Nikxius Inc.

NIKXIUS RESEARCH · SEPTEMBER 2026

When AI Gets Write Access

The execution gap between agent decisions and production outcomes

REPORT
NIKXIUS-EXECUTION-2026

DATE
2026-09-15

VERSION
1.0

CLASSIFICATION
PUBLIC

ABSTRACT

A research synthesis and engineering framework for consequential agent actions: authority, exact operations, native state, uncertain outcomes and recovery.

Contents

1. Executive summary
2. From generated answers to production actions
3. How this research was assembled
4. What the enterprise literature agrees on
5. The economics constrain the architecture
6. Defining the execution gap
7. Sixteen problems consequential agents bring into focus
8. The enterprise agent stack is not one straight line
9. What existing infrastructure already solves
10. What composition still has to establish
11. The agent action contract
12. UNKNOWN is a production state
13. Recovery is part of the product, not a retry button
14. Kubernetes makes the problem concrete
15. A crowded market, with different boundaries
16. Observability, authorization, execution and recovery
17. What platform engineering should do next
18. What security engineering should ask
19. What AI platform teams should keep outside the model
20. What not to build by default
21. The thesis Nikxius is testing
22. Open questions and claims this research cannot support
23. The operational standard to aim for
Sources

1. Executive summary

An AI agent proposes a production rollback. The proposal is sensible, the incident is real, and the earlier image is available. None of those facts establishes that the agent may change this Deployment, that the approved image is still the permitted target, or that the cluster remains in the state the proposal assumed. If the update request times out, none establishes whether the change occurred.

This is the execution gap: the distance between deciding on an operation and establishing its authorized effect, outcome and remaining obligations. It appears wherever automation crosses a boundary into a system it does not control completely. Agents make that familiar distributed-systems problem harder to manage by generating more varied proposals, delegating work and selecting tools dynamically.

The research supports taking that gap seriously. It does not establish that every enterprise needs another platform. McKinsey describes shared execution layers that enforce enterprise rules while extending existing standards. BCG describes identity, runtime policy and deployment controls, then explicitly recommends composing an enterprise control plane from appropriate tools. Bain argues that systems of record retain a central role in validating actions. Those are requirements for a working architecture, not endorsements of a new vendor category.[1] [2] [3]

Our review identified **130 relevant first-party publication/report units across nine principal organizations, representing 113 deduplicated report/study families**. These are not 130 independent empirical studies. The collection includes surveys, practitioner reports, case studies and mutable technical documentation. We also examined primary vendor and standards material. Coverage is uneven, particularly for McKinsey, and the synthesis does not establish purchasing demand for a standalone execution product.

Five findings matter for platform teams:

1. **Permission to use a tool is only part of permission to produce an effect.** A consequential operation also needs an exact target, parameters, native preconditions, limits and an enforcement path.
2. **Execution state needs more than success and failure.** A request can be denied, durably admitted, possibly dispatched, confirmed committed, rejected, partially effective or unresolved. A healthy application is a separate conclusion.
3. **Recovery belongs in the contract before execution.** Teams should decide what evidence can resolve ambiguity, who owns the investigation, and which retries are safe before granting write access.
4. **The existing stack is the starting point.** Identity, native RBAC, admission, workflow engines, GitOps, observability and system APIs already solve substantial parts of the problem. Composition may be the best answer.

5. **The economic test is operational.** A control is useful when it makes a real workflow cheaper or more reliable to delegate, with acceptable failure handling. More approval screens or more signed logs alone do not establish that result.

This report introduces an **agent action contract** as a practical design and review framework. It combines established security, identity and distributed-systems concepts; it is not a claim of invention or a new standard. Its central question is simple: for this specific action, what is permitted, what can change, what establishes the outcome, and what remains owed if the outcome cannot be established?

Nikxius is testing execution control and recovery infrastructure around a narrow set of production operations. That product thesis occupies the final portion of the report. The framework and native-stack comparison are intended to be useful whether a team buys software, composes existing tools or decides not to automate an operation.

2. From generated answers to production actions

A generated answer and an external mutation have different failure surfaces. An answer can be reviewed before anyone acts on it. A tool invocation may alter an account, deploy an image, revoke a credential, authorize a purchase or send a message while the surrounding conversation still looks like a draft.

The distinction is not that text is harmless. Bad recommendations, disclosed information and fabricated facts can cause serious damage. The narrower point is architectural: once a system can invoke a writer, the organization must control the effect path independently of the agent's confidence or explanation.

Consider three stages of an incident workflow. An assistant summarizes logs. An agent proposes a remediation. An automation changes the cluster. Each stage can share the same model and interface, yet each needs a different authority boundary. Read access for investigation does not imply permission to mutate. Permission to recommend a rollback does not imply permission to choose any image. Permission to request an update does not establish that the update took effect.

First-party enterprise research increasingly describes this shift in concrete terms. Oliver Wyman discusses agents working across legacy IT, tickets, logs, configuration databases and runbooks. Bain describes contextual, least-privilege tool permissions and identity propagation for nonhuman principals. Google SRE describes agent-assisted operational work while explicitly retaining successful classic automation rather than replacing it simply because AI is available.[4] [5] [6]

These publications do not imply that autonomous production change is mainstream. In Bain's June 2026 publication, drawing on its 951-company Automation and AI Pathfinder survey, 7% of companies reported running fully autonomous agents in production. The same publication reports human-approval and guardrail/exception operating models. Its categories describe respondents' reported autonomy; they are not a census of every agent deployment or an independent safety audit.[7]

The useful distinction is therefore between **capability**, **deployment permission** and **operating maturity**. A model can be capable of selecting an action that an enterprise cannot yet authorize economically. An enterprise can deploy a guarded workflow without delegating full autonomy. A workflow can run in production without having a satisfactory recovery model for every consequential effect.

