeping utility.

Chronicle may return:

Decision	Meaning
Admit	Skill may enter the graph within explicit scope, proof level, and rollback conditions.
Repair	Create new AGI Jobs for missing proof.
Reject	Preserve the reason; do not reuse.
Quarantine	Isolate because of risk, ambiguity, or unsafe authority.
Retire	Remove future influence while retaining provenance and history.
Supersede	Replace with a newer validated version, preserving lineage and rollback.

Table 4: *Chronicle decision vocabulary.*

14 FLEXIBLE PROOF LEVELS

Not every mission requires the same evidence burden.

Level	Name	Required posture	Memory effect
L1	Structural / Demo	Mission structure, explicit Proof Debt, coherent local artifacts	No durable Chronicle influence
L2	Internal Review	Operator or internal reviewer confirms usefulness, scope, and obvious safety	Temporary, clearly labeled reuse
L3	External Review	Independent reviewer or AGI Node validation, replay path, provenance	Scoped reusable skill
L4	Mainnet Handoff	Validated evidence, job specification, authorization, live-boundary review	May inform external execution preparation
L5	Chronicle / Formal	Multi-validator evidence, replay, baselines, risk gate, rollback, freshness	Durable future-mission influence

Table 5: *Proof-level ladder. Higher levels increase cost, validation burden, and reuse authority.*

The proof level is an operational policy, not a decorative label. It controls job granularity, validator density, evidence strictness, Chronicle threshold, mainnet readiness, and claim authority.

15 VALIDATED SKILL GRAPH

The durable asset is not the prompt. It is validated reusable capability.

A note says, "this worked once." A Validated Skill says, "this capability may be reused under these conditions because it survived evidence, replay, validation, policy, and rollback review."

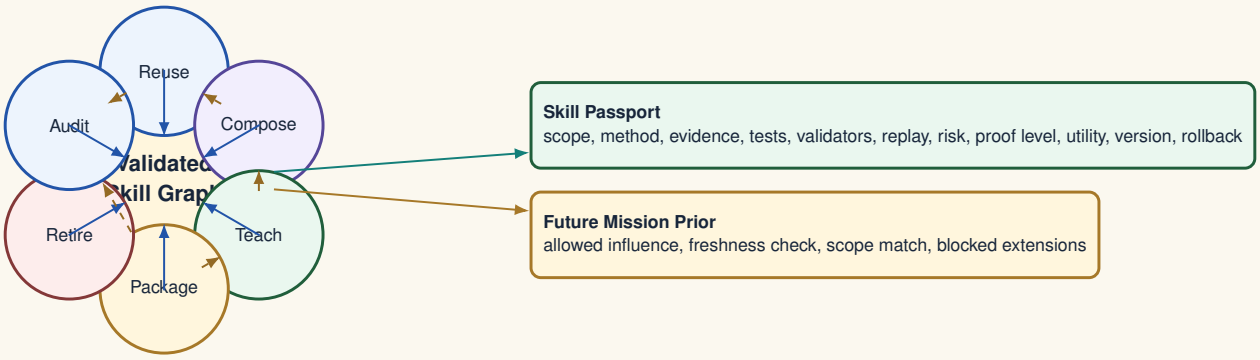


Figure 6: *Validated Skill Graph. The graph stores governed capability objects and policy-aware operations, not prompt fragments or unreviewed transcripts.*

A skill node is a governed capability object:

$$s = \langle id, type, scope, method, evidence, tests, validators, replay, risk, policy, version, rollback, utility, lineage \rangle. \tag{6}$$

Field	Operational meaning
Identity	Stable identifier, mission lineage, content hash, and version.
Scope	Admissible-use envelope: domain, assumptions, proof level, tools, jurisdiction, excluded uses.
Method	Procedure, sequencing, heuristics, and toolchain.
Evidence	ProofBundles, Evidence Docket, replay manifests, reviewer notes, and metrics.
Tests	Acceptance tests required before reuse or composition.
Validators	Human, AGI Node, council, or hybrid validators and quorum state.
Risk / policy	Failure modes, open debt, retention period, permitted reuse, mainnet boundary.
Rollback	Conditions for demotion, quarantine, retirement, or supersession.
Utility	Measured and expected contribution to quality, speed, cost, trust, or safety.

Table 6: *Validated Skill Passport fields.*

Listing 5: *Validated Skill Passport skeleton.*

```
{
  "skill_id": "skill-ai-claim-to-proof-room-001",
  "name": "Convert AI vendor claims into a buyer-ready proof room",
  "scope": {
    "domain": "AI vendor trust / procurement",
    "allowed_use": ["claim matrix", "Evidence Docket", "reviewer room"],
    "excluded_use": ["legal certification", "security certification", "investment advice"]
  },
  "proof_level": "L3_EXTERNAL_REVIEW",
  "evidence_docket_hash": "0x...",
  "proofbundle_refs": ["proofbundle-001", "proofbundle-002"],
  "validator_route": ["human-reviewer", "agi-node-validator"],
  "replay_manifest_hash": "0x...",
  "risk_ledger_hash": "0x...",
  "rollback_condition": "retire if verdict or scope is invalidated",
  "worth_keeping_score": 0.86,
  "chronicle_admitted": true,
  "version": "1.0.0"
}
```

16 GRAPH OPERATIONS

The graph supports evidence-preserving, policy-aware operations:

- **Reuse**: apply a validated skill inside its admitted scope.
- **Compose**: combine multiple skills into a higher-order candidate; the composition requires a new gate.
- **Teach**: derive training material, checklists, or masterclass lessons while preserving scope and provenance.
- **Package**: emit a proof room, playbook, job template, or handoff packet.
- **Audit**: reopen a skill when evidence, policy, context, or freshness changes.
- **Retire**: remove future influence while preserving lineage and reason.
- **Supersede**: replace a skill with a newer validated version and explicit rollback path.

