Beyond Periodic Compliance: An Evidence-Driven Architecture for Continuous SOC 2 Control Assurance in AI-Enabled Cloud Environments

Sachin Suryawanshi

Solutions Architect, Even & Odd Minds LLC, United States
sachinmcm009@gmail.com | ORCID: 0009-0001-8464-908X

Abstract:
Cloud controls can change faster than the evidence assembled to assess them. Artificial intelligence adds further uncertainty when agents modify software, invoke tools, or help interpret compliance records. This paper proposes an evidence-driven architecture for continuous internal assurance of selected SOC 2 controls. The architecture binds each assessment to a resource population, control version, observation time, collection history, and executable predicate. It distinguishes supported, violated, unknown, and approved-exception states, while retaining historical findings and the evidence available when decisions were made. A design-science method is used to construct the model and evaluate its decision semantics with executable synthetic cases. Twelve synthetic scenarios and 576 combinations of evidence gates, predicate values, and exception conditions were evaluated. In a separate constructed population of 1,000 assessment obligations, excluding missing evidence produced a 90 percent observed-only pass rate; retaining the full denominator yielded 80 percent evidence coverage and 72 percent supported obligations. An idealized timing experiment illustrates the dependence of detection delay on collection frequency and delivery assumptions. These results establish reproducibility and internal consistency within the model, not effectiveness in a deployed environment. The contribution is a conservative, temporally explicit assessment contract that connects cloud telemetry to reviewable control claims without granting AI authority to certify compliance or replace an independent SOC 2 examination.

Keywords: SOC 2; Continuous Control Monitoring; Evidence Provenance; Cloud Security; AI Governance; Compliance as Code

1 Introduction

A deployment approval proves that a change was authorized at a particular point. It does not establish that the deployed artifact matches the approved build, that its identity retains the intended permissions, or that subsequent configuration changes remained within policy. These distinctions become operationally important when cloud resources are short-lived and software agents can initiate changes. An evidence record may be authentic yet irrelevant to the current resource, policy, or assessment period. Increasing the quantity of collected records does not resolve that mismatch.

System and Organization Controls (SOC) 2 examinations address controls relevant to applicable Trust Services Criteria (TSC) [1]. A Type 2 examination includes consideration of operating effectiveness over a specified period. The phrase “beyond periodic compliance” in this paper concerns the organization's

evidence collection and internal assessment practices. It does not imply that SOC 2 examines only isolated snapshots, or that an automated score can express an auditor's opinion. Continuous internal assurance means maintaining timely, traceable support for specifically bounded control claims between formal reviews.

The research question is: how can an architecture maintain defensible control states when cloud configuration, artificial intelligence (AI) behavior, and evidence availability change independently? The proposed answer is an assessment contract that makes evidence admissibility explicit, preserves unknown states, and reconstructs results against the population and rule version applicable at the time. Its contribution is the combination of these semantics with a practical separation between AI assistance, deterministic evaluation, and accountable human decisions. No claim of universal novelty for continuous monitoring, evidence graphs, or compliance as code is made.

2 Related Work and Research Position

Continuous auditing predates contemporary cloud services. Chan and Vasarhelyi [2] describe changes to audit practice enabled by automation and more frequent assessment. NIST Special Publication 800-137 [3] supplies a continuous-monitoring foundation concerned with assets, threats, vulnerabilities, and deployed controls. These works establish why monitoring frequency and organizational response matter. The present problem is narrower: defining the meaning of a current control result when some supporting observations are delayed, incompatible, or absent.

The Open Security Controls Assessment Language (OSCAL) already provides machine-readable assessment plans, observations, findings, and risks [4]. It can represent continuous assessments and result expiration. Cilla Ugarte et al. [5] propose an OSCAL-based AI compliance architecture linking policy, evidence, and enforcement. Their preprint demonstrates that machine-readable AI assurance is an existing research direction. This paper builds on that direction through resource-level admissibility rules and explicit population accounting; it does not propose a replacement interchange format or claim that existing platforms lack these capabilities.

The NIST Generative AI Profile [6] broadens the risk context for AI systems. OWASP's prompt-injection guidance [7] explains why external text can influence model behavior and why privilege boundaries remain necessary. A related contribution by Suryawanshi is also relevant. The Specification Governance and Validation Framework [8] connects agentic software changes to requirements and release decisions. Here, release evidence is treated as one input to continuing operational assessment. This earlier paper does not establish the effectiveness of the architecture proposed here.

