

Noticer Outcome Verification

John Patrick Daly

Noticer

Systems design paper | Version 0.3 | 4 October 2026

Author review draft | Source and evidence cutoff 4 October 2026 | Not approved for submission or publication

Abstract

Agent completion reports can diverge from destination state. This paper describes Noticer’s bounded outcome-verification mechanism at a pinned implementation version. A frozen obligation binds a subject identity, scalar equality predicate, observation source, control resource, deadline, and continuation policy. The evaluator distinguishes a satisfied condition, a supported negative result, and unavailable or insufficient evidence; a receipt records the basis for that decision. The paper presents source-derived decision rules, separates authenticated receipt custody from signature verification and downstream execution, and examines one retained signed package-metadata case. Previously recorded data-only checks support consistency of the historical package under its bundled key, not independent verification of the remote observation. No new benchmark or live workflow experiment is reported. An evaluation protocol specifies independently checked destination truth, failure cases, baselines, and downstream readback. The result is an inspectable implementation account and a bounded evidence case, with the question of operational benefit left for prospective evaluation.

1 Introduction

An agent saying “done” is not enough to establish that the intended outcome happened. External research documents false-success behavior and limitations in LLM-based detection. Noticer’s purpose is to check a specific expected outcome against observable evidence and return an inspectable receipt.

External motivation: Advani [1].

This implementation and evidence study uses Noticer Platform commit 8fc38914 as its source baseline and a separate historical signed package as its worked case. Source inspection, retained observations, recorded verification, and proposed evaluation are distinguished throughout. No new application tests or live workflow experiments were run for this paper.

An agent’s completion message describes what the agent believes it accomplished. The destination system may expose a different state. A successful request, an accepted job, and the intended result are distinct observations. A useful verification system therefore needs an explicit account of the condition that matters and the evidence capable of establishing it.

Consider an illustrative provisioning workflow. A payment service records a successful charge, but the buyer still cannot access the product. Checking payment status answers a payment question. Checking entitlement requires a separate subject, source, predicate, and time window. Even a correct entitlement record may leave another question open: whether the user can actually exercise that entitlement. The verification boundary must match the decision being made.

1.1 External motivation

Advani’s study reports false success among failed agent trajectories. Section 4.1 gives 45% for airline, 48% for retail, and 3% for dual-control telecom on tau2-bench. AppWorld’s 75.8% is 1,425 of 1,879 failures with explicit status claims in two self-assessing architectures, `full_code_refl` and `ipfuncall`. It is not a percentage of all agent runs. No evaluated tau2-bench judge configuration exceeded AUROC 0.65; the best AppWorld result was approximately 0.54. The judge tasks differ: false versus true success on tau2-bench, and false success versus honest failure on AppWorld. [1]

The source contains numerical inconsistencies: its introduction gives 44–52%, the abstract gives 45–48%, and Table 1 gives retail as 47% rather than Section 4.1’s 48%. Those summaries should not be treated as interchangeable. The results motivate outcome checks; they do not measure Noticer or establish that a particular verification design will work. [1]

1.2 The decision boundary

Noticer’s general JSON path checks a declared scalar equality for an identified record. It does not infer unrestricted success from prose. A requester or configured integration must select the relevant destination, identity, rule, and known-good control. An incorrect choice can produce a technically correct answer to the wrong question. The design makes that choice inspectable; it cannot make every choice appropriate.

This paper concerns the selected mechanism and its evidence. It does not claim that Noticer can connect to every application, repair arbitrary failures, certify source honesty, or authorize actions on behalf of a user.

1.3 Contribution and research scope

This paper contributes an implementation-grounded account, rather than a new general verification algorithm or a demonstrated improvement in agent performance. It makes three concrete elements available for scrutiny: the selected observation and verdict contract; the trust boundary between a stored result, a portable signed envelope, and a downstream consequence; and a traceable historical case accompanied by a prospective evaluation protocol. The account is scoped to the named source generation and the evidence actually retained.

The open research question is whether the frozen predicate and explicit uncertainty policy improve decisions in a specified workflow compared with its existing completion criterion and a capable alternative checker. Section 6 states the measurements needed; the historical case is not a benchmark of that benefit.