Let $G = (S, E)$ be the skill graph. Each edge

$$e = (s_i, s_j, \rho, w, p) \tag{7}$$

connects skills with relation type ρ, utility weight w, and policy condition p. A future mission may use an edge only if both endpoint skills satisfy the mission's proof level, freshness, scope, and risk requirements.

17 WORTH-KEEPING POLICY

Chronicle admission is necessary but not sufficient for durable retention. A skill should persist only when expected value exceeds maintenance and governance cost.

$$W(s) = \alpha\Delta Q + \beta\Delta T^{-1} + \gamma\Delta C^{-1} + \delta\Delta R^{-1} + \eta U + \zeta H + \xi P - \mu M - \nu D - \omega B, \tag{8}$$

where ΔQ is quality improvement, ΔT^{-1} time reduction, ΔC^{-1} cost reduction, ΔR^{-1} risk reduction, U expected reuse, H teaching value, P packaging value, M maintenance cost, D drift risk, and B boundary complexity.

A skill is retained when

$$\mathrm{Admit}(s, \ell) = 1 \quad \wedge \quad W(s) \geq \theta_{\mathrm{retain}}. \tag{9}$$

Memory-Bloat Rule

Chronicle admission proves that a skill may be reused. Worth-keeping policy decides whether it should remain durable memory.

18 FUTURE-MISSION INFLUENCE

A future mission does not inherit a vague memory blob. It receives a bounded capability prior:

Listing 6: *Future mission seed.*

```
{
  "future_mission_id": "mission-002",
```

```
  "seeded_by_skill": "skill-ai-claim-to-proof-room-001",
  "allowed_influence": [
    "claim extraction",
    "Proof Debt creation",
    "reviewer route selection",
    "Evidence Docket scaffolding"
  ],
  "blocked_influence": [
    "certification claims",
    "financial advice",
    "production authorization",
    "live settlement"
  ],
  "required_rechecks": ["scope match", "evidence freshness", "risk posture"]
}
```

This is how capability compounds without contaminating future work.

19 MERKLE-NATIVE SKILL MEMORY

A Validated Skill Graph is large, dynamic, partially private, and audit-sensitive. Merkle roots are a natural commitment primitive because they compress large sets into small digests while preserving membership proofs, tamper detection, selective disclosure, and append-only version continuity.

19.1 Domain-separated leaves

Let H be a collision-resistant hash and $\mathrm{canon}(x)$ deterministic canonicalization. A typed leaf is

$$\mathrm{Leaf}_D(x) = H(\texttt{GoalOS} : \| \; D \; \| : \| \; \mathrm{schemaHash}(D) \; \| \; \mathrm{canon}(x)). \tag{10}$$

Domain separation prevents evidence from being replayed as a skill, policy from being replayed as an attestation, or Chronicle state from being replayed as evidence.

Recommended domains include:

`GoalOSSkillV1` `GoalOSEvidenceV1` `GoalOSProofBundleV1` `GoalOSPolicyV1` `GoalOSAttestationV1`
`GoalOSCapsuleV1` `GoalOSChronicleV1` `GoalOSGraphRootV1`

19.2 Sorted-pair Merkle construction

For child digests a, b,

$$\mathrm{Parent}(a,b) = H(\texttt{GoalOSMerkleNodeV1} \; \| \; \min(a,b) \; \| \; \max(a,b)). \tag{11}$$

The odd-node rule, leaf ordering, canonicalization version, and hash function are part of the policy commitment. Silent tree-policy changes invalidate proofs.