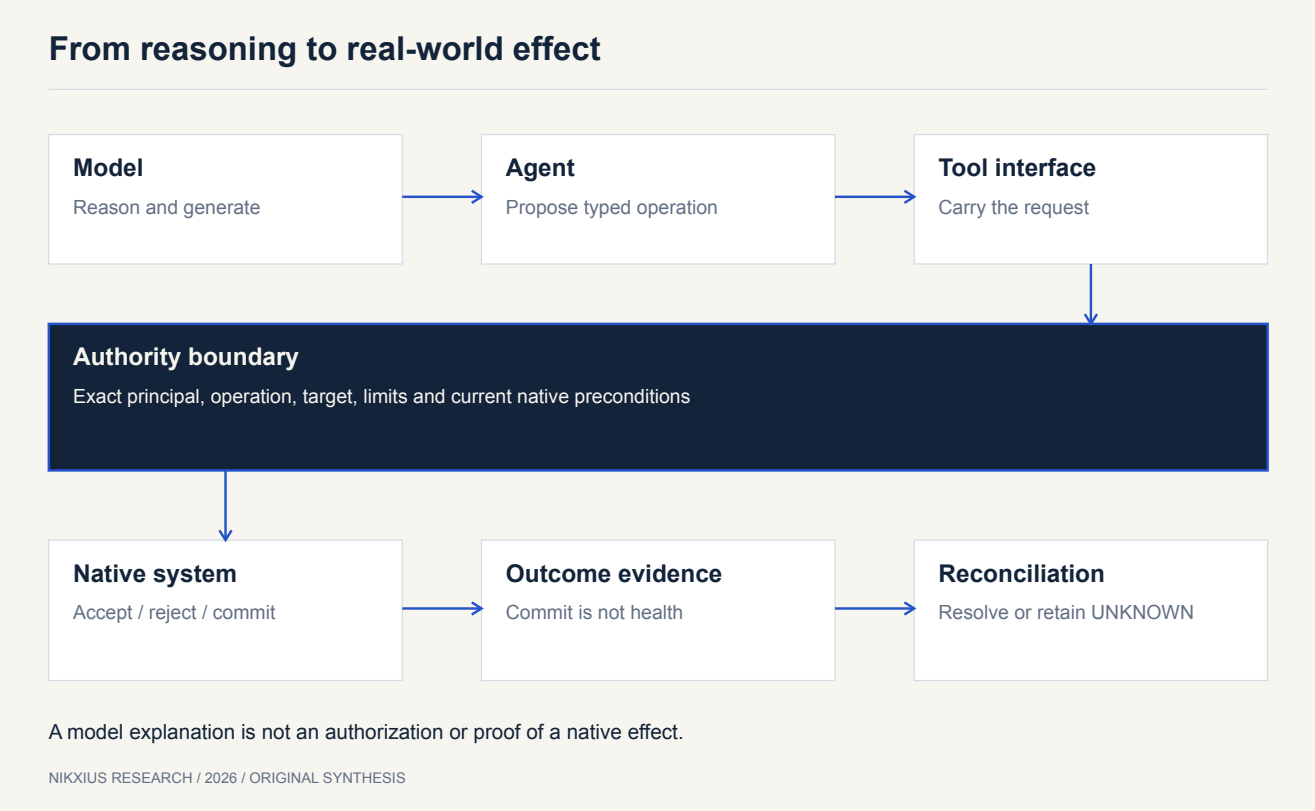

Figure 1. A model helps an agent propose an operation. An authority boundary constrains the native write; effect, observation and reconciliation follow. The authority boundary is independent of the model's explanation.

The practical unit of analysis is the operation. “We have AI agents” says little about who may change what. “This workload may request this exact image change on this Deployment under these conditions” is a statement engineering and security teams can inspect.

3. How this research was assembled

The principal corpus covers McKinsey & Company, BCG, Bain & Company, PwC, EY / EY-Parthenon, KPMG, Accenture, Oliver Wyman and Palantir. Palantir is included as a technical platform publisher, not described as a traditional consulting firm. Its documentation exposes mechanisms and limitations that executive surveys cannot.

The review prioritized 2024 through **15 September 2026**, with particular attention to 2025–2026. Of the 130 counted units, 56 have a 2026 publication date, 45 a 2025 date, 12 a 2024 date and one a 2023 date. Sixteen have no established original publication date. Those include mutable documentation; retrieval dates and SEO updates were not substituted for publication dates.

Discovery combined search-engine queries, official topic and publication pages, related links, PDFs and technical documentation. The inventory preserves discovery methods, original publisher URLs, review scope and retained source captures. It distinguishes publication units from shared study families. Related chapters or summaries do not become independent studies merely because they have different URLs.

The full working inventory contains 218 records: 210 reviewed records and eight unreviewed, inaccessible or excluded candidates. The 210 reviewed records consist of 130 counted principal-corpus units, eight additional principal-organization summaries or chapters excluded from that count, and 72 supplemental sources. The public bibliography exposes source metadata and original links; it does not republish captured reports.