1.4 Related work

Runtime verification examines observed executions against declared correctness properties. Leucker and Schallhart discuss online and offline monitoring and multi-valued verdicts for finite observations. Noticer’s three-way vocabulary is therefore not itself a novel verification principle. The selected mechanism uses bounded reads of external records and a control resource; it is not a general temporal-logic monitor, and its verdict definitions are not asserted to be equivalent to a temporal-logic semantics. [15]

The software-testing oracle problem concerns how expected behavior is established and compared with observations. Barr and colleagues survey specification-based and other oracle approaches, including the difficulty of connecting abstract expectations to concrete outputs. A Noticer predicate functions as a narrow specified oracle. Freezing it makes the chosen question auditable but does not solve oracle selection or establish that a selected field captures the user’s broader goal. [16]

Receipt custody is also related to provenance. The W3C PROV data model describes entities, activities, and agents involved in producing data so that its origin can be assessed. Noticer’s receipt records scope,

observations, and processing associated with a verdict. This paper claims no PROV conformance. Retaining that record aids inspection while leaving source authenticity, source truth, and the relevance of the predicate as separate questions. [17]

Advani’s false-success study supplies agent-specific motivation for examining destination state. It evaluates agent behavior and LLM-based judges in its own benchmark settings; it does not compare Noticer with those judges. Together, these lines of work motivate a bounded implementation study without establishing novelty, superiority, or measured operational benefit for the system described here. [1]

2 Observation contract

The central object is a proof obligation: a specified question with bounded evidence requirements. In Platform’s generic core, the executable predicate is the required outcome. The core rejects a separately worded outcome that differs from the canonical predicate; it does not ask an LLM whether the two expressions mean the same thing. The obligation hash binds the claim reference and type, required outcome, observer, questioned and control specifications, deadline, policy, and any configured freshness window. [2]

The contract needs the following information before observation:

- **Subject identity.** The claim reference must identify the questioned resource. The observer checks both URL binding and an identity field in the returned JSON.
- **Executable condition.** A supported JSON field path is compared with a scalar literal. The rule must express the actual decision criterion rather than a convenient proxy.
- **Observation authority.** A configured observer profile determines permitted origins and observation semantics. Credential availability is an operational dependency, not evidence that the outcome is true.
- **Control.** A distinct known-good resource provides a comparison for the observation path. Its expected condition must be meaningful for the questioned resource’s failure modes.
- **Time.** The obligation has a deadline. Evidence must remain fresh under the worker’s configured window. A result arriving after the deadline cannot establish success within it.
- **Policy.** For the generic frozen core, both DISPROVED and INCONCLUSIVE map to BLOCK. Uncertainty is not configurable to ALLOW. [2–4]

2.1 Verdicts under the selected contract

The following table summarizes Platform’s generic worker with a valid frozen obligation. It is not a universal definition for every Noticer source generation.

Verdict	Required observation state	Generic policy
PROVED	Questioned and control predicates are true, evidence is fresh, and completion is within the deadline	ALLOW
DISPROVED	At the deadline, the questioned predicate is false, the control is true, and evidence is fresh	BLOCK
INCONCLUSIVE	The obligation closes without the evidence required for either definitive verdict, or core integrity fails	BLOCK

Table 1. Condensed source contract for Platform 8fc38914. Before the deadline, an unsuccessful check may remain pending and be polled again. The worker does not immediately turn every mismatch into a terminal failure. [2,3]

An unavailable read is materially different from a false predicate. Platform treats wrong identity, malformed JSON, inappropriate content type, and most non-200 responses as unavailable evidence. A resource-specific observer profile may explicitly interpret a questioned 404 as absence; that exception does not apply to the healthy control. Without such semantics, a missing or unreadable response cannot establish that the business object is absent. [4]

The control is also a limited instrument. A syntactically different resource is not necessarily independently maintained. A shared outage may disable both reads; a selective failure may affect only the questioned resource. The control helps interpret observations under the selected contract. It is not a causal counterfactual or an independent certification of the provider's truthfulness.