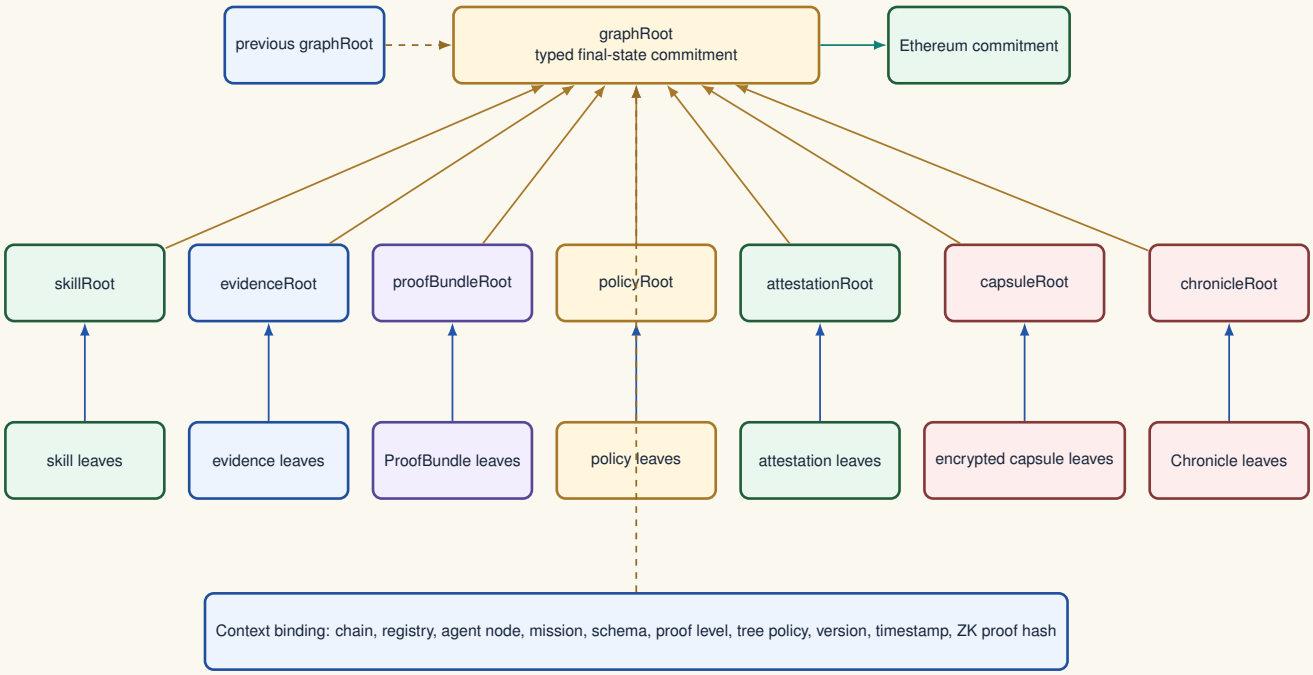


Figure 7: *Merkle forest. Typed roots preserve meaning; the final graph root binds admitted capability, proof state, policy, privacy commitments, and previous-root continuity.*

Typed roots are

$$\text{skillRoot}_t = \text{MerkleRoot}(\{\text{Leaf}_{Skill}(s_i)\}), \tag{12}$$
$$\text{evidenceRoot}_t = \text{MerkleRoot}(\{\text{Leaf}_{Evidence}(e_i)\}), \tag{13}$$
$$\text{proofBundleRoot}_t = \text{MerkleRoot}(\{\text{Leaf}_{ProofBundle}(p_i)\}), \tag{14}$$
$$\text{policyRoot}_t = \text{MerkleRoot}(\{\text{Leaf}_{Policy}(\pi_i)\}), \tag{15}$$
$$\text{attestationRoot}_t = \text{MerkleRoot}(\{\text{Leaf}_{Attestation}(a_i)\}), \tag{16}$$
$$\text{capsuleRoot}_t = \text{MerkleRoot}(\{\text{Leaf}_{Capsule}(\gamma_i)\}), \tag{17}$$
$$\text{chronicleRoot}_t = \text{MerkleRoot}(\{\text{Leaf}_{Chronicle}(k_i)\}). \tag{18}$$

The final graph root binds typed roots and context:

$$\text{graphRoot}_t = H(\texttt{GoalOSGraphRootV1} \parallel chainId \parallel registry \parallel agentNode \tag{19}$$
$$\parallel missionId \parallel version \parallel \text{graphRoot}_{t-1} \tag{20}$$
$$\parallel \text{skillRoot}_t \parallel \text{evidenceRoot}_t \parallel \text{proofBundleRoot}_t \tag{21}$$
$$\parallel \text{policyRoot}_t \parallel \text{attestationRoot}_t \parallel \text{capsuleRoot}_t \tag{22}$$
$$\parallel \text{chronicleRoot}_t \parallel zkProofHash_t). \tag{23}$$

Root Law

The root is not the skill. It is the tamper-evident public commitment to the admitted skill-graph state.

20 MEMBERSHIP, TAMPER, AND CONTINUITY PROOFS

A reviewer may verify that a skill, evidence item, ProofBundle, policy, attestation, or Chronicle decision belongs to a committed state without downloading the entire graph.

A Merkle proof is valid when repeated parent hashing reconstructs the committed root. A modified leaf, sibling, domain, schema, tree policy, or context must fail verification. Each new graph root binds the previous root, producing append-auditable lineage.

Production acceptance tests include:

1. identical canonical objects hash identically across independent clients;
2. a modified leaf fails verification;
3. a modified proof path fails verification;
4. a skill cannot enter `skillRoot` unless Chronicle admitted it;
5. rejected or quarantined candidates retain a reason hash but no future influence;
6. a new `graphRoot` binds the prior root;
7. local packets record `live_transaction=false`;
8. no plaintext private skill content appears in a public packet.

