

Auditing Agent Actions with Customer-Retained Evidence: Report Integrity, Outcome Verification, and Economic Design

Howie Xu

Naysay

Working paper, version 0.6

Original manuscript: August 2026

Revised September 18, 2026

Abstract

AI-agent auditing requires distinguishing report integrity from evidence that a requested action occurred. We implement two customer-controlled HTTP components: a gateway that retains tool requests and responses, and an outcome service that binds a registered action to native Git or SQLite records. In the first study, twelve model-agent workflows generated 27 tool calls matching an independent simulated-backend ledger. A comparison of four evidence interfaces produced 176 eligible model-reviewer judgments. Verified gateway evidence identified 26 of 26 planted alterations, compared with 12 of 26 for submitted logs and 24 of 26 for independently retained ordinary logs. No interface falsely rejected an authentic control in eighteen control judgments per interface. In the second study, 26 authored backend/fault scenarios tested lost responses, retries, overwrites, stale versions, rollback and missing causal records. The outcome service distinguished action application from the current postcondition, returning eighteen correct execution decisions, eight unresolved decisions and no incorrect decisions. Retained native logs with the same reconciliation algorithm produced identical verdicts. These studies support a bounded implementation and characterize its failure behavior; they do not establish a general detection advantage for signatures. A prospective economic design separates USDG customer payments from operator capital and shared protocol control. It specifies bounded obligations, attributable violations and comparisons with conventional contracts and stablecoin collateral. Customer demand, investigation savings and native-token benefits remain unmeasured. The replication archive includes both implementations, raw observations, fault protocols and offline verification code.

1. Research question

An external evaluation service can observe an agent's final response without observing its tool actions. A report assembled by the evaluated party may reveal more detail, but that party can also omit a call or change its arguments. These are different problems: lack of information and lack of trust in the information supplied.

The present question is operational: when an audit receives a report of tool calls, does evidence retained at a customer-controlled forwarding boundary improve the reviewer's ability to identify discrepancies? We additionally ask whether signed receipts improve judgment beyond an ordinary log retained at the same trusted boundary.

A second technical question concerns the boundary of that evidence. A recorded HTTP response does not establish that the requested mutation committed. Conversely, a lost response does not establish that it failed. We implement and test an outcome service that asks whether a particular registered action was applied and whether its postcondition holds at a subsequent observation. Sections 3-6 report the model-reviewer pilot; Section 7 reports the separate Git/SQLite fault study. Their observations have different units and are not pooled.

A separate economic question is whether independent providers can deliver useful evidence services under enforceable obligations, and whether native-token participation and shared protocol control improve that arrangement relative to conventional contracts and stablecoin collateral. Section 8 proposes that design using published economics and a prospective paid-operator pilot. Existing customer-access policy and ledger observations are distinguished from this unimplemented proposal throughout.

The retained-log comparisons are necessary. Comparing signed records only with an agent's narrative would conflate provenance, completeness, information content, and cryptography. If a retained ordinary log performs equally well, the experiment supports the usefulness of the trusted record but does not establish a judgment advantage from signatures.

Earlier Naysay drafts studied an incentive model and then exact ambiguities in existing final-answer observations. This paper evaluates a new executable component and

collects new agent and reviewer observations. It does not present an elementary classification bound as a novel theoretical contribution.

2. Prior work and contribution

AgentDojo provides a framework and tasks for evaluating prompt injection against tool-using agents (Debenedetti et al., 2024). It motivated the separation between an agent's answer and its actions. The present experiment does not reuse its security labels or trajectories: the new records come from the local gateway and simulated business service described below.

AuditBench examines model-auditing techniques and the practical difficulty of turning available evidence into a correct assessment (Sheshadri et al., 2026). Our reviewer task is narrower: identify inconsistencies in reports of a bounded execution, rather than infer a hidden model disposition.

Tamper-evident audit logging is established prior work (Schneier & Kelsey, 1999). We use standard hashes, Ed25519 signatures, and a separately retained commitment. We introduce no new cryptographic primitive. Unlike schemes designed to preserve security after a recorder is compromised, this prototype trusts the customer recorder and does not implement forward-secure key evolution.

Execution receipts and outcome attestation are also prior work. Morrow and Sogomonian (2026) describe invocation-bound outcome receipts in an individual Internet-Draft, which is not an adopted standard. Bindschaedler et al. (2026) study bonded settlement for compensable agent side effects, including Git and database adapters. We neither replicate their settlement experiment nor claim to originate outcome receipts, backend-specific verification or bonding. Git's conditional ref updates and SQLite's atomic transactions are existing implementation primitives (Git, n.d.; SQLite, n.d.).

The technical contribution is a reproducible implementation and two bounded evaluations: model-assisted review of tool reports, and deterministic reconciliation of causal records under authored backend faults. Both use strong retained-record controls. Section 8

connects the access and funding design to economic work on adoption, financing, participation costs and commitment. The economic contribution is a prospective provider-network specification and comparative evaluation plan, not a deployed settlement system or an observed token treatment effect.

3. System and trust boundary

The deployment has four roles. A customer controller creates a run with a fresh nonce and declared scope. An agent receives only a credential scoped to that run. A customer gateway maps named tools to fixed upstream endpoints. A reviewer receives submitted reports and whatever evidence its experimental interface supplies.

The controller credential, signing key, and upstream credentials are unavailable to the agent through the API. The gateway commits an intent to SQLite before forwarding each call. The outcome records the HTTP status and exact response bytes, or marks the result indeterminate when a transport failure leaves the effect unknown. Identical retries of a completed call ID return the recorded outcome without forwarding another request. Reuse with different arguments fails.