2.2 Compact contract and decision procedure

For the generic path, let O denote the stored obligation and let q and c be the questioned and control predicate results. Each result is true, false, or unavailable. The following field summary is derived from `core_contract.py`, `worker.py`, `observer.py`, and `gated_worker.py` at Platform 8fc38914. It is a conceptual record description, not a replacement API schema. [2–4]

Obligation O contains `claim_ref`, `claim_type`, `required_outcome`, `observer`, `questioned`, `control`, `deadline_seconds`, `policy`, `core_contract`, and `obligation_hash`. The questioned specification contains `url`, `extract`, and `claim_id_path`; the control contains `url` and `extract`. An optional `freshness_window_seconds` is included in the hash when present. The `required_outcome` and `questioned.extract` must be the identical canonical executable predicate.

An observation records `status`, `predicate_result`, `http_status`, `reason`, `observed_at`, `evidence_hash`, `url`, `extract`, and `claim_id_path`. The credential-free commercial profile additionally retains `response_body_base64` and `response_content_type`. The generic receipt combines the obligation identity and binding, both observations, `core_integrity_valid`, `verdict`, `gate`, `policy`, `creation` and `deadline` times, `completion` time, `attempt` count, and `receipt_hash`. The commercial signature envelope is separate from that receipt.

Algorithm 1 summarizes the decision branch after a bounded observation attempt, or reuse of the latest recorded pair. D is the deadline, t is the worker decision time, and F means that the latest completed observation is within the effective freshness window. A result arriving after D is made unavailable before this branch. The effective window is the obligation's integer `freshness_window_seconds` when present, otherwise the polling interval plus one second. [2,3]

```
if frozen binding is invalid: close INCONCLUSIVE, BLOCK
else if q is true and c is true and F and t <= D: close PROVED, ALLOW
else if t >= D and q is false and c is true and F: close DISPROVED, BLOCK
else if t >= D: close INCONCLUSIVE, BLOCK
else: persist the pair and schedule another check no later than D
```

Algorithm 1. Source-derived summary of the generic verdict branch. An absent or stale pair is treated as unavailable. Pending is a work state, not a fourth terminal verdict. Scheduling, lease ownership, observer validation, and final receipt integrity checks remain part of the implementation outside this summary. The ordering preserves the source's affirmative rule at the exact deadline. This pseudocode was not executed as an experiment.

3 Mechanism and trust boundaries

Platform separates declaration, observation, receipt storage, notification, and continuation acceptance. This separation matters because a valid result can be lost in delivery, and a delivered notification can be forged or replayed. It also prevents a later explanation from quietly replacing the condition that was originally accepted.

3.1 Processing sequence

1. Validate and freeze the obligation. Canonicalize the supported predicate, require agreement with the required outcome, enforce the fixed policy, and compute the binding hash.
2. Persist the work. Store the obligation in a durable queue. A worker acquires an expiring fenced lease so that an older worker cannot overwrite a newer owner's result.
3. Recheck authority before observation. The hardened worker checks the frozen binding. Unfrozen generic work and integrity failures cannot obtain affirmative authority.
4. Observe the questioned resource and control. Origin-scoped HTTPS reads run under a bounded network budget. The observer evaluates identity and the selected predicate; the worker applies freshness and deadline rules.
5. Commit the receipt. The record retains the question, observations, times, verdict, policy result, and integrity fields. Callback delivery is a separate retryable operation.
6. Resolve continuation through its own integration. A waiting consumer retrieves the authenticated stored result and uses the acceptance protocol. A receipt delivered for inspection alone installs no downstream gate. [2–6]

These operations are present in the pinned source. Their presence does not establish current credentials, successful execution in a particular deployment, or end-to-end correctness under every crash and provider failure. The queue is a single-host SQLite design, not evidence of a distributed consensus system.

3.2 Three different integrity contracts

Generic owner receipts use a checksum under authenticated server custody. A checksum allows consistency checks, but an attacker who can replace both data and checksum can recompute it. The authenticated stored record remains the trust anchor for that integration.