3 Research Method and Scope

The study follows the design-science sequence of problem definition, objective setting, artifact construction, demonstration, and evaluation described by Peffers et al. [9]. A targeted literature review covered continuous auditing, NIST monitoring guidance, machine-readable assessment, and AI security. It was not a systematic review or an exhaustive comparison of commercial products. The artifact consists of an assessment contract, a reference architecture, four-state decision rules, and a small executable model. Evaluation uses constructed inputs whose expected behavior is specified directly from the contract.

Five requirements guide the design. R1 requires every claim to identify its resource scope and control version. R2 prevents inadmissible or missing evidence from becoming a passing result. R3 separates exception approval from evidence of successful operation. R4 retains both observation time and receipt time

for historical reconstruction. R5 prevents an AI assistant from approving its own evidence or changing an authoritative verdict. The model addresses selected technical and hybrid controls. Governance, personnel, supplier, and other qualitative controls still require appropriate human assessment. No customer records, employer systems, auditor judgments, or production incident data are used in the evaluation.

4 Evidence Contract and Assessment Semantics

4.1 Evidence and Population Boundaries

A control definition records its identifier, applicable TSC area, owner, version, effective interval, resource-selection rule, required tests, freshness limits, and exception authority. Each observation records its source identity, tenant and resource identifiers, test identifier, predicate result, event time, receipt time, rule version, collector version, and a digest of the retained payload. Evidence about an AI-mediated change additionally identifies the agent principal, model deployment, tool permission set, approval, and resulting artifact. Model or prompt identifiers describe configuration; they do not establish that an action was safe.

The assessment population is established independently of successful evidence retrieval. Resource inventories are reconciled with deployment records and account boundaries, including deleted resources relevant to the period. Otherwise, a collector can silently narrow the denominator by failing to discover what it cannot observe. A failed inventory reconciliation makes the population itself uncertain; the system reports that limitation instead of presenting a complete percentage. Discovered resources enter the assessment population when they become in scope, not when their first passing observation arrives.

4.2 Admissibility and Four States

An observation is admissible only when its identity and integrity checks succeed, its scope and rule version match, its collector is healthy for the relevant interval, and its observation age is within the control-specific limit. It must also have been received by the decision time. Future-dated, malformed, unresolved contradictory, or indeterminate observations are inadmissible. Freshness is measured from observation time, so replaying an old record cannot refresh it. A payload hash detects alteration relative to a trusted digest; it does not authenticate a dishonest source. Collector credentials, clock bounds, and protected manifests are separate trust assumptions.

For control c, resource r, and time t, the evaluator applies the following precedence across required tests. A known, admissible failure produces VIOLATED, or EXCEPTION when every known failed obligation is covered by a valid, scoped approval. In the absence of known failures, any unresolved required test produces UNKNOWN. SUPPORTED requires admissible passing evidence for every required test. EXCEPTION never means that the failed predicate passed. Missing tests remain listed even when a known failure determines the headline state. Conflicting observations for the same authoritative resource generation are unresolved until their ordering or provenance is established.

Table 1: Decision States and Required Follow-up

State	Evidence Basis	Required Follow-up
SUPPORTED	All required tests pass with admissible evidence.	Continue collection; expire support when its basis lapses.
VIOLATED	At least one known failure lacks valid exception coverage.	Open or retain a finding; assign remediation.

State	Evidence Basis	Required Follow-up
UNKNOWN	No known failure; at least one required test is unresolved.	Resolve evidence gaps; retain earlier open findings.
EXCEPTION	All known failures have active, scoped approval.	Track expiry and residual risk; retain unresolved tests.

State changes do not delete findings. If a failing observation becomes stale, the current assessment may become UNKNOWN, but the earlier violation remains open until a separately recorded closure is justified. A successful retest establishes a later condition and cannot rewrite the earlier interval. Policy revisions likewise create new effective versions. A historical result is reproducible using the original policy, population, evidence, and decision time, subject to the retained trust assumptions.