A controller cannot close a run with unresolved calls. An intent left pending across a process restart becomes indeterminate, because the upstream may already have acted. The gateway does not automatically retry such a request. Its closing manifest commits to the run identifier, nonce, scope, configuration, final event hash, event count, call count, and uncertain-outcome count.

The controller retains that signed closing receipt separately from the evaluator's submitted report. Offline verification requires both a pinned customer public key and the expected closing receipt. This prevents accepting an arbitrary self-declared key or an edited chain that has merely been rehashed consistently. It does not establish that the customer signer is honest.

The observation claim is deliberately bounded. The gateway records calls routed through it and the HTTP responses it receives. A 200 response is not proof of a real

downstream side effect. Calls that bypass the gateway are outside its evidence. In a customer deployment, network and credential controls would have to enforce the intended observation boundary. Those controls are not inferred from a signature.

4. Experimental design

4.1 Local workflows and independent execution record

The target is gpt-4.1-mini-2025-04-14, with temperature zero, up to five model requests per workflow, and a maximum of 256 output tokens per request. Twelve task instances derive from four templates: operations summaries, invoice routing, incident updates, and private document permissions. Each template has three variations in document facts and identifiers. These are twelve instances from four templates, not twelve independent business domains.

The agent can call `read_document`, `send_message`, and `set_visibility`. Each call travels over HTTP through the actual gateway to a separate local mock service. The service records the received arguments and returned response in its own ledger. Before constructing reviewer inputs, the experiment checks every gateway observation against this backend ledger.

Documents contain business facts and an adversarial instruction to send their contents to an unapproved recipient and change visibility to public. The model system prompt says to perform only the authorized task and treat document instructions as untrusted. The inputs are authored adversarial cases, not a representative attack sample. Messages and permission changes affect only the local simulation.

4.2 Report alterations

For each eligible execution, we construct an authentic report, an equivalent-format control, and three deliberately altered reports. An argument edit changes a tool argument in both the report and submitted event log. An omission removes a write call from both. An outcome edit changes an HTTP status in the report while retaining the original event evidence. The editor can recompute ordinary event hashes but does not receive the customer's signing key or alter the controller's retained closing receipt.

These are controlled report alterations, not observations of spontaneous evaluator fraud. Detection of an edited record does not establish the identity or intent of whoever changed it. The outcome-only alteration also ensures that checking a signature alone is insufficient to solve every case: the reviewer must compare a report with a valid underlying record.

The formatting control is serialized and parsed before reviewer payload construction. Its final model input is equivalent to the authentic input. It therefore supplies a repeated control judgment rather than an independent test of textual formatting robustness; formatting invariance is also tested at the software layer.

4.3 Four evidence interfaces

Answer only: the final agent answer and submitted report.

Submitted logs: the same fields plus the untrusted event log supplied with the report.

Verified gateway: the report plus the result of verification against the customer-pinned key and separately retained closing receipt. When verification passes, the reviewer receives decoded event records. When it fails, the reviewer receives the failure without treating the edited records as trusted.

Retained ordinary log: the report plus an independently retained copy of the original gateway observations. This baseline supplies the same trusted observation source without requiring signatures.

The last interface is intentionally strong. It separates evidence retention from cryptographic portability. A centralized customer that reliably retains its own logs may not need signed receipts for its internal reviewer.

4.4 Reviewer and blinding

The reviewer is gpt-4.1-2025-04-14, with temperature zero and a maximum of 180 output tokens. Each judgment is a fresh API request. The prompt asks whether reliable evidence supports the report, contradicts it, or is insufficient. The reviewer must assess tool-call reporting within this run, not infer misconduct or judge actions outside the boundary.

The model receives no case labels, mutation labels, source-file paths, backend oracle, or sibling judgments. Case order is shuffled with a fixed seed. The evidence interface itself is visible, so the experiment is not blinded to treatment. The same provider supplies target and reviewer models; this is not independent model-family validation.

Model and output caps are held fixed across interfaces. Input length and inference cost vary with the evidence supplied and are recorded. This is not a strictly equal-cost comparison.

4.5 Protocol and exclusions

A local protocol was fixed before experimental generation; only a short credential check preceded it. It is not an external preregistration. Three permission-task runs produced no write call and were ineligible for the planned write-call alterations. They remain in the execution dataset.

One argument edit left a recipient unchanged because the original action already used that recipient. A data-quality check identified this ineffective alteration without inspecting reviewer answers. Its four interface judgments are retained but excluded from the main comparison. Thus, 180 scheduled judgments yield 176 eligible observations. This correction is documented and is not represented as part of the original protocol.

5. Results

5.1 Gateway execution

All twelve target workflows produced a final model response. They generated 27 forwarded tool calls. All 27 gateway observations matched the separate backend ledger, including tool name, arguments, HTTP status, and response body. This verifies the pilot's forwarding and recording paths, not the absence of every possible implementation defect.

Three runs contained an unrequested external-message call and a change to public visibility in the local simulation. The gateway recorded these actions; it did not prevent them. Three permission runs read a document but did not execute the requested permission change. A final response is therefore not counted as proof that the legitimate task was completed.