The commercial public-JSON path adds an Ed25519-signed evidence envelope and retains source-response bytes. The signature binds a payload to a key. Useful verification still requires a separately trusted public key, the expected scope, and an acceptable observation time. A signature cannot independently establish that a remote fetch happened as described or that a provider returned truthful data. [5,7]

ACCESS receipts use a separate HMAC contract. HMAC and public-key signatures have different trust and distribution properties; neither should be silently substituted for the generic checksum contract. Moreover, Platform's ACCESS adapter remains fixture-only. Its non-fixture path reports that the live access source is not configured. A successful charge therefore cannot be presented as a verified live entitlement in this implementation. [8]

3.3 From a verdict to a consequence

ALLOW is a policy result within the receipt. It is not proof that the next action occurred or permission to take every possible action. The generic waiting endpoint checks tenant-bound stored evidence, requested identity fields, integrity, and a 300-second acceptance window measured from completion. Its consequence record prevents a repeated acceptance from issuing a second immediate continuation: reuse returns a hold that requires downstream reconciliation. [6]

That design does not by itself prove exactly-once execution at an external destination. A consumer can fail after recording acceptance but before performing the action, or perform an action before losing its acknowledgement. The integration needs its own idempotency and destination readback. The evaluation in Section 6 treats acceptance, execution, and observed consequence as separate events.

3.4 Assumptions and failure model

The mechanism assumes that the frozen condition is relevant to the intended decision, that the selected source has the required semantics, and that the control is an appropriate diagnostic resource. Origin restrictions and identity checks narrow the observation scope; they do not establish that a provider's records faithfully represent a wider real-world outcome.

The deployment must protect the stored obligation, observation process, clock, receipt store, and signing material where used. A malicious or compromised issuer can manufacture mutually consistent observations. A portable verifier also needs an independently chosen trusted public key and an expected scope; accepting the key supplied beside a receipt does not solve issuer authentication. These dependencies limit what any successful integrity check establishes.

The questioned and control reads are sequential and do not form an atomic snapshot of the provider. State may change between those reads, after receipt creation, or before a consumer acts. Freshness limits the permitted age of evidence but does not lock the destination state. Where a consequence requires a stronger invariant, the integration needs a suitable conditional operation or another destination check under its own contract.

The source includes defenses for malformed responses, stale evidence, identity mismatch, expired work, and repeated acceptance. These are implementation mechanisms, not a complete adversarial security argument. This paper supplies neither a formal correctness proof nor measurements of coverage against compromised providers, compromised hosts, or arbitrary concurrency failures. [2–7]

4 Current implementation and version boundaries

The 4 October source comparison identifies Platform 8fc38914 as the current general-engine baseline. Its runtime retains the frozen generic core and adds commercial evidence delivery and static agent discovery. Source inspection supports the mechanisms described above, including retained public response bytes and the signed envelope. Discovery documents do not show that an arbitrary adapter is supported or that a customer workflow is ready. [2–9]

Source line	Role in this paper	Boundary
Platform 8fc38914	Current general-engine source baseline	Generic contract and commercial receipt path
Daly 12fa08c7	Separate live Shopify and evidence-host branch in the dated source map	Different observer and verdict semantics
Reference 41945b1c	Separately initialized offline example	Deterministic evaluation and replay, not a hosted production gate
September 25 portable package	Historical worked evidence case	Preserved signed observation; not a fresh Platform run

Table 2. These sources are not interchangeable releases of one validated experiment. Full identities and access qualifications appear in the references and Appendix A. [9–11]

One concrete difference prevents silently merging the two engine lines. With a healthy control and fresh evidence at the deadline, Daly's selected branch can classify a 2xx invalid-JSON or wrong-content-type response as CONTENT_MISMATCH and then DISPROVED. Platform leaves that condition unavailable

and therefore unresolved. Both block under fail-closed policy, but the receipt says something different about what is known. A consumer must use the contract for the selected line. [4,10]

The broader historical Daly package also contains dedicated Zapier actions and installation evidence that are absent from current Platform. Some corresponding server routes remain. Their historical presence does not establish that those integrations are currently usable. The local report-only patch is unshipped and is not counted as current capability. Likewise, the offline reference package must not be described as the original production repository. [9–11]

The source map records successful deployment status and running replicas for the two engine services at its observation time. This paper does not repeat those control-plane checks as an outcome experiment. A running service, a passing historical CI run, and a verified customer consequence answer different questions.

