SALESFUSION

Evidence-Gated Agentic Workflows

Technical architecture and implementation report

PUBLIC RELEASE · VERSION 1.0.0 · 2026-07-17

Evidence-Gated Agentic Workflows

Publication metadata

Field	Value
Author	SalesFusion Research
Publication date	2026-07-17
Version	1.0.0
Canonical slug	evidence-gated-agentic-workflows
Canonical path	/technical-papers/evidence-gated-agentic-workflows
Publication class	Technical architecture and implementation report
Approval state	Founder-approved public release

Revenue Cortex connection

Revenue Cortex uses evidence-gated workflows to keep buyer research, outbound drafts, campaign setup, approval, execution, replies, and outcomes in distinct states. This matters because revenue work becomes risky when an approved idea, a generated draft, a scheduled campaign, and a verified result collapse into the same record.

Evidence-status box

> SalesFusion uses separation of evidence, approval, execution, and outcome records as an operational governance pattern. This paper defines that pattern and its control logic. It does not claim a measured reduction in incidents, policy violations, or review time. Those claims require the false-authorization and policy-adherence evaluation described below.

> This is a public technical architecture and implementation report. It uses provenance and append-only event-history concepts, risk framing consistent with the NIST AI Risk Management Framework, and LLM-application threat awareness consistent with the OWASP Top 10 for Large Language Model Applications. These references support vocabulary and risk framing only; they do not validate SalesFusion-specific effectiveness.

Abstract

Agentic workflows often collapse several different questions into one operational flag: can the system act? That collapse is unsafe for consequential business workflows. A sourced statement may be well proven but not approved for use. A human may approve one action without authorizing a different version, channel, audience, or time window. A workflow may execute after approval but fail to produce the expected artifact. An outcome may be recorded without proving that the action was authorized. Evidence-gated workflows treat these as separate state machines.

We define five state families: provenance state, claim state, approval state, execution state, and outcome state. Paper 12 specializes the authority and approval portion of this broader evidence-gating model. The paper specifies the records, transitions, invariants, and interfaces needed to keep them separate. The central rule is simple: approval is not execution, and execution is not outcome. Approval grants authority to attempt a specific action under a specific scope. Execution records what the system attempted and what receipts or artifacts it produced. Outcome records what happened after the action, including null, blocked, failed, and adverse states. The design is intended for practitioners building AI-assisted business systems where generated analysis can inform work but must not silently become fact, authority, or completion proof.

The problem

The practical failure mode is not usually that a system has no evidence. The failure is that it has the wrong kind of evidence for the decision being made. A source may prove that a company policy exists, but it does not prove that the current request fits that policy. A reviewer may approve a recommendation, but the approval may refer to a prior draft. A workflow may return a successful process exit code, but the external action may not have occurred or may have occurred twice. A metric may move after the run, but the movement may be unrelated, delayed, duplicated, or outside the approved measurement window.

Many workflow systems hide these distinctions behind coarse states such as ready, approved, done, or sent. Those states are convenient for queues but weak for governance. They encourage last-write-wins memory, approval replay, stale context reuse, and false completion. In an AI-assisted workflow, the risk grows because generated summaries can launder uncertainty. A model can present a sourced fact, a plausible inference, and an unsupported completion statement in the same fluent paragraph. If the runtime treats that paragraph as operational state, the boundary between evidence and authority disappears.

Evidence gating solves a narrower but important problem: it gives each consequential transition a typed reason to be allowed, blocked, escalated, retried, or rejected. It does not make the workflow correct by itself. It makes incorrectness easier to locate and harder to convert into an authorized action without leaving a record.

Contributions

This paper contributes a reusable workflow model with five separated state families, an event envelope for transitions, a policy for binding approvals to exact actions, and a verification protocol designed to keep approval, execution, and outcome from substituting for one another.

The implementation contribution is a SalesFusion pattern: source-backed assertions, approval records, workflow runs, action receipts, and outcome reviews are represented as separate records joined by correlation

and causation identifiers. The design contribution is the set of invariants that must hold across those records. The evaluation contribution is a test plan focused on false authorization, stale evidence, approval replay, duplicate action, and false completion.

Core definitions

Provenance state records where an input came from, who or what supplied it, when it was observed, what restrictions apply, and whether it is current, stale, retracted, or superseded. A provenance state can support a claim but cannot approve an action.

Claim state separates epistemic status from lifecycle state. Epistemic status distinguishes observed_fact, authorized_assertion, inference, hypothesis, estimate, and unknown; lifecycle state distinguishes active, stale, disputed, superseded, and retracted. A claim can be eligible as evidence only when its source, freshness, restrictions, and conflict status satisfy the current decision rule.

Approval state records a human or policy decision about a specific action digest, version, scope, actor class, approver role, conditions, and expiry. Approval authorizes an attempt; it does not create the attempt.