4.3 Coverage and Temporal Accounting

For an assessment period, let O contain the scheduled resource-test obligations actually in scope. State checks can use fixed time slots; event-triggered controls require an independently reconciled event population. The four counts nS, nV, nU, and nE correspond to SUPPORTED, VIOLATED, UNKNOWN, and EXCEPTION, with N equal to their sum. Each obligation is counted once. The following quantities are internal management measures, not SOC 2 ratings:

$$\text{Evidence coverage } C = \frac{N - nU}{N} \quad (1)$$

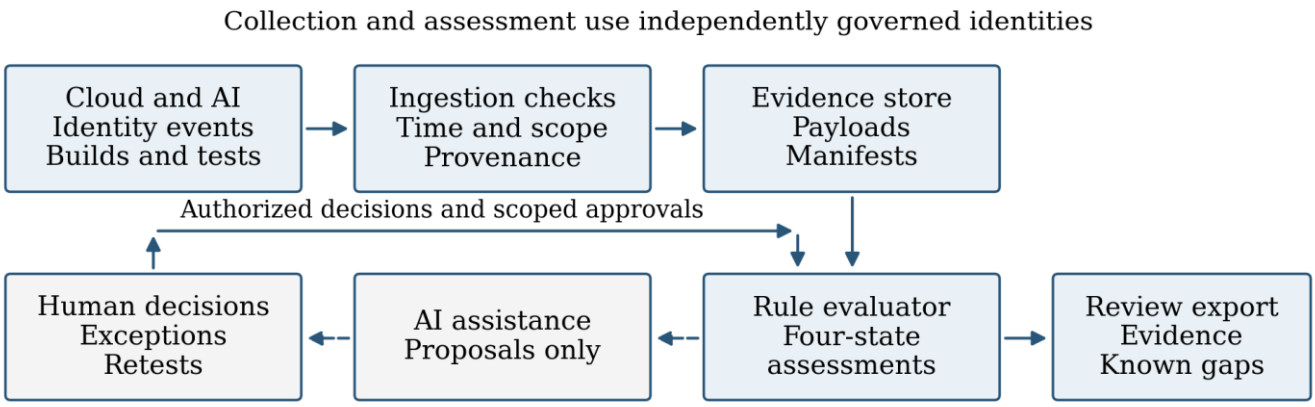
$$\text{Supported fraction } S = \frac{nS}{N} \quad (2)$$

Coverage measures the fraction for which an admissible determination exists, including failures and approved exceptions. The supported fraction measures positive support across the entire defined population. Violation, unknown, and exception fractions are also reported separately. Unequal-duration intervals require duration weighting; otherwise, changing sampling frequency changes the denominator. A state observed at one instant does not prove that it persisted between samples. Retrospective reconstruction from later-arriving evidence is labeled separately from the record of what was known at the original decision time.

5 Reference Architecture and AI Boundaries

Figure 1 separates collection, protected evidence storage, assessment, and disposition. Cloud configuration interfaces, identity events, deployment pipelines, and operational tests emit evidence envelopes. A durable ingestion service validates schemas, removes duplicates using stable source identifiers, and records arrival times. Reconciliation jobs detect gaps that event delivery alone cannot reveal. Invalid records are quarantined with reasons. Both valid and rejected submissions remain traceable under an explicit retention policy, while sensitive payload access is restricted.

Figure 1: Evidence Collection, Assessment, and Decision Boundaries



The evidence store retains payloads and manifests independently of the operational dashboard. Versioned rules produce append-only assessments with links to their inputs. A logically separate findings register tracks owners, remediation, approvals, retests, and closure decisions. The export service packages the applicable population, rule versions, observations, exceptions, and collection gaps for review. OSCAL assessment results can carry observations and findings [4], while local extensions preserve additional provenance fields. Schema validation and reviewer interpretation are required before claiming interchange compatibility; no OSCAL exporter is implemented in the evaluation.

Table 2 illustrates how technical tests can support selected control areas. These are proposed implementation mappings, not prescribed AICPA controls or a complete TSC crosswalk. Each organization must establish which criteria, commitments, system boundaries, and tests apply to its engagement. The design deliberately keeps automated checks narrower than the overall control objective.

Table 2: Illustrative Control Areas and Evidence Tests

Control Area	Proposed Evidence Test	Assessment Boundary
Logical access	Compare observed agent permissions with approved scope.	Approval quality and identity attribution require review.
Change management	Match deployed artifact digest to reviewed build and approval.	A matching digest does not establish software correctness.
System operations	Verify required event coverage and investigated alert records.	Event presence does not prove investigation quality.
Availability, if in scope	Retain scheduled restore-test result and recovery timing.	Backup existence alone does not show recoverability.

In an Azure implementation, candidate inputs include identity audit records, resource inventories, policy results, deployment attestations, and recovery-test artifacts. Event delivery should be paired with scheduled reconciliation and freshness timers. Azure Policy documentation describes a standard reevaluation cycle of approximately 24 hours [10]; a compliance dashboard therefore cannot be assumed to report every change immediately. A custom .NET worker could normalize inputs and evaluate versioned predicates, but this paper does not report an Azure deployment or benchmark that implementation.