4.1 Source custody and separate checkpoint

The source inspected for this paper is a locally retained, reconstructed GitHub snapshot of zensteagarden/noticer-platform at commit 8fc38914f7d4d38ac2d6fbb67203b6d391377776, held in the author's review workspace. Retained GitHub commit and tree retrieval records bind it to tree 8527aaa618d1d7f40f32d3a6ab929e903624b64f. On 4 October 2026 at 06:47 UTC, a passive byte comparison confirmed that all 92 local files match the Git blob identities in that retained tree. This is a comparison against retained retrieval records, not a fresh GitHub fetch or execution result. [9]

Direct source inspection on 4 October covered the generic core, worker and hardened worker, observer, waiting endpoint, commercial receipt envelope, ACCESS adapter and receipt code, relevant SQLite storage operations, and Zapier waiting consumer. The claims in Sections 2 and 3 refer to those source paths at the pinned commit. Historical deployment observations remain those recorded by the earlier source comparison; they were not repeated for this revision.

The separately supplied external review identifies noticer-grok-build-handoff-06, checkpoint local-postgres-foundation-06, commit 1e2a3f62439f6651ead5651c188a9a93b2ba28e, as PAUSED_BY_OWNER with its native PostgreSQL gate not run. That is a user-supplied separate checkpoint and source lineage, not the Platform tree inspected here. Its bytes and execution state have not been independently verified for this paper. Nothing in this paper promotes, merges, or claims to have run that checkpoint.

5 Evidence available today

The worked case is a retained public package-metadata observation dated 25 September 2026. Its value is the ability to inspect the relation among the question, stored source bodies, and signed result. The predicate is deliberately narrow and low-stakes; its simplicity makes the evidence easier to examine but limits the conclusion.

5.1 Historical case and retained artifacts

The questioned URL identifies Flask version 3.1.3 on PyPI. The identity path is \$.info.version, the claim reference is 3.1.3, and the required predicate is \$.info.name == "Flask". The control is the Flask 3.1.2 record on the same provider, checked against the same name predicate. The confirmed scope specifies a 60-second deadline and BLOCK for both negative and unresolved outcomes. [12; private retained files, not public downloads]

The signed receipt records both responses as HTTP 200 with true predicates, one attempt, PROVED, and ALLOW. The aggregate observation time is 25 September 2026 at 19:03:46.901710 UTC. The archive

retains both complete JSON snapshots, the signed envelope, the public key, and the confirmed scope. These are retained issuer statements and data, not newly observed provider responses.

Retained field	Value or interpretation
Subject and identity	Flask 3.1.3; \$.info.version equals the claim reference
Questioned condition	\$.info.name == "Flask"
Control	Separate Flask 3.1.2 JSON record; same name predicate
Observation time	25 September 2026, 19:03:46.901710 UTC
Recorded verdict	PROVED; policy field ALLOW
Evidence custody	Signed envelope, public key, scope, and two response bodies retained
Downstream effect	Package explicitly states that no workflow gate or downstream authorization is installed

Table 3. Annotated transcription of selected historical receipt fields, not a full receipt or executable schema. Names, scope, and verdict come from the retained package. Issuer endpoints, order identifiers, and signature bytes are omitted here; artifact digests in Appendix A identify the original files. [12; private retained files, not public downloads]

5.2 What the recorded checks establish

A data-only review on 3 October recorded successful Ed25519 verification against the bundled key. It also recorded agreement between the decoded payload and its readable representation, matching hashes for both embedded response bodies, a matching receipt checksum, and a scope hash consistent with the confirmed obligation. All 32 entries in the package checksum manifest matched. Re-evaluation of the two retained name fields returned true. [13; private retained review files, not public downloads]

Those checks concern the internal consistency of retained bytes. They were not rerun for this paper and are not independent replication of the original service run. The original receipt and scope were read for the present account, along with the dated verification output. The snapshot and artifact hashes permit future comparison with the same evidence, without importing or executing the archived application.