The corpus is a structured review of selected relevant material, not an exhaustive systematic literature review. Some relevant pages were inaccessible at their origins; public text extractions were used where available and recorded as such. McKinsey coverage is materially thinner than several other organizations. Commercial publishers have incentives to promote transformation, platforms or services. Surveys are usually self-reported, populations differ, and selected customer stories are not representative samples.

We separate four kinds of statement throughout the public work:

Statement	What it establishes	What it does not establish
Source fact	What a named publication reports or a documented mechanism specifies	Universal enterprise behavior, independent validation or vendor endorsement
Product fact	Behavior supported by identified implementation and dated local evidence	Customer production acceptance or protection outside the documented boundary
Synthesis	Our interpretation of several sources or mechanisms	Something a source explicitly concluded
Hypothesis	A proposition to test in use, deployment or purchasing	Established market demand or product-market fit

Where a numerical claim lacks a recoverable denominator, we either identify that limitation or omit the number. We do not combine unrelated survey percentages into an industry adoption rate. We do not infer a software market size from broad AI spending intentions. The methodology, source library and consensus matrix provide the fuller audit trail.

Nikxius is not affiliated with or endorsed by the organizations referenced in this research. The report contains original paraphrase, synthesis and engineering examples. Publication dates describe the source, not an assertion that every

underlying study was conducted on that date.

4. What the enterprise literature agrees on

The strongest convergence is broader than execution security. Organizations need useful workflows, accessible data, appropriate operating models, measurable results and clear ownership. A permission boundary cannot repair a workflow that should not exist or a data source whose meaning is unknown.

McKinsey's architecture report places shared semantics, stable interfaces, measurable behavior and enterprise controls together. Bain's platform analysis connects application and orchestration concerns to analytics and knowledge layers. Palantir's Ontology documentation describes an operational layer combining semantic objects with actions, functions and security. Those architectures differ, but each puts business meaning and controlled operations close together.[1] [5] [8]

A second convergence is that controls must operate where work happens. BCG explicitly describes policies blocking noncompliant calls at execution. PwC describes embedding testing, access controls and telemetry in design and deployment. EY discusses decision rights, monitoring and escalation within the infrastructure. KPMG describes agent identity, scoped permissions, runtime isolation and enterprise control systems.[2] [9] [10] [11]

A third is continuity with existing enterprise controls. Accenture argues for contextual, just-in-time access and identity lifecycle management. KPMG's nonhuman-identity analysis emphasizes ownership and scoped access. PwC's agent insider-threat material recommends adapting IAM and incident response. These are reasons to extend and integrate existing controls, not reasons to replace every identity provider or invent a separate incident organization.[12] [13] [14]

The weakest inference would be to turn these points into a unanimous demand for a new trust platform. BCG's control-plane report warns against expecting a single off-the-shelf answer. Bain locates valuable constraints and transitions inside systems of record. Palantir already implements meaningful action controls. Consensus on a function is compatible with disagreement about where it belongs or whether it warrants a separate purchase.[2] [3] [15]

Consulting and technical-publication consensus

	McK	BCG	Bain	PwC	EY	KPMG	Acc.	OW	Pal.
Workflow redesign	S	S	S	S	S	S	S	S	S
Data / context / integration	S	S	S	S	S	S	S	S	S
Runtime policy	S	S	S	S	S	S	S	S	S
Identity / delegation	S	S	S	S	M	S	S	M	M
Authorization before effects	M	S	S	M	M	M	S	S	S
Human oversight	S	S	S	S	S	S	S	S	S
Observability / evaluations	S	S	S	S	S	S	S	S	S
Outcome reconciliation	W	W	W	W	W	W	W	W	M
Security / isolation	S	S	S	S	S	S	S	S	S
Audit evidence	S	S	S	S	S	S	S	S	S
Portable verification	N	N	N	N	N	N	N	N	N
Interoperability	S	S	S	S	S	S	S	S	M

S Strong M Moderate W Weak N No meaningful evidence C Contradictory

Coverage is editorial and sometimes composite. Identity ≠ every delegation control.

See the linked HTML matrix for source-level citations, definitions and limitations.

NIKXIUS RESEARCH / 2026 / ORIGINAL SYNTHESIS

Figure 2. Editorial coverage grades across nine organizations and twelve infrastructure themes. Grades describe the reviewed material, including broad composite themes; they are not survey measurements or support for a standalone Nikxius product.

The accompanying matrix uses Strong, Moderate, Weak, No meaningful evidence and Contradictory evidence. A strong grade means substantive coverage of the defined theme. It does not mean the publication proves all possible subfunctions. In particular, identity coverage does not automatically establish attenuated multi-agent delegation; security coverage does not automatically establish every isolation mechanism; observability coverage does not establish recovery.

Recovery and independent portable verification receive less uniform explicit attention than workflow redesign, data integration and security. That difference is informative, but not proof of a market opening. It may reflect publication emphasis, mature infrastructure being taken for granted, technical details living elsewhere, or an operational problem customers have not yet prioritized.