AI has two distinct roles: it may operate inside the system being assessed, and it may assist the people reviewing evidence. An assistant can suggest mappings, summarize a finding, or propose remediation with source references. Retrieved evidence is treated as untrusted content, including any embedded instructions. The assistant receives no permission to alter retained records, approve exceptions, suppress findings, or certify a control. A proposed change enters the usual authorization process. If an AI agent performs a production action, its independent authorization and execution records become assessment inputs. Its explanation of why the action was safe is insufficient evidence by itself.

6 Executable Synthetic Evaluation

6.1 Protocol and Reproducibility

The accompanying Python program uses only the standard library. It implements the decision rules for one control and a supplied set of required observations. Authenticity, scope matching, rule identity, and collector health are Boolean inputs, rather than implemented cryptographic or cloud integrations. Twelve fixtures cover ordinary results and evidence faults. A separate enumeration evaluates 576 input combinations: six binary evidence gates, three predicate values, and three exception conditions. Five additional assertions cover a known failure alongside missing support, the freshness boundary, future receipt, expiration, and a duplicate passing record. These checks test the specified logic, not the accuracy of a deployed collector.

For the fixtures, decision time is 1,000 seconds, the ordinary observation time is 950 seconds, receipt time is 951 seconds, and the freshness limit is 60 seconds. Age equal to the limit remains admissible. Exception approval is represented by an expiry value after presuming that scope and authority were already validated; expiry equal to decision time is inactive. This simplification is explicit so that the experiment cannot be mistaken for a complete approval system. Inputs, assertions, fixture results, and aggregate results are included in the supplementary material.

6.2 Decision Behavior and Denominator Effects

All twelve fixtures produced their specified states (Table 3). None of the 576 enumerated combinations produced SUPPORTED when an evidence gate failed or the predicate was indeterminate. The additional assertions also passed. This is conformance to the proposed contract, not a measured false-positive or false-negative rate against independent audit judgments. In particular, the model correctly represents a supplied integrity failure; it does not demonstrate the ability to detect a forged record in practice.

Table 3: Synthetic Fixture Outcomes

Fixture	Injected Condition	Expected and Actual State
F01	Fresh, valid passing evidence	SUPPORTED
F02	Fresh, valid failing evidence	VIOLATED
F03-F08	Stale; integrity failure; wrong scope; old rule; unhealthy collector; absent evidence	UNKNOWN in all six cases
F09	Failure with active exception	EXCEPTION
F10	Failure at exception expiry	VIOLATED
F11-F12	Unresolved conflict; future observation	UNKNOWN in both cases

A separate population contains 100 synthetic resources and ten equal observation slots per resource, giving 1,000 obligations. Twenty resources have no evidence in any slot. Of the remaining 80 resources, each fails in one slot and passes in nine. The resulting counts are 720 supported, 80 violated, and 200 unknown. An observed-only calculation reports 720 of 800, or 90 percent, as passing. Equations (1) and (2) instead give 80 percent evidence coverage and 72 percent supported obligations. The 18-percentage-point difference is a denominator effect, not a reduction in actual security. It exposes how omission can make an incomplete assessment appear stronger.

6.3 Collection Timing Sensitivity

An idealized experiment enumerates 86,400 equally weighted change onsets at half-second positions across one day. A change is assumed to persist until observed; polling and detection are perfect and instantaneous at each scheduled sample. Daily polling yields a mean detection delay of 43,200 seconds and a maximum of 86,399.5 seconds. Hourly polling yields 1,800 seconds and 3,599.5 seconds respectively. A hypothetical event path with fixed five-second delivery yields five seconds for both measures. These outputs reproduce the schedule assumptions and are not measured cloud-service latency or evidence of product superiority.

For persistent changes with uniformly distributed onset within a polling interval, expected delay is half that interval. Short-lived violations can occur entirely between polls and remain undetected, which the experiment excludes. Lost events, throttling, and queue congestion would also change the result. The architectural implication is to declare a latency budget for each evidence source, measure it in deployment, and use reconciliation to detect delivery gaps. High event frequency alone cannot establish continuous control effectiveness.

7 Discussion and Limitations

The main result is a distinction between favorable observations and defensible support across an explicitly defined population. The architecture makes missing evidence actionable without equating every collection fault with a control failure. It also prevents approved risk acceptance from improving the supported fraction. The earlier software-governance framework [8] can provide release records, but it should not override an assessment predicate or erase an unresolved finding.