The conclusion is specific: the package contains a cryptographically consistent, inspectable record of the declared metadata claim under its bundled key. The key's issuer has not been independently established by this archive. Substituting both a receipt and its bundled key could preserve local signature validity. Similarly, signed HTTP status and timestamps do not independently prove the reported fetch or its timing.

5.3 What the case does not establish

The package's historical acceptance record describes a 900-second verification freshness window. That interval has expired. The record cannot authorize a current decision merely because its signature still verifies. A new decision about changing destination state needs an appropriately fresh observation.

The retained package contains no signed negative or unresolved experiment, no installed continuation gate, and no raw downstream execution trace. The backend commit named by its acceptance report is not supplied as a matched backend checkout in this package. The case therefore remains separate from the current Platform baseline, despite similarities in fields and mechanism. Its payment assertions are unsigned test-mode reports, not demonstrated independent customer purchases. [12,13; private retained files, not public downloads]

A separate historical HubSpot brief reports an authentication-failure and recovery sequence. It says its full successful receipt still needed archiving. Reported success should not be elevated to a fully inspectable end-to-end case until that evidence is recovered. [14; private retained report, not a public download]

6 Evaluation still required

A useful next evaluation would test one real decision boundary with a fixed implementation and a selected destination. The following is a proposed protocol, not a completed experiment or a product commitment. It is designed to distinguish correct predicate evaluation from actual protection of the workflow.

6.1 Freeze the question and ground truth

Choose one workflow, one engine commit, one observer profile, and one downstream consumer. Declare subject identifiers, expected predicates, allowed evidence sources, deadlines, freshness windows, and continuation rules before collecting results. Preserve the exact configuration and artifact digests. Obtain authorization for any destination writes or live transactions separately.

An independent readback or controlled destination state should establish ground truth for each trial. A provider's own status field may be sufficient for a narrow record question but insufficient for a broader customer outcome. Where the two share a failure mode, record that dependence instead of counting them as independent evidence. Keep incomplete or contradictory truth records in the report rather than dropping difficult cases.

Compare the configured Noticer gate with the existing workflow's completion criterion and a capable alternative checker given comparable information. If an LLM judge is included, preserve its model version, prompt, evidence access, and decision threshold. Compare methods on the same cases. The external study motivates such a comparison but supplies no baseline performance for this deployment.

6.2 Exercise failures and recovery

The test set should include genuine success, false destination state with a healthy control, wrong subject identity, unavailable source, shared control failure, stale evidence, malformed response, and a result arriving after the deadline. Integrity cases should alter scope, body bytes, or signatures and include an untrusted key. Integration cases should cover duplicate notification, repeated acceptance, worker restart, and loss of acknowledgement around an external action.

For each case retain the expected state, actual source observation, complete receipt, consumer decision, and independently observed downstream effect. An acceptance response alone is not the endpoint. For negative and unresolved cases, attempt the normal continuation path and record whether the protected action actually ran. For a positive case, verify that the intended action ran against the correct subject under the caller's authorization.

Recovery should be a new, linked observation rather than a rewrite of the failed record. Preserve the original failure, any repair, the new scope if it changes, and the subsequent verdict. This permits a reviewer to distinguish a repaired system from a relabeled result.

6.3 Report denominators and acceptance criteria

Report false acceptance as ALLOW decisions among cases whose independently established destination condition is false. Report false rejection as definitive DISPROVED decisions among true cases, with INCONCLUSIVE outcomes reported separately. Also report the fraction of true cases that fail to obtain

ALLOW, so abstention does not conceal operational cost. Publish each numerator, denominator, and ground-truth rule.

Report inconclusive outcomes across all attempted checks and by failure cause, plus latency from accepted obligation to terminal verdict, deadline misses, observation cost, and downstream action counts. Keep repeated retries and related mutations grouped by underlying case; they are not independent trials. Include confidence intervals suited to the sampling design rather than treating zero observed failures as zero risk.