21 PRIVATE INTELLIGENCE AND PUBLIC PROOF

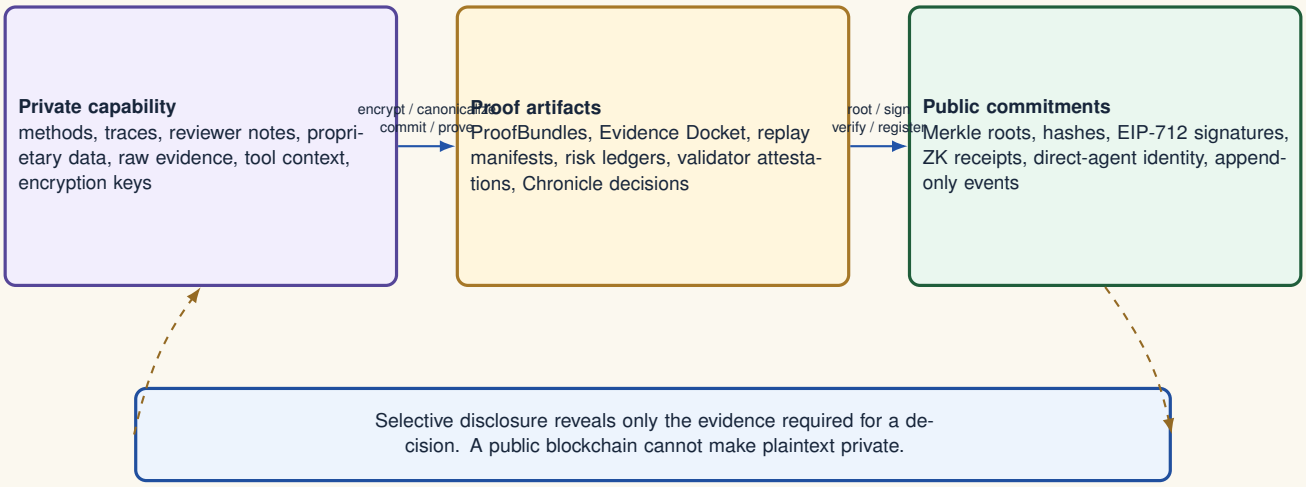


Figure 8: *Public proof / private intelligence boundary. On-chain state carries compact commitments and authorization, not plaintext private intelligence.*

The phrase "private on-chain intelligence" must be interpreted precisely. A public blockchain cannot keep plaintext private. Secure storage means ciphertext, commitments, controlled pointers, and proofs.

Mode	Public material	Use
Commitment-only	Graph root, Evidence Docket root, ProofBundle root, policy hash	Default for sensitive or high-volume skill state
Encrypted URI	Roots plus content-addressed pointer to encrypted payload	Practical private graph storage
Encrypted on-chain capsule	Compact ciphertext plus commitments	Public availability of ciphertext when economically justified
ZK-verified hidden capsule	Public inputs and proof over hidden witness	Prove eligibility without revealing method, evidence, or reviewer notes

Table 7: *Privacy-preserving storage modes.*

A zero-knowledge policy proof may establish that:

- the capsule commits to the published root;
- the Evidence Docket and ProofBundles match their roots;
- validator quorum is satisfied;
- required proof level and replay conditions are met;
- blocked claims are excluded;
- rollback state exists;

without revealing the private method, traces, reviewer notes, or proprietary evidence.

22 DIRECT-AGENT AUTHORIZATION

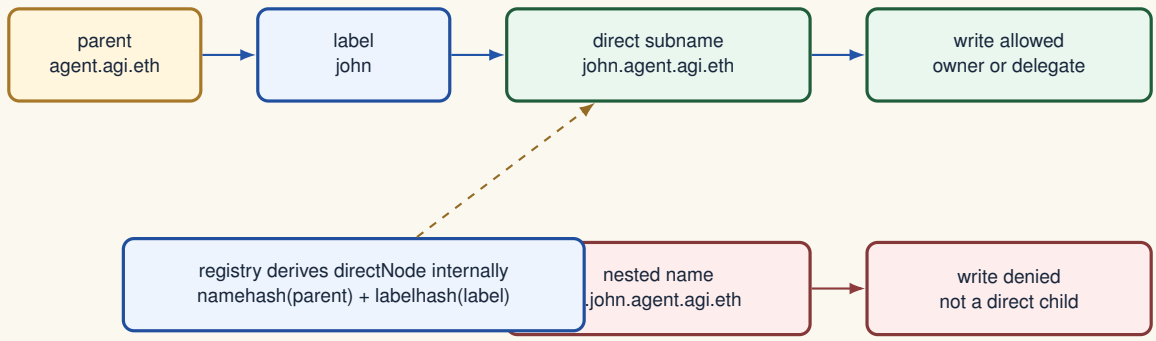


Figure 9: *Direct-agent authorization. The registry derives the direct child internally and rejects arbitrary or nested-name authority claims.*

Only the controller of a direct subname under `agent.agi.eth`, or a delegate approved by that current controller, should be able to update that agent's graph commitments. The registry should derive the direct child node internally from the parent namehash and supplied labelhash.

Allowed examples:

`john.agent.agi.eth` `research.agent.agi.eth`

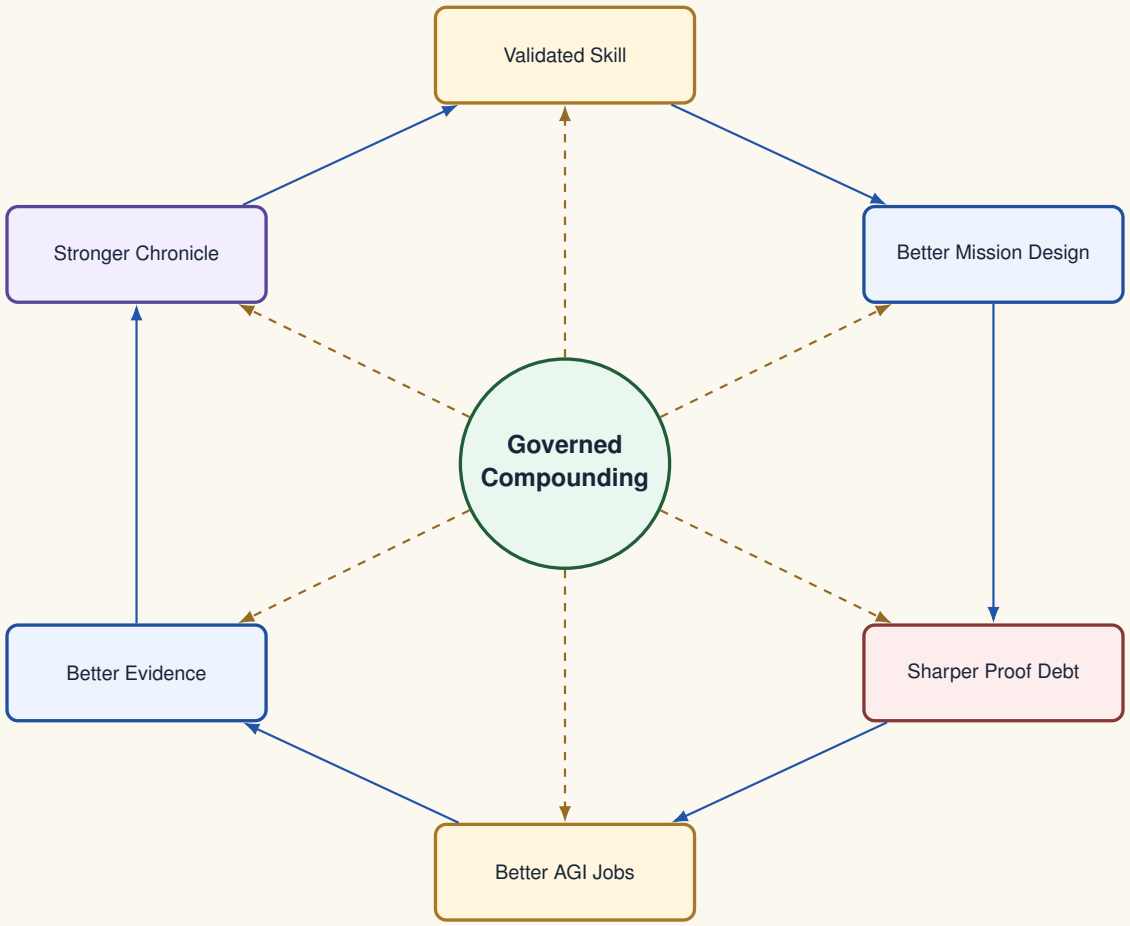
Rejected as direct-agent authority:

`x.john.agent.agi.eth` `random.eth`

The authority model should support current ownership checks, wrapped-name ownership where applicable, revocable delegation, pause control, append-only root history, supersession, retirement, and event-based audit.

23 GOVERNED CAPABILITY IMPROVEMENT

The capability-improvement loop is not unrestricted self-modification. It is a governed process in which only validated, bounded capability may influence future work.



Governance corridor

Exploration may guide allocation, but outcome authority remains mechanical: evidence, replay, baseline comparison, risk, validator coverage, persistence, scope, and rollback govern promotion.

Figure 10: *Governed capability improvement. Future work improves only from Chronicle-admitted, replayable, scope-bounded skill.*

The loop is:

Validated Skill → Better Mission Design → Sharper Proof Debt → Better AGI Jobs → Better Evidence → Stronger Chronicle Decision → Stronger Skill Graph.

The governing invariant is:

$$\text{FutureInfluence}(s, m) = 1 \tag{24}$$

only if the skill is admitted, its scope matches the mission, its evidence is fresh enough, its risk posture remains acceptable, and its proof level satisfies the mission.

Improvement Law

Exploration may be open-ended. Promotion is not. Novelty raises the burden of proof rather than lowering it.

24 AEP-001 Governance Alignment

The proof-to-capability architecture aligns with a constitutional doctrine:

$$\textbf{Aim} \rightarrow \textbf{Act} \rightarrow \textbf{Prove} \rightarrow \textbf{Evolve}$$

At the protocol-object level:

$$\textbf{Commit} \rightarrow \textbf{Execute} \rightarrow \textbf{Prove} \rightarrow \textbf{Evolve}$$

The two views map as follows:

Public doctrine	Protocol object	Meaning
Aim	GoalOSCommit / Mission Contract	Objective, authority, constraints, risk, budget, evaluators, rollback
Act	Run Commitment / AGI Jobs	Bounded agents, tools, context, policies, approvals, execution
Prove	ProofBundles / Evidence Docket	Trace roots, artifacts, evaluations, risk, cost, signatures, verdicts
Evolve	Chronicle / Skill Admission / Graph Root	Governed right to influence future work under scope and rollback

Table 8: *Public doctrine and protocol-object mapping.*

The constitutional rules are:

- proof before propagation;
- private execution, public commitments;
- score is advisory, gates are mandatory;
- rollback before release;
- artifact-level credit and blame;
- commercial neutrality across model, runtime, evaluator, storage, and chain providers;
- claim discipline: architecture is not empirical achievement.

25 Threat Model

The system must assume that models, agents, tools, validators, interfaces, keys, and operators can fail or be adversarial.