5. The economics constrain the architecture

More spending does not remove the need to demonstrate value. Bain reports data access and integration as the biggest barrier to AI progress for 41% of respondents in its 951-company survey. A security-only explanation for stalled adoption would miss that result.7

KPMG's Global AI Pulse Q2 2026 surveyed 2,145 senior leaders across 20 countries and territories between 28 April and 25 May. Its cost-response chart reports 24% scaling back and 25% delaying deployments when cost outweighed expected value: **49% combined**, an arithmetic sum of those two categories. A separate chart reports 35% with full cost visibility. These are self-reports from the survey population, not measurements of every enterprise agent or evidence of a dedicated recovery-software budget.16

Human intervention is part of that cost. If a workflow saves drafting time but demands an operator inspect every tool call, its economics may differ sharply from the automation case that justified it. Conversely, removing review from a high-impact action without a defensible contract may move cost into incidents, recovery and organizational refusal to delegate.

Oliver Wyman's July 2026 research, published in September, used separate samples of 130 technology leaders and 70 CEOs or executive-committee members. Among the technology respondents, 70% reported limiting agents to human approval at each step. That is evidence about reported operating constraints. It is not proof that those approvals are unnecessary, nor evidence of disagreement between a CEO and CIO in the same company.17

The design objective is **bounded autonomy with intelligible exceptions.** Routine actions that satisfy a narrow contract may need no additional human click. A changed target, stale native state, revoked grant or unresolved prior effect may require refusal or escalation. The policy should explain why intervention is needed instead of treating human presence as a universal safety property.

Oliver Wyman's work on government operations makes a related point: excessive approvals, duplicative reporting and unclear decision rights can make automation amplify existing institutional friction. Simplification may create more value than adding an agent to every step.18

A useful business case therefore asks four questions together. How much useful work can be delegated? How much engineering and review effort does the boundary require? How often do exceptions occur? How expensive are unresolved outcomes? The answer can favor a narrow runtime, a native workflow, a simpler deterministic job or no automation at all.

6. Defining the execution gap

The execution gap is not a claim that enterprises lack authorization. It is a failure of composition when authority, action definition, native state, dispatch, observation and recovery do not refer to the same operation.

A policy engine can approve a request against the context it receives. A workflow engine can retain durable history. A service identity can authenticate a caller. Kubernetes can enforce native authorization and reject a stale update. An audit pipeline can retain events. If those components disagree about the operation's identity, permitted target, current state or completion criteria, each may work as designed while the overall workflow remains unsafe or expensive to operate.

For example, imagine an approval for "roll back the checkout service." The automation later resolves that phrase to a mutable deployment name and image tag. Another controller changes the Deployment between review and execution. The update request times out. A retry is issued with a new request key. A dashboard reports the second invocation as successful. The organization has approvals, logs and a green tool result, yet still lacks a defensible account of the intended mutation and its first effect.

Each element is repairable using established techniques. Resolve resource identity. Bind exact parameters. Check current native state. Preserve durable request identity. Use native concurrency and idempotency semantics. Distinguish acknowledgment from effect. Retain uncertainty. Reconcile against appropriate evidence. The difficulty is making those repairs agree across the operation lifecycle.

This framing has a deliberate boundary. It concerns operations that change consequential external state. It does not replace model evaluation, sensitive-data controls, prompt-injection defenses, network isolation or task-level supervision. A correctly authorized action can still be a bad business decision. A perfectly preserved record can still describe harmful behavior permitted by a bad policy.

The test is specific: **can the organization state and enforce what this operation may do, then account for what happened without inventing certainty?** If its existing stack answers that question well, the execution gap may already be closed for that workflow.

7. Sixteen problems consequential agents bring into focus

These problems are not all new, and they do not all require new software. Autonomy changes their frequency, combinations and ownership. The distinctions below are a review instrument, not a claim that a single platform should own every control.

Authority

A principal may be authenticated without being authorized for the requested effect. Authorization should identify the operation, resource and limits rather than merely the ability to call a generic tool. OWASP's excessive-agency guidance recommends granular tools, least privilege and downstream authorization instead of relying on an LLM to decide what is allowed.[19]

The practical question is whether "can use Kubernetes" has accidentally become "can issue any mutation the tool can express." Narrow operation interfaces reduce that expressive authority. They do not eliminate the need for a trustworthy caller and a correctly configured native credential.

Identity and attribution

An execution path can involve a human requester, service account, agent instance, delegated agent and credential-holding service. Those identities should not collapse into one unqualified "agent" label. Accenture and KPMG both emphasize identity lifecycle and scoped access; Microsoft documents distinct application and delegated authorization for agent identities.[12] [13] [20]

A useful record states which identity was verified, which identity was asserted and which principal actually presented the native credential. Model names and trace IDs can help investigation, but neither is a substitute for an authenticated principal.

Delegation

Delegation can broaden authority accidentally when an agent invokes another agent that holds stronger permissions. The receiving service needs the original authority context and a reason to accept the delegation. BCG discusses on-behalf-of identities across agent interactions; Bain emphasizes nonhuman identity propagation.[2] [5]

Propagation alone is not attenuation. Carrying the original caller's name does not prove that a child agent stayed within the parent's resource, time or quantity limits. Every relevant boundary must enforce the scope it accepts, and budgets must not silently reset with a new subtask or session.

Typed intent

"Fix the incident" is a goal, not an executable authorization. A consequential operation needs an externally meaningful representation: operation type, target, parameters, preconditions and expected effect. This is our engineering synthesis of the control requirements, not a claim that recording a payload proves what a model mentally intended.