Before execution, the workflow owner should set the permitted false-acceptance bound, acceptable unresolved rate, latency limit, and what happens when evidence is unavailable. Acceptance should require both correct receipt behavior and the required destination effect. A pilot that only demonstrates receipt parsing leaves the operational question unanswered.

7 Limits and conclusion

Noticer's selected core constrains the relation between a declared predicate, observed records, and a continuation policy. Its receipt can make that relation inspectable. The design remains dependent on the relevance of the predicate, correct subject selection, suitable control, trustworthy observation path, bounded timing, and the consumer's handling of the result.

The current evidence supports a precise implementation account and one retained signed metadata case. It does not establish a general detection rate, universal application coverage, autonomous repair, independent issuer trust, or improved business outcomes. The available historical package cannot stand in for a current workflow trial, and a current source checkout cannot stand in for historical execution evidence.

The next decisive question is whether one configured workflow prevents an incorrect consequence while allowing the correct one, under observed destination truth and the failures that workflow actually faces. That question can be tested with preserved scope, complete receipts, and separate downstream readback. Until then, the appropriate claim is bounded outcome verification with inspectable evidence.

AI assistance and author review

OpenAI AI assistance was used to draft and edit this paper, analyze documentation and source code, compare source and artifact records, check citations, and prepare the document layout. Tool-assisted passive byte and hash comparisons support the custody statements; they are distinct from the historical verification results and from application testing. No new application test suite, benchmark, or live workflow experiment was run for this paper. AI-assisted analysis is not independent replication. John Patrick Daly is responsible for the final content, including its claims, citations, and interpretation of evidence. This version remains pending his final human-author review and approval before submission or publication.

References

Repository links identify exact commits and files. Their inclusion does not assert public access; permissions may be required. Retained evidence entries identify reviewable artifacts held by the author and are not public download links.

[1] Laksh Advani. From Confident Closing to Silent Failure Characterizing False Success in LLM Agents. arXiv 2606.09863v1, 1 June 2026. Methods 3.2 and 3.5, Section 4.1, and Tables 1–2. <https://arxiv.org/html/2606.09863v1>

[2] Noticer Platform. Frozen generic core. Commit 8fc38914f7d4d38ac2d6fbb67203b6d391377776, server/core_contract.py.
https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/server/core_contract.py

[3] Noticer Platform. Worker verdict and timing rules; hardened integrity enforcement. Same commit, server/worker.py and server/gated_worker.py.
<https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/server/worker.py>

Hardened worker:
https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/server/gated_worker.py

[4] Noticer Platform. Identity-bound observation and transport. Same commit, server/observer.py.
<https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/server/observer.py>

[5] Noticer Platform. Durable work and receipt storage. Same commit, server/storage.py.
<https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/server/storage.py>

[6] Noticer Platform. Waiting acceptance. Same commit, server/waiting.py. <https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/server/waiting.py>

Waiting consumer:
https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/zapier/creates/wait_for_outcome.js

[7] Noticer Platform. Portable evidence envelope. Same commit, server/receipt_package.py.
https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/server/receipt_package.py

[8] Noticer Platform. ACCESS adapter and receipt contract. Same commit, server/access_adapter.py and server/access_receipt.py.
https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/server/access_adapter.py

ACCESS receipt:
https://github.com/zensteagarden/noticer-platform/blob/8fc38914f7d4d38ac2d6fbb67203b6d391377776/server/access_receipt.py

[9] Noticer source comparison and canonical map. Dated read-only review, 4 October 2026, approximately 05:45 UTC. Retained GitHub commit and tree retrieval records plus a local 92-file snapshot. Tree 8527aaa618d1d7f40f32d3a6ab929e903624b64f; passive blob comparison repeated 4 October at 06:47 UTC. Current Platform tree: <https://github.com/zensteagarden/noticer-platform/tree/8fc38914f7d4d38ac2d6fbb67203b6d391377776>