Threat	Failure mode	Primary controls
Unsupported claim leakage	Draft or partial result is treated as validated	Claim states, Chronicle gate, reviewer UI, blocked-claim ledger
Validator capture	Colluding or low-quality validators accept false evidence	Quorum, reputation, commit-reveal, independent replay, challenge path
Replay substitution	Old or modified packets are presented as current	Mission IDs, nonces, content hashes, expiry, graph-root lineage
Cross-domain replay	One object type is reused as another	Domain-separated leaf hashing and schema hashes
Root equivocation	Different audiences receive inconsistent graph states	Append-only history, previous-root binding, public events
Unauthorized write	Unapproved identity commits state	Direct-child derivation, owner/delegate checks, pause and revocation
Private-data leakage	Plaintext methods or customer data reach public state	Encryption, redaction, controlled pointers, pre-commit scanners, ZK
Memory contamination	Unsupported capability influences later missions	Chronicle-only priors, freshness and scope checks, retirement
Prompt / config drift	Reproducibility fails silently	Hash-bound prompt packs, configs, state, schemas, and signed revisions
Settlement confusion	Local preparation is mistaken for live execution	Explicit boundary flags, no hidden wallet calls, user consent, external action

Table 9: *Threats and controls.*

26 Security Invariants

A conforming implementation should enforce:

1. no unsupported claim may enter Chronicle;
2. no skill may influence a future mission outside admitted scope;
3. no graph root may omit its policy, schema, proof level, agent identity, or previous root;
4. no private plaintext may be published by default;
5. no local-mode interface may silently sign or transmit a live transaction;
6. no validator may approve without an inspectable rubric and evidence reference;
7. no promoted skill may lack a rollback or retirement path;
8. no high-impact irreversible action may rely solely on model self-evaluation.

27 User Experience: The Proof Loop Must Be Playable

A powerful architecture is insufficient if users cannot understand what changed after each action. The product should teach by operation.

The canonical user journey is:

1. Start Objective.
2. Create Proof Mission.

3. Generate AGI Jobs.
4. Return Evidence.
5. Run Chronicle Gate.
6. Admit Validated Skill.
7. Build and verify Graph Root.
8. Improve Future Mission.

Every click should visibly update state, artifacts, metrics, and explanation. The interface should answer four questions in plain language:

- What happened?
- Why did it happen?
- What is still blocked?
- What can the user do next?

Action	Visible system change	Teaching moment
Start Objective	Mission state appears; Workbench remains empty	Jobs must come from real Proof Debt
Create Proof Mission	Scope, proof level, blocked claims, validators, and criteria appear	Authority begins with a bounded mission
Generate AGI Jobs	Custom work objects populate the Workbench	Unsupported claims become executable proof work
Return Evidence	ProofBundles and Docket entries appear	Output becomes inspectable evidence
Run Chronicle	Gate scores and decision paths update	Evidence does not become memory automatically
Admit Skill	Skill node and passport appear	Reusable capability is a governed object
Build Graph Root	Typed roots, proof path, and tamper test appear	A root commits state without revealing the full graph
Improve Mission	Measured bounded lift appears	Only admitted capability may compound

Table 10: *Masterclass interaction design.*

The interface should support progressive disclosure: a non-technical user sees the loop and explanations; an operator sees claims, jobs, and Dockets; a reviewer sees evidence and gates; an engineer sees hashes, schemas, proofs, and contract packets.

28 Production Artifact Standard

A complete run should emit a deterministic, reviewer-ready artifact set:

Listing 7: *Canonical run package.*

```
proof-run/
  00_manifest.json
  01_objective.md
  02_mission-contract.json
  03_proof-level-policy.json
```

```
04_claims-matrix.json
05_proof-debt-register.json
06_agi-job-queue.json
07_proofbundle-manifest.json
08_evidence-docket.json
09_replay-manifest.json
10_risk-ledger.json
11_cost-latency-ledger.json
12_validator-report.json
13_chronicle-decision.json
14_validated-skill-passport.json
15_graph-root-packet.json
16_future-mission-brief.json
17_action-graph.csv
18_reviewer-room.html
19_run-trace.md
```

Each artifact should carry schema version, mission ID, content hash, generation time, public/private classification, and provenance references. The manifest should bind every file and include toolchain versions.

29 CONFORMANCE LEVELS

Level	Name	Minimum requirement
C0	Narrative	Loop and claim boundary are stated, but artifacts are not machine-checkable
C1	Structured	Mission, Proof Debt, jobs, Docket, and Chronicle objects validate against schemas
C2	Replayable	Inputs, tools, versions, traces, and ProofBundles support deterministic or bounded replay
C3	Reviewer-ready	Independent reviewer can inspect evidence, claims, risk, and decision state
C4	Root-verifiable	Typed Merkle roots and proof paths reproduce; tamper tests fail correctly
C5	Testnet-authorized	Direct-agent authorization and graph-root commitments operate on reviewed testnet deployment
C6	Externally validated	External review and independent replay confirm a real mission result
C7	Production-governed	Operational monitoring, incident response, key policy, rollback, privacy, and recurring audits are active

Table 11: *Conformance ladder. Production authority requires more than a polished local interface.*

30 IMPLEMENTATION ARCHITECTURE

A practical system can remain modular:

- **Mission service**: objective capture, contract generation, proof-level policy.
- **Claim and Proof Debt service**: claim extraction, classification, priority, blocked states.
- **Job service**: custom AGI Job creation, worker/validator routing, acceptance tests.
- **Evidence service**: ProofBundle ingestion, provenance, replay, Docket assembly.
- **Chronicle service**: policy evaluation, admit/repair/reject/quarantine/retire/supersede.
- **Skill Graph service**: node and edge lifecycle, versioning, freshness, utility, rollback.
- **Commitment service**: canonicalization, typed Merkle roots, proof generation, packet export.
- **Authorization service**: ENS-style identity resolution, direct-child derivation, delegation.

- **Review surface**: reviewer room, audit logs, claim boundaries, signatures, decisions.
- **Observability**: mission traces, job traces, evidence traces, validator traces, decision traces.