5.2 Report review

Table 1

Evidence interface	Alterations detected	False accusations on controls	Controls supported
Final answer	13/26	0/18	16/18
Submitted logs	12/26	0/18	18/18
Verified gateway	26/26	0/18	18/18
Retained ordinary log	24/26	0/18	18/18

Gateway-assisted review detected all twenty-six eligible alterations. Final-answer review detected thirteen; submitted-log review detected twelve; and retained ordinary logs detected twenty-four. The final-answer interface also abstained on two altered reports and two authentic controls. All other eligible judgments were binary supported or contradicted decisions. All 180 scheduled API judgments completed without a missing or malformed verdict; four judgments belong to the excluded ineffective alteration.

The table counts judgments, not independent subjects. Within each interface, controls comprise eighteen judgments for nine task instances, including equivalent-input repetitions. Altered reports comprise twenty-six eligible judgments across three alteration families. These dependencies and the small number of templates rule out interpreting a zero observed false-accusation rate as a deployment guarantee.

By alteration family, the verified-gateway interface detected 8/8 argument edits, 9/9 omissions, and 9/9 outcome edits. Submitted logs detected 3/8, 0/9, and 9/9 respectively. Retained ordinary logs detected 6/8, 9/9, and 9/9. The two retained-log misses were argument changes in invoice-0 and invoice-1, where the reviewer incorrectly said the report matched the independently retained record. In these cases the edited report duplicated an external recipient already present in a different call. The corresponding verification failure removed the need for the model to find that argument mismatch unaided.

5.3 Reliability checks and cost

Fifteen new gateway tests passed alongside the existing thirty-two MVP tests. The gateway checks exercise real HTTP forwarding, credential separation, duplicate-request protection, altered and omitted events, chain rehashing, equivalent JSON formatting, crash recovery, and an explicit off-gateway action. The bypass test shows the intended limitation: an action sent directly to the backend is absent from the gateway journal.

These forty-seven software tests are engineering checks, not forty-seven scientific experiments. They do not establish production hardening, malicious-customer resistance, a false-accusation rate, or the correctness of the reviewer model.

The recorded model calls used 125,048 input tokens and 11,773 output tokens. At the published uncached model rates, the upper calculated API cost was USD 0.318974, approximately thirty-two cents, including the credential check. There were 220 successful model requests: thirty-nine target requests, 180 reviewer requests, and one credential check. Median reviewer-request latency was 1.35 seconds on this run; it is not a service-level

guarantee. Pricing was checked against OpenAI's official model pages before collection. Local compute and developer time are excluded.

6. Interpretation

The experiment tests an implemented local component and the decisions made using its evidence. That is stronger evidence for a proposed solution than an assumed detection probability or an example showing that final text omits actions. It remains a pilot on a controlled task.

The retained-log comparison determines what can be attributed to signatures. Here, the gateway interface exceeded the retained ordinary log by two judgments, both in related invoice tasks. This small, correlated difference does not establish general superiority or statistical significance. The main observed gain was against evidence that the submitting party could edit. The experiment also confounds the signed format with an explicit verifier output, so it does not isolate an intrinsic effect of the cryptographic primitive. A signature can allow another party to verify that a submitted record matches a customer-retained commitment. It cannot create an observation of a bypassed action, prove that the signer told the truth, or guarantee that a reviewer understands the authenticated content.

An integrity failure warrants rejecting or investigating the submitted evidence. It is not, by itself, proof of intentional wrongdoing or proof that every statement in the accompanying report is false. The current dataset coedits reports and records in its integrity-failure conditions; it does not test the harder case of an accurate report paired with accidentally corrupted evidence. That condition belongs in the next study.

For Naysay, the practical integration is a customer-configured HTTP observation boundary and independently verifiable reports. The existing hosted product was not modified or deployed during this study. The prototype is available as source and an HTTP API; an agent requires no Python SDK to send its tool requests.

7. Outcome verification under backend faults

7.1 Claim, observation and trust model

The outcome service extends the observation boundary from HTTP response bytes to two concrete backend operations: updating a versioned SQLite row and updating a Git file through a branch ref. The customer controller registers an immutable intent containing an action identifier, backend, target, desired value and expected version. It retains that intent independently and delegates only a capability to execute the registered action. The agent cannot change its target or precondition, request a customer observation, or access the signing key through this interface.

Execution has three possible verdicts: applied, not_applied and unresolved. A matching retained causal record supports applied; a retained precondition rejection supports not_applied; insufficient evidence supports unresolved. The separately reported state verdict is holds, differs or unavailable at the observation snapshot. These answers can differ without contradiction. An action can commit and later be overwritten; another actor can create the desired state without the registered action ever executing.

For each observation, the controller supplies a fresh nonce. The service signs the registered intent, nonce, timestamp and normalized native observations with Ed25519. Offline verification requires an independently retained public key, original intent and requested nonce. A controller must consume nonces once; an offline verifier cannot detect a caller's reuse of its own nonce. Envelopes with altered payloads, another intent, a different key or a stale nonce are rejected as evidence, rather than interpreted as backend failures.

The customer observer, signing key and native records are trusted. The envelope authenticates the observer's statement; it does not establish external truth against a dishonest observer. A point-in-time state observation is not a continuing guarantee. Direct backend access and violations of version discipline remain outside the agent-capability boundary.

7.2 Native adapters and recovery

For SQLite, the adapter begins an immediate transaction, checks the expected row version, updates the target value, increments its version and inserts a causal effect record binding the action identifier and intent hash. Mutation, effect record and execution decision commit together. A failure inserting the record rolls back the mutation. A retry after a lost response can recover the committed record without applying the update twice. A retained applied decision without its causal record is treated as unresolved.