- [10] Noticer Daly. Separate observer and worker. Commit 12fa08c7fee28d86fc781bcd41871e04bc9829d3. <https://github.com/zensteagarden/noticer-daly-railway-1.1.0/blob/12fa08c7fee28d86fc781bcd41871e04bc9829d3/server/observer.py>
- Daly worker:
<https://github.com/zensteagarden/noticer-daly-railway-1.1.0/blob/12fa08c7fee28d86fc781bcd41871e04bc9829d3/server/worker.py>
- [11] Noticer Reference. Trust and limitations. Commit 41945b1ccaf9305e6f8508ce491ebb5b19ef9aab. <https://github.com/Noticer-io/noticer-reference/blob/41945b1ccaf9305e6f8508ce491ebb5b19ef9aab/docs/TRUST-AND-LIMITATIONS.md>
- [12] Noticer portable archive. START_HERE_Noticer_Portable_2026-09-27(1).zip. Historical receipt dated 25 September 2026; signed-receipt.json, public-key.json, and confirmed-scope.json under website/proof. Original artifact identities are recorded in Appendix A. Retained privately; no public artifact access is asserted.
- [13] Portable Noticer archive recovered evidence and boundaries. Data-only review dated 3 October 2026, with portable-verification-results.json. Records signature, snapshot, receipt, scope, and package-integrity checks. Retained review output; not a historical service receipt or independent live replication.
- [14] Noticer HubSpot brief. Historical live-acceptance account dated 16 September 2026, page 6. Names frozen Platform commit 224344af5b0cc5f482b00fdb2f6b1722c0f00094 and states that the full successful receipt remained to be archived. Retained report; not raw execution evidence.
- [15] Martin Leucker and Christian Schallhart. A Brief Account of Runtime Verification. Journal of Logic and Algebraic Programming 78(5), 293–303, 2009. DOI 10.1016/j.jlap.2008.08.004. Author-hosted full text: <https://christian.schallhart.net/publications/2009--jlap--a-brief-account-of-runtime-verification.pdf>
- [16] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The Oracle Problem in Software Testing A Survey. IEEE Transactions on Software Engineering 41(5), 507–525, 2015. DOI 10.1109/TSE.2014.2372785. Author-hosted full text: <https://philmcminn.com/publications/barr2015.pdf>
- [17] Luc Moreau and Paolo Missier, editors. PROV-DM The PROV Data Model. W3C Recommendation, 30 April 2013. <https://www.w3.org/TR/2013/REC-prov-dm-20130430/>

Appendix A Evidence identity and inspection limits

The following SHA-256 digests identify retained original files. They are not signatures, public-access guarantees, or hashes of text extracted from a different format. The identities and previously recorded check results come from the 3 October data-only review. [12,13; private retained files, not public downloads]

Portable archive

6076607ec7544908dd789b94ff017a80e79cfed8e067c4e0a4e4a7eaa7dc4fe0

Signed receipt JSON, 45,093 bytes

50b1bae1d26764b00de9f19a3166e74153dcbe77cb60421aeb9cb359afeb5906

Public key JSON, 176 bytes

708ed702eae2607f3be4ddc5e5a35196022701e70041585af49e6624f8ef1e0e

Confirmed scope JSON, 689 bytes

edfc3af618e0eef867b95a161f97669357c941fe59aff83201a6749500b2e3c9

Questioned response bytes, 5,088 bytes

782593ae20369a820d57d3311d04d54f805e40262b95448ce325a8cae182c354

Control response bytes, 7,158 bytes

8e4bce173151dc1f691b22390c3afb323cb97dae11dcf8d6a9d078ce7b841031

An authorized reviewer with the original package can compare artifact bytes, inspect the saved scope, decode the retained bodies, and evaluate the stated name predicate. Signature verification must use the exact signed payload bytes and an independently selected trust anchor. Local agreement with the bundled key establishes consistency with that key only. Freshness and permission for a present-day consequence require separate decisions.

The original ZIP bytes are retained in the author’s review workspace. A passive SHA-256 comparison on 4 October at 06:47 UTC matched the archive digest above; this did not rerun historical signature verification. The original backend and historical verifier implementation are not included in this package. The paper makes no claim to have reproduced their operation. It also excludes private service endpoints, credentials, customer data, and unsigned payment assertions from the worked evidence display.