Typed intent should be resolved before approval. Otherwise, a human may approve an abstract phrase while later code chooses a materially different target. Mutable tags, aliases and names deserve particular attention when the native system offers stronger identities.

Runtime policy

A policy decision is only effective if the effect path honors it. Existing products already do real enforcement. AWS AgentCore evaluates tool interactions at a gateway boundary, and Palantir submission criteria can prevent action side effects.[21] [22] [15]

The design question is the scope of that enforcement: which paths are mediated, what inputs are trusted, how state enters evaluation, and what can bypass it? A policy file in a repository is not evidence that every production mutation passed through it.

Human review economics

Review should bind the operation that will execute and remain meaningful at its volume. Asking someone to approve hundreds of nearly identical prompts can transfer responsibility without supplying understanding. The evidence on limited autonomy and operational friction supports examining review cost, not abolishing review.[7] [17] [18]

A review interface should expose consequence-bearing differences: resource, exact change, scope, current state and expiry. If any of those change materially, the system should apply the documented reauthorization rule. Whether a human is required is a policy choice tied to the operation and context.

Uncertain external effects

A lost response can follow a committed change. Stripe documents indeterminate outcomes and provider-specific retry behavior; Palantir documents a writeback case where an external request succeeds while internal Ontology changes fail.[23] [24]

The general lesson is to preserve uncertainty, not to ban every retry. A provider can offer safe same-key retries under documented conditions. A different writer may offer no equivalent guarantee. The contract must state which behavior is available and what the caller must preserve to use it.

Concurrency and revocation

Authority and resource state can change between proposal and dispatch. A grant checked at the beginning of a workflow may be revoked before the write. A Deployment may be replaced or edited after approval. Revalidation and native conditional updates address different parts of that window.

There is also a hard boundary: revocation cannot necessarily recall an external request that has already crossed the writer. A worker lease coordinates local ownership; it does not pull packets back from a remote API. These are distributed-systems constraints that the action contract must acknowledge rather than hide behind a “kill switch.”

Evidence

A useful execution record relates authority, exact operation, policy, dispatch and native observations. It also identifies missing evidence. A signed file can protect the integrity and attribution of those records within its trust model; it cannot make the records complete or true by itself.

The distinction follows established evidence practice. PCAOB's audit-evidence standard treats relevance and reliability as properties of evidence quality, and says more low-quality evidence does not compensate for poor quality. That is not an agent-specific certification requirement; it is a useful warning against equating volume or cryptography with assurance.25

Independent review

A reviewer should be able to distinguish what the issuing system observed from what the target system established. Portable records help only if their semantics, trust anchors, omissions and retention boundaries are understandable. Transparency standards such as RFC 9943 provide useful signed-statement and receipt mechanisms, but do not establish the business correctness of every registered statement.26

Independence also has levels. Checking a signature without connecting to a vendor is different from independently validating the original external event. A product should say which kind of verification it supports.

Agent-to-agent semantics

A protocol can carry a request without giving both sides the same definition of success. One agent may interpret "rollback completed" as a request accepted by a deployment service; another may interpret it as a healthy application. Bain explicitly distinguishes common call syntax from business vocabulary, policy and approval semantics.3

An action contract should travel across handoffs with stable operation identity and outcome definitions. Otherwise, each hop can make a locally reasonable interpretation that breaks the end-to-end promise.

Cost and resource amplification

Agents can branch, retry and delegate. Limits on one API call or one session may not bound the entire task. EY discusses loop-related cost exposure; KPMG's survey documents cost-driven deployment changes.10 16

The budget boundary must match the unit of accountability. AWS temporal-policy documentation, for example, explicitly scopes history-based counts to a session and describes what happens when a new session starts. That is a documented scope to compose with other controls, not evidence that AWS lacks stateful policy.27

Stale or corrupted context

An agent can reason coherently from outdated information. The correct response is not to treat its context as the authoritative state of the target. Native reads and preconditions should establish what matters for the mutation at execution time.

That protects only the checked properties. An exact Deployment version does not establish that a database migration is compatible with an older image. Teams must decide which preconditions the runtime can verify, which require another system and which remain a human or application-level responsibility.

Blast radius

A tool can be narrow in name but broad in effect. “Run command,” “apply manifest” or “call arbitrary API” can give a planner much more authority than the interface suggests. OWASP’s guidance on excessive functionality and permissions makes this a concrete security design concern.19

Bound the actual writer, not only the visible tool description. Consider credentials in Secrets, token creation, shell access, CI jobs and other controllers. If the agent retains an alternate writer, a mediation service may be optional in precisely the scenario where it is needed.

Machine economic activity

Purchasing and payment agents need precise mandates, identity, transaction binding and outcome semantics. Accenture discusses enforceable intent chains while also locating important advantages in banks, payment networks and wallets. AP2 describes mandate delegation and transaction authorization.28 29

Those mechanisms should not be confused with settlement or fulfillment. An authorized payment attempt is not the same event as funds reaching the recipient. A purchase instruction is not evidence that the goods arrived. Financial operation families need their own contracts and established domain controls.

Outcome ownership

When an operation remains unresolved, someone must own it. A trace can show that calls occurred while leaving no team responsible for deciding what happens next. PwC’s incident-response guidance includes recovery verification; Oliver Wyman’s operating-model research describes incomplete agent ownership and control artifacts.14 17

Our synthesis is that the unresolved obligation deserves explicit status, ownership and escalation. It may be more operationally useful than another undifferentiated event stream. That proposition still needs testing against real exception volume and operator workflow.