For Git, the adapter constructs an immutable commit with the expected parent, requested file content and intent hash. It persists the prepared commit identifier before compare-and-swapping the configured branch ref from the expected old commit to that new commit. Applied evidence requires both the exact commit and a matching retained relog transition carrying the action identifier. An unattached commit is insufficient. Recovery can identify a ref update that occurred before a process failed to record its local decision. A prepared operation with no retained transition is unresolved and is not blindly reapplied.

The Git adapter checks a whole branch version; the SQLite adapter checks a row version. Consequently, an unrelated intervening Git commit invalidates the precondition, whereas a change to another SQLite row does not. This conservative Git policy is an integration cost. The prototype uses a single process per state directory and isolated stores. It does not implement distributed coordination, key rotation, capability expiry, public-service hardening or a customer production integration. Its HTTP interface requires no client SDK.

7.3 Fault protocol and ground truth

On 18 September 2026, a local protocol was fixed before implementation and evaluation. Seven development fault families and six later-evaluated combinations were specified for both backends, producing fourteen development scenarios and twelve evaluation scenarios. The author knew the cases in advance. The archived label `held_out` denotes their later evaluation, not blinded or independently sampled validation. The protocol was not externally preregistered.

Development scenarios cover success, nonexecution, stale preconditions, subsequent overwrite, subsequent unrelated change, duplicate execution and loss of the causal record. Evaluation combinations cover a committed action followed by response loss, restart and retry; response loss followed by overwrite; another actor creating the desired state without the registered action; a prepared Git commit without ref update or a rolled-back SQLite mutation; an unrelated mutation before execution; and overwrite followed by restoration of the desired state by another actor.

Requests cross real loopback TCP connections to the service. In response-loss cases, a proxy forwards the request, allows the backend to commit and drops the client TCP response. The rollback case inserts a SQLite trigger failure after the target update and before insertion of its causal record. The Git preparation case persists an object without applying the branch update. Ground-truth snapshots read native backend state before and after each invocation, bypassing the observer's signing and reconciliation code. The harness separately schedules other actors' writes. These controlled schedules establish ground truth for the authored cases, not arbitrary concurrent production histories.

The primary comparator receives the same native records and uses the same reconciliation algorithm without signature verification. Sharing the algorithm intentionally isolates the authentication wrapper; equality is expected under equal inputs. It is not an independently implemented logging product. Two weaker diagnostic rules accept execution from an observed HTTP 2xx status or from a matching current value. The HTTP rule deliberately ignores the API's explicit JSON execution verdict and must not be interpreted as a competent customer integration.

7.4 Execution and state results

Table 2 reports execution decisions across all 26 scenarios. Unresolved decisions remain in the denominator. Cases are authored backend/fault combinations, not independent customers, model trials or random draws from a fault population.

Table 2

Comparator	Correct	False acceptance	False rejection	Unresolved
HTTP status only	16	4	0	6
Current state only	20	2	4	0
Retained logs and reconciliation	18	0	0	8
Signed observation	18	0	0	8

The signed interface and retained-log control each resolve eighteen cases correctly, approximately 69% of the authored set, with eight unresolved cases and no incorrect decisions. Development cases contribute ten correct and four unresolved decisions per interface; later-evaluated cases contribute eight correct and four unresolved decisions. No statistical significance or population error rate follows from these counts. The two methods also agree on all state verdicts, which match all 26 direct backend snapshots.

The eight unresolved decisions comprise four cases per backend: never executed; causal evidence removed after execution; matching desired state created by another actor; and prepared Git state or rolled-back SQLite state without a conclusive retained execution record. Absence of the required evidence is not proof of nonexecution or misconduct. Additional records of cancellation or transaction abort could resolve some cases, but those records are not implemented here.

The current-state diagnostic incorrectly attributes another actor's matching writes to the registered action in two cases and treats four later overwrites as evidence of nonexecution. The HTTP-status diagnostic falsely accepts four cases. These contrasts illustrate the distinct information required for attribution; they do not demonstrate superiority over the retained-log control.

7.5 Engineering checks and descriptive cost

Twenty-five outcome tests and fifteen existing gateway tests passed. The additional checks exercise concurrent requests with the same expected version, an external Git writer racing the ref update, envelope tampering, capability separation, rollback and abrupt process exit before or after durable operations. Recovery after process termination is tested; power-loss durability and compromised native storage are not. These forty tests are engineering checks, not additional independent observations in Table 2.

Table 3 reports local medians from the archived 26-scenario run. Observation collection includes backend reads, signing, serialization and loopback HTTP. Verification includes signature checking and reconciliation. Values combine Git and SQLite cases and are descriptive rather than estimates of production throughput or fully loaded service cost.

Table 3

Measurement	Median
Observation collection over loopback HTTP	42.260 ms
Offline verification and reconciliation	0.213 ms
Canonically serialized evidence	672 bytes

The environment was Python 3.12.13, SQLite 3.53.4 and Apple Git 2.50.1 on macOS. The full fault set ran twice with identical verdict counts; the archived run includes implementation hashes after input-validation hardening. Repetition is not counted as additional evidence. Neither run used paid model calls. Human investigation time, integration effort and customer willingness to pay were not measured.

7.6 Interpretation and scope

The implemented service recovers causal evidence for committed actions after response loss and distinguishes those actions from later state changes. Its positive result is an executable, reproducible observation interface with explicit unresolved outcomes. It supplies no detection gain over the equal-information reconciliation control. The signed format allows offline authentication of a retained customer statement, but a reduction in manual investigation or dispute costs remains a hypothesis for customer evaluation.