31 Deployment Path

The recommended sequence is conservative:

1. Freeze the canonical schemas and object semantics.
2. Run browser-local and repository-local validation suites.
3. Conduct smart-contract, cryptography, privacy, and key-management reviews.
4. Deploy a reviewed registry to testnet.
5. Run Proof Mission 001 with an external reviewer.
6. Admit one real skill, commit one testnet root, and measure a second mission.
7. Publish the Evidence Docket and reviewer verdict.
8. Integrate live AGIJobManager flows only after explicit wallet and settlement review.
9. Consider mainnet only after monitoring, pause, incident response, and rollback procedures are exercised.

32 Evaluation Program

The architecture becomes scientific only when compared against simpler baselines under controlled conditions.

Recommended baselines:

ID	Baseline	Purpose
B0	Null / manual	Establish no-agent and human-only reference
B1	Report-only LLM	Measure fluent output without proof loop
B2	Prompt-library reuse	Test whether copied prompts create reliable capability
B3	Ungated agent memory	Measure contamination and authority leakage
B4	GoalOS without Chronicle	Isolate the value of memory gating
B5	GoalOS with Chronicle, no skill reuse	Isolate skill-graph contribution
B6	Full GoalOS loop	Mission, jobs, evidence, Chronicle, VSG, Merkle root, future-mission prior

Table 12: *Evaluation baselines.*

Key metrics include:

- verified work per unit cost, time, token, tool call, and human review;
- Proof Debt retired and unresolved debt surfaced;
- provenance coverage, replay success, and validator agreement;

- unsupported-claim leakage and Chronicle false-admission rate;
- skill reuse success, scope violations, drift, and rollback frequency;
- future-mission quality, speed, cost, and risk change under equal constraints;
- Merkle proof success, tamper detection, unauthorized-write rejection, and privacy leakage;
- delayed outcomes and external reviewer reproduction.

33 Falsification Criteria

The architecture is weakened or falsified if:

1. the full loop does not outperform simpler baselines on verified work per cost;
2. Chronicle admits unsupported or unsafe capability at an unacceptable rate;
3. skill reuse increases ungrounded authority or scope drift;
4. reviewers cannot reconstruct why a capability was admitted;
5. replay is unreliable under pinned inputs and versions;
6. graph roots do not match admitted state or tamper tests pass incorrectly;
7. privacy-preserving representations leak sensitive content;
8. direct-agent authorization can be bypassed by nested or arbitrary names;
9. future-mission improvement disappears under controlled baselines;
10. governance overhead exceeds the value of proof and reuse.

34 Proof Run 001

The decisive next milestone is not another interface. It is one complete, externally reviewable proof loop.

Figure 11: *Proof Run 001. The decisive next milestone is one externally reviewable loop from objective to measurable future-mission improvement.*

Recommended mission:

Proof Mission 001

Convert one AI vendor claim set into a buyer-ready Evidence Docket; create custom AGI Jobs for missing proof; obtain an independent reviewer verdict; admit one reusable skill if the gate passes; build a testnet graph-root packet; and measure whether that skill improves a second mission under equal constraints.

The first candidate skill is deliberately narrow:

Convert AI vendor claims into a buyer-ready proof room.

Required outputs:

- Mission Contract and claims matrix;
- Proof Debt register and custom AGI Job queue;
- returned ProofBundles and source provenance;
- Evidence Docket, risk ledger, and replay manifest;
- external reviewer verdict;
- Chronicle decision and Validated Skill Passport;
- Merkle graph-root packet and membership proof;
- second-mission comparison against B1, B2, B4, B5, and B6;
- public-safe reviewer room and private audit appendix.

Success requires more than completion. The second mission must show a measurable improvement without increased unsupported-claim leakage, privacy exposure, or review burden beyond the configured threshold.

35 Release Gates

Before claiming strong production status, the following gates should pass:

1. schemas and canonicalization independently reproduce;
2. browser-local and repository-local tests pass;
3. contract and cryptographic review complete;
4. testnet registry deployed with pause, delegation, retirement, and event audit;
5. external reviewer replays Proof Run 001;
6. first skill is admitted with explicit scope and rollback;
7. first graph root and membership proof verify independently;
8. second mission demonstrates bounded improvement;
9. claim boundary and public/private review pass;
10. incident response and rollback are exercised.

Final Empirical Standard

No Evidence Docket, no empirical authority. No independent replay, no external validation claim. No measured second-mission improvement, no compounding-capability claim.

36 Product and Business Architecture

The immediate commercial wedge is a Proof Mission Sprint. The deliverable is not a long report; it is a governed decision state and a reusable capability candidate.

A complete offer may include:

- Mission Contract;
- Proof Debt Register;
- custom AGI Job Queue;
- Evidence Docket and Reviewer Room;
- Chronicle recommendation;
- Validated Skill Passport;
- graph-root commitment packet;
- executive brief, action graph, and future-mission improvement plan.

The product line can expand from Proof Mission Factory to Skill Capture Studio, AI Trust Rooms, Enterprise Proof OS, private Validated Skill Graph deployments, and eventually a validator / AGI Jobs marketplace. The durable moat is the method and the accumulated graph of evidence-backed capability, not prompt volume.