8. The enterprise agent stack is not one straight line

Model, orchestration, identity and evidence diagrams often appear as a vertical stack. Real execution is less tidy. Identity and policy surround several boundaries. Native systems retain authority over their own state. Observability spans the lifecycle. Recovery can return to an earlier decision while preserving the record of an already dispatched action.

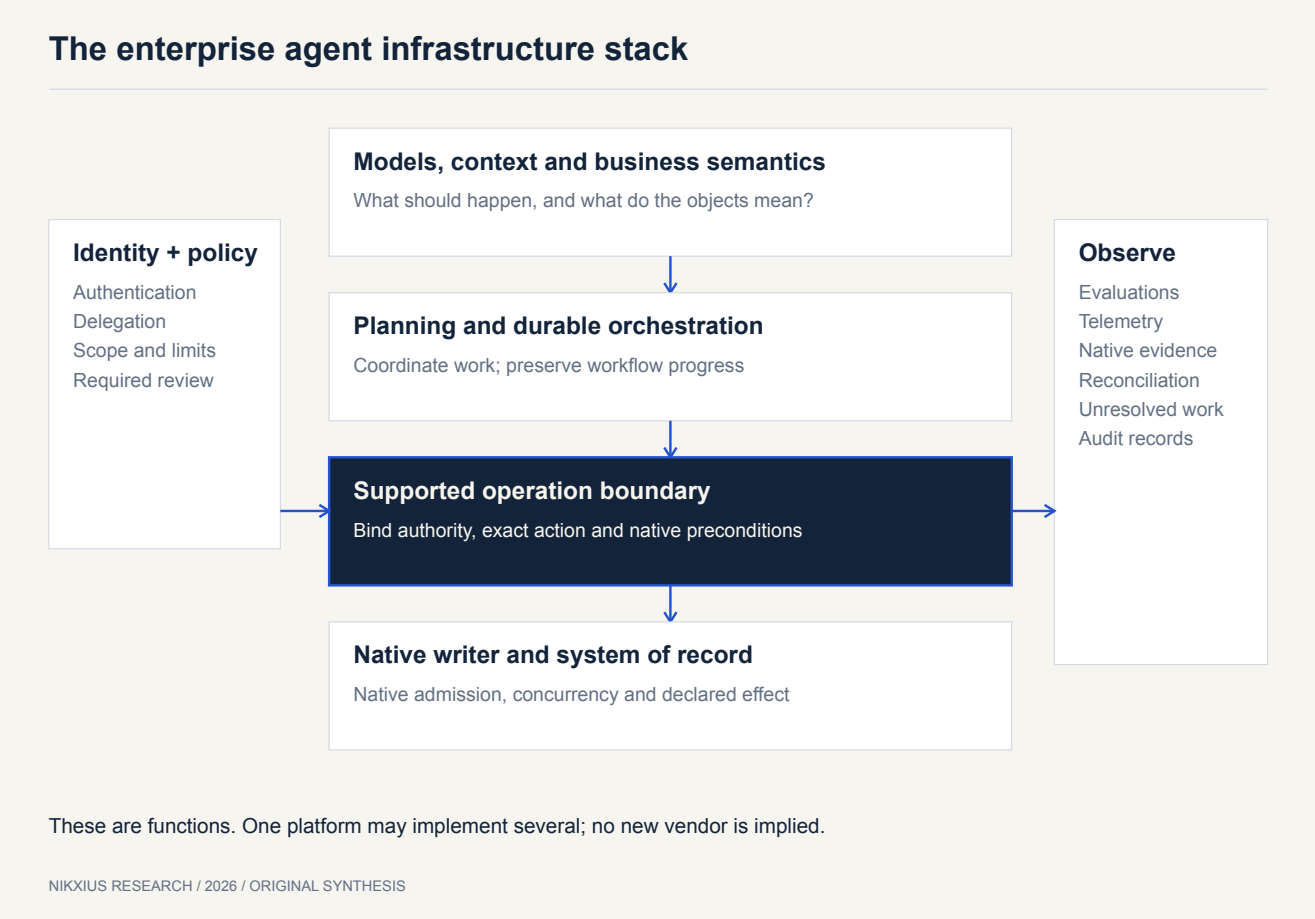

Figure 3. An enterprise agent architecture with planning and orchestration above an operation boundary, a native writer below it, identity and policy alongside it, and observation, evidence and recovery spanning the lifecycle.