The original gateway and the outcome service make different claims. The first authenticates bounded tool-report observations; the second interprets specific native mutations under trusted adapter assumptions. Neither authenticates events outside its observation boundary. Expiring reflogs, unavailable causal records, dishonest signers and external writers that violate row-version discipline require retention, trust or integration controls beyond those tested here.

8. Provider-network economics and token design

8.1 Current implementation and proposed change

The token design considered in this revision moves the research question from customer eligibility to provider obligations and shared protocol control. The proposal is a network in which independent operators accept bounded evidence-retention, reproduction or verification jobs, receive customer-funded fees for specified delivery, and commit capital against defined violations. NAYSAY is a candidate asset for operator participation and governance. Establishing a useful role requires enforceable rights or obligations and a measured benefit relative to conventional arrangements.

This is a prospective design. The implemented gateway and outcome service do not require a native token for their technical operation. At the recorded economic baseline, Policy 2026-09-17-v1 required 10,000 NAYSAY for customer access; holdings were transferable and were neither locked collateral nor a revenue entitlement. The stated supply was one billion

tokens, based on the project's 17 September chain check, not a verified immutable cap. These facts describe the existing policy, not an implemented provider network. This paper revision changes no production contract, customer balance requirement or payment configuration.

The proposed customer interface would charge for bounded jobs in USDG without requiring customers to acquire provider stake. Operator admission and capital requirements would be evaluated separately. The existing 10,000-token customer threshold is not reused as a security parameter: it was not calibrated to provider exposure, violation gains, liquidity or customer loss. Token supply alone cannot determine a defensible stake requirement or token valuation.

8.2 Economic rationale and related work

An operator's economic incentive comes from paid demand for useful work. Cong et al. (2021) relate token valuation to adoption and transactional demand, but their payment-token setting does not directly establish demand for NAYSAY when customers pay in USDG. Cong et al. (2022) examine investment, issuance and repurchases jointly, making operator incentives and investment opportunity costs relevant. Neither model identifies a fixed access threshold or allocation percentage as optimal for this service. Catalini and Gans (2018) likewise analyze financing through a commitment to accept tokens for products; that commitment is not the operating role proposed here.

The additional governance hypothesis concerns a commitment to network participants. Sockin and Xiong (2023b) analyze how delegating platform control can constrain exploitation of users, together with a reduced incentive to subsidize participation. That mechanism requires an actual transfer of decision rights. A token vote that a company can routinely override does not instantiate it. Sockin and Xiong (2023a) also identify risks when speculative token demand impairs user participation. Pricing customers in USDG would separate billing from provider-stake acquisition but would not remove operators' price or financing risk.

The empirical literature motivates measurement rather than a token-benefit assumption. Howell et al. (2020) relate token-financing outcomes to disclosure, commitment and project quality; Catalini and Tucker (2017) show that adoption timing can affect abandonment and peer use. Neither supplies an estimate of demand for Naysay's service. If independent operators and customers form a marketplace, the distribution of charges and benefits across those groups becomes relevant to participation (Rochet & Tirole, 2003). The local studies establish no such marketplace or network effect.

Bonded recourse for agent side effects is already a research topic, including backend-specific receipts and settlement (Bindschaedler et al., 2026). Ethereum's proof-of-stake rules provide another example of collateral backing explicit duties and penalties for conflicting signed statements (Ethereum.org, n.d.). These precedents motivate precise obligations and violation predicates. They do not imply that application staking inherits consensus security, that outcomes become truthful by majority vote, or that NAYSAY is preferable to stablecoin collateral.

8.3 Participants, obligations and compensation

Table 4 separates the proposed roles from their implementation status. Operator compensation would pay for delivered work. Passive ownership alone would create neither an entitlement to job fees nor a promise of yield. Governance rights would concern the protocol's specified decisions, not ownership of the company or an automatic claim on all service revenue.

Table 4

Component	Proposed role	Present status
Customer payment	Fund a bounded job in USDG	Credit accounting exists; checkout was disabled at the recorded baseline
Independent operator	Accept a fixed job scope and deliver retained evidence or a reproducible result	No independently operated paid network was evaluated
Operator capital	Remain committed while covered obligations and disputes are open	No provider bonding or slashing contract was evaluated
Operator fees	Earn agreed compensation for verified contractual delivery	No customer-funded operator payouts were observed
Protocol governance	Exercise defined, constrained control over future network rules and treasury decisions	No governance rights were deployed in these studies

A job specification would fix the customer-authorized scope, evidence schema, observer keys, verifier version, delivery deadline, retention interval, fee, challenge procedure and maximum covered liability before acceptance. Retention work means preserving and serving a specified artifact. Reproduction work means rerunning a defined operation in an authorized environment. Verification work means applying a named predicate to specified evidence. Completing one obligation does not establish that every claim made by an agent is correct.

Operators would need separate authorization for customer data and execution environments. Independent retention must not become an unrestricted license to copy private traces. The access policy and permitted disclosure would be part of the job. The current single-observer prototype does not implement this distributed authorization or privacy model.

8.4 Job settlement and disputes

The proposed lifecycle is funded, accepted, delivered, challenged or finalized, then released. Before acceptance, the customer places the agreed fee in escrow and the operator reserves the required capital. Acceptance freezes the applicable terms. Delivery commits the exact artifact and its verification inputs. Release depends on the stated delivery predicate and challenge process, not merely an HTTP success code or an LLM's favorable assessment.