Several failure modes remain outside the executable model. A compromised source may emit plausible evidence with valid credentials. Two collectors may share the same upstream blind spot, so their agreement is not independent confirmation. Resource discovery may omit an entire account. An incorrect control-to-test mapping may produce consistently wrong conclusions. Protection therefore requires separation of evidence administration from control operation, independent reconciliation where feasible, review of mapping changes, and explicit monitoring of the assurance infrastructure itself. Retention and access policies must balance reproducibility with sensitive-data minimization.

The evaluation has limited external and construct validity. Fixtures were designed from the same rules being evaluated, and the synthetic population intentionally includes missing data. No auditor assessed whether the proposed evidence would be sufficient for an examination. No measurements establish operational cost, scalability, detection accuracy, or improvements in audit effort. Freshness thresholds and observation schedules remain governance choices. A credible next evaluation would use a controlled cloud testbed with injected faults, independently labeled expected outcomes, event loss, clock skew, resource churn, and

manual controls. It should measure detection delay, unsupported passing claims, evidence gaps, processing cost, and reviewer agreement before moving to a consented field study.

8 Conclusion

Continuous internal assurance requires a claim to remain connected to the evidence, population, and rule that justify it. The proposed architecture defines that connection through explicit admissibility gates, four distinguishable outcomes, and time-aware assessment records. Its executable evaluation shows consistent behavior under the stated synthetic conditions and demonstrates how missing evidence changes reported support. The next research step is independent validation of collectors, mappings, and reviewer judgments in realistic environments. Until then, the framework is a reproducible design proposal for strengthening control evidence, with no claim to replace professional judgment or an independent SOC 2 examination.

Research Transparency

Data and code availability: The supplementary package contains the executable synthetic model, fixtures, and generated results. The study contains no human-subject or organizational production data. AI assistance: Generative AI tools assisted literature discovery, drafting, and preparation of the synthetic evaluation. The reported numbers are outputs of the accompanying deterministic program, not observations from an enterprise deployment.

REFERENCES:

1. AICPA, 2017 Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy, with revised points of focus 2022, accessed October 10, 2026. https://www.aicpa-cima.com/resources/download/2017-trust-services-criteria-with-revised-points-of-focus-2022
2. D. Y. Chan, M. A. Vasarhelyi, Innovation and practice of continuous auditing, International Journal of Accounting Information Systems, 2011, 12(2), 152-160. https://doi.org/10.1016/j.accinf.2011.01.001
3. K. Dempsey, N. S. Chawla, A. Johnson, R. Johnston, A. C. Jones, A. Orebaugh, M. Scholl, K. Stine, Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations, NIST SP 800-137, 2011. https://doi.org/10.6028/NIST.SP.800-137
4. NIST, OSCAL Assessment Layer: Assessment Results Model, updated March 3, 2025, accessed October 10, 2026. https://pages.nist.gov/OSCAL/learn/concepts/layer/assessment/assessment-results/
5. R. Cilla Ugarte, M. A. Patricio Guisado, A. Berlanga de Jesús, J. M. Molina López, Making AI Compliance Evidence Machine-Readable, arXiv preprint 2604.13767v1, 2026, submitted for publication. https://doi.org/10.48550/arXiv.2604.13767
6. C. Autio, R. Schwartz, J. Dunietz, S. Jain, M. Stanley, E. Tabassi, P. Hall, K. Roberts, Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, NIST AI 600-1, 2024. https://doi.org/10.6028/NIST.AI.600-1
7. OWASP Foundation, LLM01:2025 Prompt Injection, OWASP Gen AI Security Project, accessed October 10, 2026. https://genai.owasp.org/llmrisk/llm01-prompt-injection/
8. S. Suryawanshi, Spec-driven development for enterprise agentic software engineering: A governance framework for reliable AI-generated software, World Journal of Advanced Research and Reviews, 2026, 31(3), 881-886. https://doi.org/10.30574/wjarr.2026.31.3.2332
9. K. Peffers, T. Tuunanen, M. A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, Journal of Management Information Systems, 2007, 24(3), 45-77. https://doi.org/10.2753/MIS0742-1222240302

10. Microsoft, Get policy compliance data, Azure Policy documentation, accessed October 10, 2026. https://learn.microsoft.com/en-us/azure/governance/policy/how-to/get-compliance-data