The following division is useful because each layer answers a different question:

Layer	Principal question	Existing examples or mechanisms
Models and evaluation	Is the generated decision useful, robust and appropriate?	Model services, workflow test suites, Palantir AIP Evals
Context and business semantics	What do the objects and actions mean?	Data platforms, application schemas, Palantir Ontology
Planning and orchestration	What work happens next, and how does the process resume?	Agent frameworks, Temporal, existing automation

Layer	Principal question	Existing examples or mechanisms
Identity and delegation	Which principal is acting and on whose authority?	Enterprise IdPs, Microsoft Entra Agent ID, workload identity
Policy and authorization	Is this requested interaction allowed in this context?	OPA, AWS AgentCore Policy, native authorization
Operation boundary	What exact native effect is admitted and dispatched?	Typed adapters, application actions, controlled native writers
Native state and concurrency	What can the target system accept or establish?	Kubernetes API semantics, native transactions and provider idempotency
Observation and recovery	What happened, and what remains unresolved?	Native histories, workflow state, reconciliation procedures
Evidence and assurance	What records can a reviewer trust for which conclusion?	Application audit records, signed statements, retained observations

These examples demonstrate overlapping capabilities, not equivalent products. OPA deliberately separates policy decisions from enforcement in applications. Temporal's durable workflow model still requires attention to Activity idempotency for external effects. Palantir integrates semantics and actions; its documentation also makes cross-system limitations explicit.[30] [31] [8] [24]

A team should map its actual path, including credentials and side effects. A generic agent-stack diagram cannot establish whether the approval system, runtime service and native API agree on the same operation.

9. What existing infrastructure already solves

The most credible starting position is to give existing controls their full due.

Identity and privileged access already provide strong foundations for authenticating principals, constraining permissions and managing credential lifecycles. Agent-specific identity products extend that work. Microsoft Entra Agent ID documents authorization rules and restrictions on high-privilege permissions. New agent identity should integrate with the organization's identity model rather than become an ungoverned second directory.[20]

Runtime policy already exists in products and open-source components. AWS AgentCore Policy evaluates interactions at the gateway boundary; its temporal policies can express history-dependent constraints. OPA provides policy decisions over structured input and data. These systems are not accurately described as mere dashboards or static allowlists.[21] [27] [30]

Native application controls can bind business meaning and permissions close to the writer. Palantir action submission criteria gate side effects. Modern Treasury documents conditional approval rules, group review and quorum. Its notification sequence should not be confused with a universal guarantee that reviewers cannot act out of order; the exact product semantics matter.[22] [15] [32]

Delivery and workflow systems can provide approval gates and durable progress. GitHub deployment environments supply protection rules and environment controls. Temporal documents Activity retries and recommends idempotent Activities. Their role is not erased by an agent proposing the work.[33] [34] [31]

Observability and evaluations remain essential. Teams need to inspect bad decisions, test workflow versions, detect drift and understand cost. Palantir AIP Evals explicitly addresses nondeterministic systems through test suites, evaluation functions and comparisons. A pre-effect boundary does not replace those functions.[35]

The resulting native composition can be entirely adequate. For a predictable deployment workflow, a reviewed change in Git, a protected delivery environment, native admission and existing operational runbooks may supply everything required. Adding an agent or an execution service simply to modernize the diagram can increase complexity without improving the outcome.

10. What composition still has to establish

Saying that tools exist is not the same as showing that their composition satisfies one operation contract. The integration must establish several cross-component invariants.

First, the approval and actual mutation must share exact meaning. If an approval references a human-friendly name while dispatch uses a late-resolved resource, the binding can drift. The native target's identity and relevant version should be part of the operation definition where the system supports them.

Second, durable workflow completion must not be mistaken for native effect completion. A task can complete when an API accepts a request, while a controller continues changing the system. For some APIs, successful queue admission is the strongest immediate result available. The contract should name that result honestly.

Third, history must be sufficient for the question being asked. A current value equal to the requested value may show present convergence. It may not show that this operation caused the value, particularly after intervening writes. Event correlation, continuity, native identity and retention can determine whether an original effect is attributable.

Fourth, retry policy must match the writer. Stripe's documented same-key network retry guidance and Temporal's idempotent-Activity guidance illustrate why a generic instruction to retry is incomplete.[23] [31]

Fifth, alternate paths must be controlled. An agent that can write through another token, tool, CI job or privileged helper can bypass a well-designed mediation endpoint. This is a deployment property, not something an SDK wrapper can establish by itself.

Finally, unresolved work needs an owner and a safe terminal behavior. Some operations cannot be conclusively resolved with the available history. A product that always turns uncertainty into success or failure may create an attractive dashboard by discarding the most important information.

None of these points establishes that a startup must own the integration. They define the work that a native composition, internal platform or commercial service should demonstrate.

11. The agent action contract

An agent action contract is a documented agreement between a proposer, an enforcing runtime and a native writer about one class of consequential operation. It connects permission with effect and outcome. It borrows from access control, conditional updates, idempotency, durable workflows, state machines and evidence practice.

It is intentionally more specific than “the agent may use this tool.” It is also narrower than a universal agent governance standard. A Kubernetes image change, payment authorization and customer-account closure have different native semantics and should not be forced into identical completion rules.