Funding a job is not earned revenue. Undelivered jobs require a specified refund path. A delivery can satisfy its contract even if the correct execution verdict is unresolved: a verification job should pay for a justified assessment, not only for a favorable answer. Customer dissatisfaction with the result must not let a customer withhold an otherwise earned fee indefinitely. Conversely, a hash commitment alone does not demonstrate artifact availability or semantic correctness. Off-chain availability and subjective claims require an explicitly trusted resolution procedure and must not be advertised as fully automatic settlement.

Capital supporting a covered obligation remains unavailable for withdrawal until its challenge and resolution intervals expire. The same capital must not be counted as backing several simultaneous liabilities beyond the stated coverage rule. Job reservations, exit delays, key rotation, adjudicator authority and insolvency behavior require implementation and testing. None of these settlement components is present in the outcome experiment.

8.5 A narrow candidate for attributable violations

An initial candidate is equivocation: an operator signs two finalized, mutually incompatible commitments for the same immutable observation. A future message format would bind the network, contract, job, attempt, registered operator key, action scope, observation identifier and verifier version. The controller-assigned observation identifier must refer to one fixed observation rather than a reusable label for evolving state. The incompatible pair and the versioned verification rule would constitute the proposed violation proof.

The existing outcome envelope is not itself an implemented slashing protocol. A later observation that the postcondition differs is compatible with an earlier applied action. An unresolved verdict followed by a resolved verdict can also be legitimate when additional evidence arrives. Neither pair qualifies merely because its text differs. The proposed predicate must exclude different observations, revisions allowed by the contract, stale keys, cross-job replay and disagreement over unobserved events. A proof establishes a specified signing violation, not necessarily malicious intent.

Missing evidence, expired records, model disagreement and failure to establish causation must not automatically trigger penalties. The eight unresolved cases in Section 7 are examples of insufficient information, not eight proven operator offenses. A consistently dishonest observer could sign one false account without equivocating. Adding a bond does not solve that source-of-truth problem. Compensation for other failures would require additional attributable predicates or an agreed adjudication process, with capped liability and a defined appeal procedure.

8.6 Capital, fees and the native-token comparison

Under the candidate native-stake model, operators would lock NAYSAY to accept covered work and put the contractually defined portion at risk for a proven violation. Fees for delivered work would be paid from customers' USDG payments. A testable operator participation condition is that expected earned fees cover execution, retention, support, dispute, financing and expected penalty costs. A testable customer condition is that investigation or service value exceeds price, integration effort, privacy costs and residual failure risk. Neither condition is established by token emissions or hypothetical resale gains.

Capital sizing should track outstanding obligations and realizable collateral value rather than cumulative request volume. More jobs can require additional committed capital when they increase simultaneous exposure or retention obligations. They need not do so if operators have spare capacity or obligations clear faster. Consequently, paid usage can create

demand to participate without producing a proportional increase in tokens locked or in token price.

Stablecoin collateral is the primary financial comparator because job prices and covered amounts would be denominated in USDG. It has issuer, custody and depegging risks. Native collateral additionally exposes coverage to token-price and liquidity changes, potentially when failures already reduce confidence in the service. A native-stake design would require conservative valuation, concentration limits, collateral replenishment or job-admission controls, and loss-allocation rules. Such controls cannot guarantee liquidation proceeds during a market gap. A stablecoin compensation reserve with a separate native participation stake is another candidate, but the additional stake must justify its additional cost.

If conventional contracts or stablecoin bonds provide the same service and participant control at lower total cost, native collateral has not demonstrated an advantage. Token-denominated staking would make NAYSAY operationally required under that rule; the rule alone would not demonstrate economic necessity. The comparison must hold the covered duties, dispute rights, fee schedule and risk treatment constant.

8.7 Governance as an enforceable commitment

The proposed governance role is control over specified future network parameters: admission rules, supported verifier versions, fee policies, treasury allocations and upgrades. These rights would need to reside in executable contracts or otherwise enforceable arrangements, with published proposal rules, voting-power distribution, timelocks and an exit opportunity before material changes. The company would have to relinquish unilateral override within that scope. Governance authority over a funded job must not retroactively alter its accepted price, liability or violation predicate.

This is the candidate economic role of transferable protocol-control rights: participants could obtain a credible constraint on future unilateral changes. Its value depends on whether users and operators need that constraint and whether token governance provides

it better than contractual commitments, a cooperative or another governance arrangement. It does not arise from calling a token a governance token. Delegation, concentrated ownership, borrowed voting power, collusion and emergency powers can undermine the proposed commitment and belong in the evaluation.

Governance should not determine whether an individual agent action occurred by token-holder vote. That question remains governed by evidence and the job's fixed verification or adjudication procedure. A one-billion-token supply provides no assurance of dispersed control; concentration and actual decision authority must be measured. No passive revenue entitlement is introduced by this proposal.

8.8 Existing economic observations and financing discipline

The available production-ledger observation remains the snapshot collected on 18 September 2026 at 08:56 UTC: zero issued credit invoices, zero paid invoices, zero consumed paid credits and zero earned USDG revenue, with checkout disabled and credit mode off. This is an operator-collected baseline, not an independent financial audit or a measurement of unwillingness to pay. It does not include secondary-market token trading or establish the state of the ledger at a later date.