Execution state records a workflow run and any external or internal actions attempted by that run. It includes idempotency keys, connector receipts, artifacts, retries, pauses, failures, and rollback records.

Outcome state records the reviewed consequence or result of the run. It may be positive, negative, null, blocked, indeterminate, delayed, or outside the measurement window.

Evidence-gated architecture

The architecture uses separate records connected by references rather than one mutable status field.

State family	Example states	Primary question	Cannot substitute for
Provenance	registered, reviewed, stale, retracted, restricted	Where did this input come from and can it be used?	Claim truth, approval, execution, outcome
Claim	observed_fact, authorized_assertion, inference, hypothesis, estimate, unknown	What is the statement's epistemic position?	Source existence, lifecycle, approval, action receipt
Claim lifecycle	active, stale, disputed, superseded, retracted	Is the claim currently eligible, contested, or replaced?	Epistemic status, approval, execution
Approval	requested, approved, rejected, expired, revoked, invalid	Who authorized exactly what?	Connector call, delivery receipt, result
Execution	requested, started, output_created, failed, rolled_back, verified_complete	What ran and what artifacts exist?	Human authorization, business outcome
Outcome	pending, no_result, accepted, rejected, adverse, inconclusive	What happened after the run?	Source proof, authority, run completion

The workflow event envelope includes tenant_id, event_id, event_type, occurred_at, recorded_at, actor_ref, source_ref, resource_refs, correlation_id, causation_id, trace_id, schema_version, and classification metadata. The envelope is append-only. Current state is reconstructed from events and record versions, not overwritten in place. Provenance relationships follow the vocabulary orientation of PROV-O.

The minimal data objects are:

```
{
  "claim_record": {
    "claim_id": "clm_123",
    "subject_ref": "entity_456",
    "predicate": "has_current_need",
    "object_value": "source-backed value or structured object",
    "epistemic_status": "authorized_assertion",
    "lifecycle_state": "active",
    "evidence_refs": ["src_version_1"],
    "confidence_label": "bounded",
    "effective_at": "2026-07-16T00:00:00Z",
    "restrictions": ["no_public_quote"]
  },
  "approval_record": {
    "approval_id": "app_123",
    "action_digest": "sha256:...",
    "workflow_version": "wf_v7",
    "scope": {"channel": "internal_review", "max_recipients": 1},
    "decision": "approved",
    "conditions": ["use_approved_claims_only"],
    "expires_at": "2026-07-23T00:00:00Z"
  }
}
```

The action digest is computed from the intended action, workflow version, parameters, target resource, consequence class, and required evidence set. If any of those fields change, a prior approval no longer authorizes execution.

Decision rules and invariants

Evidence-gated workflows require explicit invariants. The most important invariant is that a downstream state cannot backfill an upstream state. A successful run cannot make a stale claim current. A human approval cannot make a missing source exist. An outcome review cannot authorize the action that produced the outcome.

Core invariants:

Invariant	Reason
Every claim used for action must reference at least one allowed source version.	Blocks generated assertion from becoming operational fact by rule.
Restricted or stale provenance cannot satisfy a rule requiring current unrestricted evidence.	Checks for source misuse and silent staleness.
Approval must bind to action digest, workflow version, scope, approver role, conditions, and expiry.	Blocks replay against a changed action by rule.
Execution must check approval at start and before each consequential external action.	Checks for stale or revoked authority during a long run.
Verified completion requires expected artifacts and action receipts.	Rejects process exit or model self-report as sufficient completion evidence.
Outcome review must preserve null, blocked, failed, and adverse results.	Makes erasure of negative evidence detectable.

The execution gate is intentionally repetitive:

```
for each consequential step:
  collect required claim_refs
  verify provenance state for each claim_ref
  verify claim state and conflict status
  compute action_digest
  evaluate policy for consequence_class and scope
  require approval if policy says approval_required
  validate approval action_digest, version, conditions, and expiry
  execute with idempotency key
  store receipt or blocker
  verify expected artifact contract
```

Approval cannot equal execution because it answers a different question. Approval says: a qualified principal accepts this exact attempted action under bounded conditions. Execution says: a runtime attempted the action and recorded what happened. The first can exist without the second, for example when a workflow is paused, expired, blocked by missing evidence, or denied by a connector. The second can be attempted only after the first when policy requires it, but it still may fail, duplicate, roll back, or produce no result. Treating approval as execution creates false completion and hides connector failure. Treating execution as approval creates unauthorized action.

Implementation

SalesFusion uses this pattern through a service boundary with source, assertion, policy, approval, decision, workflow, run, audit, and export resources. The specific product stack is not the important part. The important implementation properties are tenant-scoped records, idempotent mutating requests, optimistic version checks, append-only domain events, and deterministic validators for state transitions.

The workflow adapter receives a decision packet rather than a free-form instruction. The packet contains current state, evidence references, unresolved conflicts, required approvals, action class, reversibility, and

expected artifacts. The adapter cannot mark a run complete by returning prose. It must produce the artifact contract: output object, connector receipt where applicable, trace reference, validation result, and blocker classification if the contract is not met.