A Canonical Graph-Root Packet

Listing 8: *Graph-root packet.*

```
{
  "schema": "goalos.graphRootPacket.v1",
  "mission_id": "mission-001",
  "agent": "john.agent.agi.eth",
  "agent_labelhash": "0x...",
  "proof_level": "L3_EXTERNAL_REVIEW",
  "previous_graph_root": "0x...",
  "skill_root": "0x...",
  "evidence_root": "0x...",
  "proofbundle_root": "0x...",
  "policy_root": "0x...",
  "attestation_root": "0x...",
  "capsule_root": "0x...",
  "chronicle_root": "0x...",
  "zk_proof_hash": "0x...",
  "graph_root": "0x...",
  "tree_policy": "SORTED_PAIR_DUPLICATE_LAST_V1",
  "canonicalization": "JCS_KECCAK_V1",
  "chronicle_decision": "ADMIT_WITH_SCOPE",
  "live_transaction": false,
  "boundary": {
    "plaintext_private_intelligence_onchain": false,
    "wallet_connected_in_local_mode": false,
    "settlement_in_local_mode": false
  }
}
```

B Minimal Registry Surface

Listing 9: *Minimal conceptual registry interface.*

```
interface IGoalOSGraphRegistry {
  event GraphRootCommitted(
    bytes32 indexed agentNode,
    bytes32 indexed graphRoot,
    bytes32 indexed previousGraphRoot,
    bytes32 evidenceRoot,
    bytes32 proofBundleRoot,
    bytes32 policyRoot,
    bytes32 attestationRoot,
    bytes32 capsuleRoot,
    bytes32 chronicleRoot,
    uint8 proofLevel,
    string metadataURI
  );

  function setDelegate(bytes32 labelhash, address delegate, bool allowed) external;
  function commitGraphRoot(bytes32 labelhash, GraphRootPacket calldata packet) external;
  function admitSkill(bytes32 agentLabelhash, bytes32 skillId, bytes32 leaf, bytes32[] calldata proof)
    external;
  function retireSkill(bytes32 agentLabelhash, bytes32 skillId, string calldata reasonURI) external;
  function verifyProof(bytes32 leaf, bytes32[] calldata proof, bytes32 root) external pure returns (bool)
    ;
}
```

This is a reference surface, not an audited or deployed production claim. A production registry requires independent security review, precise ENS and Name Wrapper semantics, verifier governance, key management, pause and recovery procedures, gas analysis, test vectors, and deployment controls.

C ACCEPTANCE CHECKLIST

A production graph-root packet should not be accepted unless:

1. mission ID, proof level, schema version, canonicalization, and tree policy are explicit;
2. skill, evidence, ProofBundle, policy, attestation, capsule, and Chronicle roots are present or explicitly empty under a domain-separated empty-root rule;
3. the graph root binds the previous root and full context;
4. every skill leaf in the reusable root has a Chronicle admission record;
5. evidence leaves carry provenance, support status, and blocked-claim state;
6. ProofBundle leaves carry job, worker, validator, acceptance, replay, and rollback references;
7. policy root includes thresholds, scope, allowed influence, and retirement rules;
8. agent identity satisfies the configured authorization model;
9. no plaintext private intelligence appears in the public packet;
10. the ZK proof state is present or explicitly marked not required for the selected proof level;
11. local mode records that no wallet, transaction, escrow, settlement, or live chain action occurred;
12. independent clients reproduce roots and membership proofs.

D Glossary

Term	Definition
AGI Job	Executable proof contract created from Proof Debt.
Chronicle	Policy gate that admits, repairs, rejects, quarantines, retires, or supersedes candidate capability.
Evidence Docket	Inspectable proof room tying claims to evidence, provenance, replay, risk, cost, validators, and decision state.
Graph Root	Typed commitment to the Validated Skill Graph state and its previous-root lineage.
Proof Debt	Difference between what the system wants to claim and what it can support.
Proof Mission	Bounded authority object that turns an objective into evidence-producing work.
ProofBundle	Returned evidence package from a completed AGI Job.
Validated Skill	Scoped, evidence-backed, replayable, versioned, rollbackable reusable capability.
Validated Skill Graph	Governed graph of admitted capability objects and policy-aware relationships.
ZK Policy Proof	Proof that hidden skill and evidence state satisfies public policy predicates without revealing the witness.

E References and Source Lineage

[1] Vincent Boucher. *GoalOS Mission OS: The Proof OS for Autonomous AI Work*. Montreal.AI Research, 2026.

[2] Vincent Boucher. *PROOF_LOOPS.md: Field Notes on Machines That Turn Work Into Capability*. GoalOS Protocol Note Series, 2026.

[3] Vincent Boucher. *GoalOS: The Validated Skill Graph*. Montreal.AI Research, 2026.

[4] Vincent Boucher. *Merkle Roots for the Validated Skill Graph*. Montreal.AI Research, 2026.

[5] Vincent Boucher. *Private On-Chain Validated Skill Graph*. Montreal.AI Research, 2026.

[6] Vincent Boucher. *GoalOS: The Proof-of-Evolution Constitution, AEP-001*. Institutional Standard, 2026.

[7] Vincent Boucher. *AGI ALPHA: A Scalable Substrate for Intelligence Organizations*. Montreal.AI Research, 2026.

[8] Vincent Boucher. *AGI Alpha RSI: Sovereign Invention Governance*. Montreal.AI Research, 2026.

F Final Operating Law

Canonical Law

Objective creates the mission. Mission creates Proof Debt. Proof Debt creates AGI Jobs. AGI Jobs create evidence. Evidence creates Chronicle candidates. Chronicle admits Validated Skills. Validated Skills create graph roots. Graph roots create trusted memory. Trusted memory improves the next mission.

GoalOS turns autonomous work into proof, proof into validated skill, validated skill into governed memory, and governed memory into stronger future work.