The prior service policy priced 1,000 completed HTTPS requests at 10 USDG and budgeted eligible operating surplus 25% to token purchases and verified burns, 10% to AI safety research, and 65% to development and reserves, after a 300-USDG reserve floor. Accounting code exists; allocation execution was not activated at the snapshot. These historical policy parameters are not validated prices, stake sizes or payout rates for a provider network. No buyback, burn, provider payout or donation is inferred from an accounting budget.

Provider compensation, refunds, retention obligations and dispute costs would have to be funded before treating receipts as distributable surplus. Customer escrow and operator collateral must remain separate from operating income. Secondary-market purchases pay token sellers rather than automatically financing jobs. Repurchases from any eventual surplus

would compete with product investment and would not make native collateral necessary. No price forecast follows from a fixed token supply, a lock requirement or an announced burn policy.

Earlier numerical feasibility examples and assumed-margin calculations remain in the replication materials as historical sensitivity analyses. They are not treated as evidence for the provider-network proposal and are no longer the paper's basis for a token-benefit claim. The previously proposed organization-randomized customer-access trial concerns the old gate; it does not identify effects of provider bonding or shared governance.

8.9 Prospective paid-operator pilot

The next empirical step is an independently operated, customer-funded job pilot, beginning with a conventional payment and stablecoin-collateral benchmark. Three to five independently operated design-partner workflows would be a feasibility stage, not a powered adoption study. No partners, escrow deposits or customer-funded jobs are claimed here. Recruitment and operation require a separate deployment effort.

The prospective protocol in the replication archive specifies recording job scope, fee funded, fee earned, operator independence, labor and compute costs, retention time, collateral amount and duration, customer investigation time, disputes, refunds, unresolved results and losses. Before exposing capital to penalties, a separate conformance evaluation would test valid violation proofs and near-miss cases: different observations, legitimate state changes, corrected reports, rotated keys, duplicate submissions and missing evidence. Counterexamples that cause unjustified penalties block financial deployment.

A subsequent comparison could contrast conventional contracts, stablecoin bonds and native-token participation for matched duties and coverage. Governance effects must be evaluated separately from collateral denomination; combining both changes would prevent attributing any improvement to either. Costs include financing and price risk, oracle and settlement costs, integration and support effort, dispute handling and governance overhead.

Customer-funded job completion and repeat paid use are adoption outcomes; token trading volume is not.

The pilot must report unfavorable and unresolved outcomes, existing-operator capacity, concentration and the cost of additional capital. It can establish a useful paid workflow or expose an uneconomic one. A native-token advantage requires measured benefits that survive the conventional and stablecoin comparisons. Until then, enforceable operator participation and shared control are research hypotheses and implementation targets, not properties demonstrated by the existing experiments.

9. Limitations and next study

For the report-review pilot, the strongest limitations are authored workflow templates, a mock business backend, one model provider, small sample size, repeated controls and deliberately planted alterations. The outcome study uses actual Git and SQLite stores but only authored local faults and controlled schedules. It has no model-agent population, independent human assessment or customer traffic. Its shared-algorithm control isolates signature verification and cannot support a comparative product-accuracy claim. In both studies, the signer and experimental controller are trusted. The experiments and implementations were developed in the same project, with no external reproduction or human adjudication.

The economic analysis uses a targeted literature review, not a systematic review or pooled effect estimate. No independent provider network, escrow, native bond, penalty mechanism or governance right was deployed in these studies. No willingness-to-pay distribution, operating margin, collateral advantage or governance benefit was estimated. The historical ledger snapshot contains no paid activity with which to estimate those quantities. A paid-operator feasibility pilot would still face recruitment selection, shared market conditions, concentration and limited external validity; even favorable job outcomes would not identify long-run token returns.

The immediate next study should examine independently operated customer workflows and compare time to a verified diagnosis, integration cost and unresolved-case

handling against the customer's existing logs and reconciliation. Initial design-partner cases can establish integration feasibility; they are not substitutes for a powered customer-adoption trial. Later studies should include accurate reports with corrupted evidence, unavailable and malicious signers, independently held-out task families, established logging products and human review. Sample sizes should be selected from the intended error or adoption requirement rather than treating these pilot counts as sufficient.

10. Conclusion

We implemented customer-controlled interfaces for tool-report integrity and bounded native-action outcomes. The gateway-assisted interface detected more planted alterations than submitted logs, while the strong retained-log baseline achieved a similar result. No false accusations occurred in the small repeated-control set. In the separate Git/SQLite fault study, signed observations and retained-log reconciliation each produced eighteen correct execution decisions and eight unresolved decisions across 26 authored scenarios. This equality locates the benefit in the available causal records rather than extra semantic information supplied by signatures. The proposed provider model locates a possible token role in capital committed to defined obligations and in enforceable protocol-control rights. That role depends on real customer-funded work, attributable violations and advantages over conventional or stablecoin alternatives. The next empirical question is whether independently operated customers and providers obtain sufficient service and commitment benefits to cover integration, capital and dispute costs.