Approval and execution are different resources. An approval endpoint creates or decides an approval request. A run endpoint starts, pauses, resumes, retries, verifies, or rolls back a workflow run. A policy evaluation endpoint may return deny, allow, or require_approval; it does not perform the external action. The policy enforcement point evaluates immediately before the action and records the result.

For example, a run may have this legitimate path:

```
source.reviewed
assertion.activated
decision.evidence_added
approval.requested
approval.approved
run.requested
run.started
action.authorized
action.started
action.receipt_recorded
run.output_created
run.verified_complete
decision.outcome_reviewed
```

It may also stop safely:

```
approval.approved
run.started
assertion.marked_stale
policy.evaluated = SOURCE_INSUFFICIENT
run.blocked
approval.expired
```

That path is not an operational embarrassment; it is a correct representation of changed evidence.

Evaluation

The primary evaluation is not a business-outcome study. It is a control-correctness and failure-detection suite. A benchmark corpus should include scenarios with stale sources, contradictory assertions, restricted source text, prompt injection inside source content, approval for an old workflow version, approval outside scope, revoked approval, connector failure, retry duplication, missing artifacts, and outcome records with no causal interpretation.

Suggested measures:

Measure	Definition
Unauthorized-action rate	Consequential actions executed without valid policy and approval state.
False-approval rate	System treats invalid, expired, mismatched, or incomplete approval as valid.
False-completion rate	Run marked verified without expected artifacts and receipts.
Duplicate consequential action rate	Retry or event duplication produces more than one external action receipt.
Evidence exclusion accuracy	Stale, restricted, retracted, or disputed claims are excluded when policy requires exclusion.
Traceability completeness	Reviewer can navigate from outcome to run, approval, claim, and source.

The pre-installation target for unauthorized action and cross-tenant leakage should be zero in the adversarial suite. That target is an acceptance criterion, not evidence that future operations will have zero incidents.

Safety and failure modes

The main safety failures are boundary failures. Source prompt injection can attempt to turn source text into instructions. Memory poisoning can promote a false assertion. Retrieval laundering can remove uncertainty from a generated summary. Approval replay can reuse a prior approval for a changed action. Duplicate action can occur when a retry loses the prior receipt. False completion can occur when a process exits successfully but the expected artifact is missing. Outcome inflation can occur when null or adverse results are omitted from review.

Controls include treating sources as data, preserving claim state, requiring current source eligibility, binding approvals to immutable action digests, using idempotency keys, recording independent action receipts, preserving blocker states, and failing closed when authority is unknown. OWASP LLM risk guidance is useful here because source instructions, tool use, sensitive information disclosure, and excessive agency are practical threats in agentic systems.

Limitations

This architecture does not prove that a workflow is commercially useful. It does not make bad sources good, remove human judgment, or ensure that a reviewer understands a decision. It can preserve uncertainty, but it cannot resolve uncertainty without better evidence or authority. It also adds operational cost: more records, more checks, and more explicit blocker states. Teams that only need low-risk internal drafting may find the full pattern excessive.

The current SalesFusion evidence supports saying that the separation pattern is implemented. It does not support policy-violation, revenue, trust, or decision-quality outcome claims. Those remain evaluation questions.

Practical implications

Practitioners should stop asking whether an agentic workflow is simply ready. They should ask which state is ready. Is the source reviewed? Is the claim active and current? Is the approval valid for this exact version and scope? Has execution produced receipts? Has the outcome been reviewed without erasing null or failed results?

This changes product design. Queue labels should expose distinct state families. Approval screens should show action digests in human-readable form. Run pages should show expected artifacts and verification status. Outcome pages should link back to source, claim, approval, and execution records. Generated summaries should be treated as explanations of state, not state itself.

Conclusion

Evidence-gated workflows make consequential agentic work inspectable by separating provenance, claim, approval, execution, and outcome state. The separation is not bureaucratic decoration. It defines controls intended to block a source from becoming authority, an approval from becoming execution, and an execution attempt from becoming proof of outcome. SalesFusion uses this as a governance pattern and proposes a focused evaluation suite for policy adherence and false authorization. The architecture is most useful where AI-assisted workflows touch external parties, business records, publication, permissions, or other consequential surfaces.

References

- W3C, "PROV-O: The PROV Ontology." https://www.w3.org/TR/prov-o/
- National Institute of Standards and Technology, "AI Risk Management Framework." https://www.nist.gov/itl/ai-risk-management-framework
- OWASP Foundation, "OWASP Top 10 for Large Language Model Applications." https://owasp.org/www-project-top-10-for-large-language-model-applications/

Changelog

- 2026-07-17 — Version 1.0.0: founder-approved first public release; publication metadata, evidence-status box, limitations, and verified references retained.