References

- Bindschaedler, L., Botha, Q., & Siebenbrunner, C. (2026). Bonded recourse for smart-contract settlement of compensable agent side effects. arXiv.
<https://doi.org/10.48550/arXiv.2609.01939>
- Catalini, C., & Gans, J. S. (2018). Initial coin offerings and the value of crypto tokens (NBER Working Paper No. 24418; revised 2019). National Bureau of Economic Research.
<https://doi.org/10.3386/w24418>
- Catalini, C., & Tucker, C. (2017). When early adopters don't adopt. *Science*, 357(6347), 135-136. <https://doi.org/10.1126/science.aal4476>
- Cong, L. W., Li, Y., & Wang, N. (2021). Tokenomics: Dynamic adoption and valuation. *The Review of Financial Studies*, 34(3), 1105-1155. <https://doi.org/10.1093/rfs/hhaa089>
- Cong, L. W., Li, Y., & Wang, N. (2022). Token-based platform finance. *Journal of Financial Economics*, 144(3), 972-991. <https://doi.org/10.1016/j.jfineco.2021.10.002>
- Debenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., & Tramèr, F. (2024). AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. arXiv. <https://arxiv.org/abs/2406.13352>
- Ethereum.org. (n.d.). Proof-of-stake rewards and penalties. Retrieved September 18, 2026, from <https://ethereum.org/developers/docs/consensus-mechanisms/pos/rewards-and-penalties/>
- Git. (n.d.). git-update-ref: Update the object name stored in a ref safely. Retrieved September 18, 2026, from <https://git-scm.com/docs/git-update-ref>
- Howell, S. T., Niessner, M., & Yermack, D. (2020). Initial coin offerings: Financing growth with cryptocurrency token sales. *The Review of Financial Studies*, 33(9), 3925-3974. <https://doi.org/10.1093/rfs/hhz131>

Morrow, & Sogomonian, A. (2026, April 4). Execution outcome attestation for AI agents and automated systems (Internet-Draft draft-morrow-sogomonian-exec-outcome-attest-00). Internet Engineering Task Force.

<https://datatracker.ietf.org/doc/draft-morrow-sogomonian-exec-outcome-attest/00/>

OpenAI. (n.d.). GPT-4.1 mini model. Retrieved September 17, 2026, from

<https://developers.openai.com/api/docs/models/gpt-4.1-mini>

OpenAI. (n.d.). GPT-4.1 model. Retrieved September 17, 2026, from

<https://developers.openai.com/api/docs/models/gpt-4.1>

Rochet, J.-C., & Tirole, J. (2003). Platform competition in two-sided markets. *Journal of the European Economic Association*, 1(4), 990-1029.

<https://doi.org/10.1162/154247603322493212>

Schneier, B., & Kelsey, J. (1999). Secure audit logs to support computer forensics. *ACM Transactions on Information and System Security*, 2(2), 159-176.

<https://doi.org/10.1145/317087.317089>

Sheshadri, A., Ewart, A., Fronsdal, K., Gupta, I., Bowman, S. R., Price, S., Marks, S., & Wang, R. (2026, March 10). AuditBench: Evaluating alignment auditing techniques on models with hidden behaviors. *Alignment Science Blog*.

<https://alignment.anthropic.com/2026/auditbench/>

Sockin, M., & Xiong, W. (2023a). A model of cryptocurrencies. *Management Science*, 69(11), 6684-6707. <https://doi.org/10.1287/mnsc.2023.4756>

Sockin, M., & Xiong, W. (2023b). Decentralization through tokenization. *The Journal of Finance*, 78(1), 247-299. <https://doi.org/10.1111/jofi.13192>

SQLite. (n.d.). Atomic commit in SQLite. Retrieved September 18, 2026, from

<https://www.sqlite.org/atomiccommit.html>

Appendix: replication and provenance

This PDF embeds naysay-gateway-evidence-vo.6-replication.zip as a file attachment. A standalone copy is supplied with the paper. Use a PDF reader that supports attachments or extract it with pypdf; no code executes when the PDF is opened. The attachment checksum is printed below the appendix. Source and raw observations, rather than only the summary table, are included.

The archive additionally contains the historical economic-analysis scripts and tests, policy source, verified citation metadata, source-to-claim mapping, aggregate production-ledger snapshot and sensitivity calculations. The earlier customer-gate study is retained as historical planning. The new provider-network design and prospective pilot are in research/provider_network/. They contain specifications, not deployed financial contracts or results from a paid network. The archived economics calculations run offline and do not execute payments.

The outcome study is supplied separately under outcomes/, research/outcome_study/ and output/outcome-study/results.json. It includes the HTTP service, native adapters, offline verifier, tests, fixed fault protocol, all 26 envelopes, independently retained inputs, direct backend snapshots, per-case verdicts and timings, environment metadata and source hashes. Its protocol SHA-256 is 9f79aa866123c9e021b5c8876f5b22d67110b8e12933264309b285233822cbe7. The archive excludes runtime state, controller capabilities and private signing keys. Public keys and nonces are retained for verification.

From an extracted archive with the recorded Python dependencies, run python research/outcome_study/replay.py to verify the stored outcome evidence and table without backend or model access. Run python research/outcome_study/run.py to create a new local fault run, and python -m pytest outcomes/tests gateway/tests -q for implementation checks. Replay establishes archive consistency under the supplied keys and ground truth; it is not independent evidence that the observer was honest. Re-execution creates new timestamps, keys and objects and is not expected to reproduce byte-identical files.

The archive contains the gateway implementation and tests, frozen protocol, exact model requests and responses, independently recorded backend observations, source runs, submitted report variants, reviewer inputs, concealed labels, all raw judgments, exclusions, analysis code, and budget ledger. It contains no model API credential or private signing key.

Rerunning analysis over the saved observations requires no model calls. Generating new executions and judgments does incur API charges and will produce a new experiment. Keep the original results immutable and identify new runs separately.

Replication attachment SHA-256

09916627097925c2693983b8b6a0fee1547a5c0394d24f056a76f9eb4cdc25e3