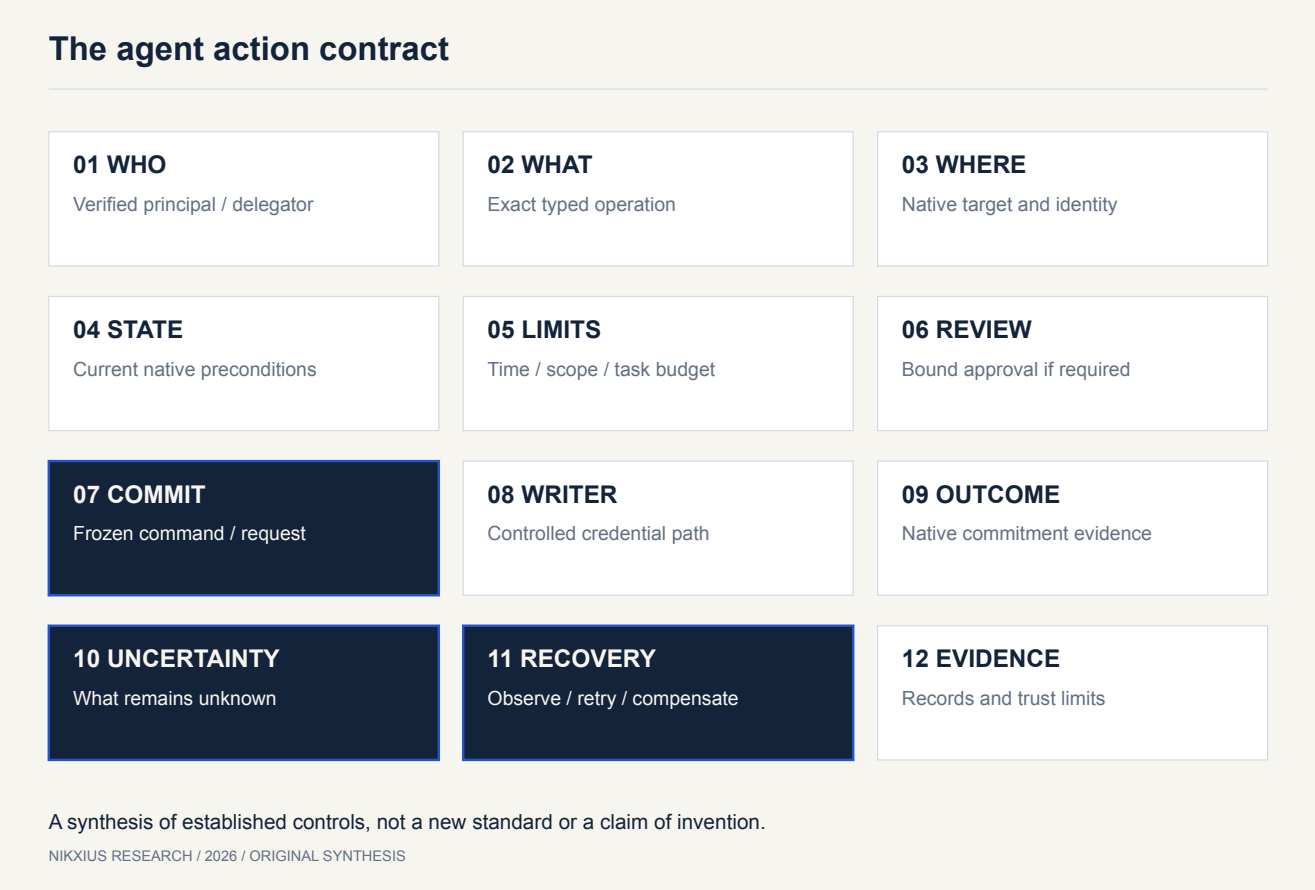

Figure 4. The agent action contract connects verified authority, exact operation and target, current native preconditions, limits and review, durable dispatch, native outcome evidence, uncertainty, recovery and retained records.

Contract field	Question to answer before dispatch
Who	Which principal is verified, and which delegator is accountable?
What	What exact typed operation and parameter set are permitted?
Where	Which tenant, environment and native resource identity may change?
Under which state	Which native preconditions must still hold?
With what limits	What expiry, quantity, cost, concurrency and task budget apply?
Review	Which additional authorization, if any, must bind this operation?
Commit	Which exact command is frozen before possible dispatch?

Contract field	Question to answer before dispatch
Writer	Which credential and authoritative path can produce the effect?
Outcome	What native evidence establishes acceptance, commitment and convergence?
Uncertainty	Which failures leave the effect unknown, and what must remain reserved?
Recovery	Which observations, retries or new compensating operations are permitted?
Evidence	What records, trust assumptions and omissions remain inspectable?

For a rollback, “what” should be an explicit image change rather than permission to run arbitrary shell commands. “Where” should distinguish a resource’s current name from its identity. “Under which state” should capture the baseline being reversed. “Recovery” should describe the lost-response case as carefully as the successful response.

The contract must also state what it cannot establish. It may control a Deployment image mutation without proving application health, schema compatibility or absence of alternate write credentials. Those are consequential boundaries, not footnotes to hide after a broad safety claim.

Not every field must be implemented by a new service. An IdP can authenticate the principal, native policy can enforce organizational constraints, a workflow system can retain process history, and the target API can provide concurrency controls. The point is to make the combined promise explicit and testable.

The technical note translates this framework into operation identity, native preconditions, commit classification and failure tests. It separates illustrative contract notation from the current Nikxius API so that readers do not mistake a public framework for an implemented interoperable specification.

12. UNKNOWN is a production state

A timeout is an observation about communication. It is not automatically a fact about the remote effect.

Suppose a worker records an attempt, sends a conditional mutation and loses the connection. The native system may reject the request, commit it or still be processing it. The caller cannot choose a conclusion merely because its own timeout expired. A retry under a fresh identity can create another logical operation before the first one has been accounted for.

Stripe's documentation is unusually useful here because it states both sides: network failures can be retried using the same idempotency key and parameters under its contract, while certain server-error outcomes remain indeterminate and may have produced side effects. Palantir's writeback documentation similarly acknowledges that external success and internal failure can coexist.[23] [24]

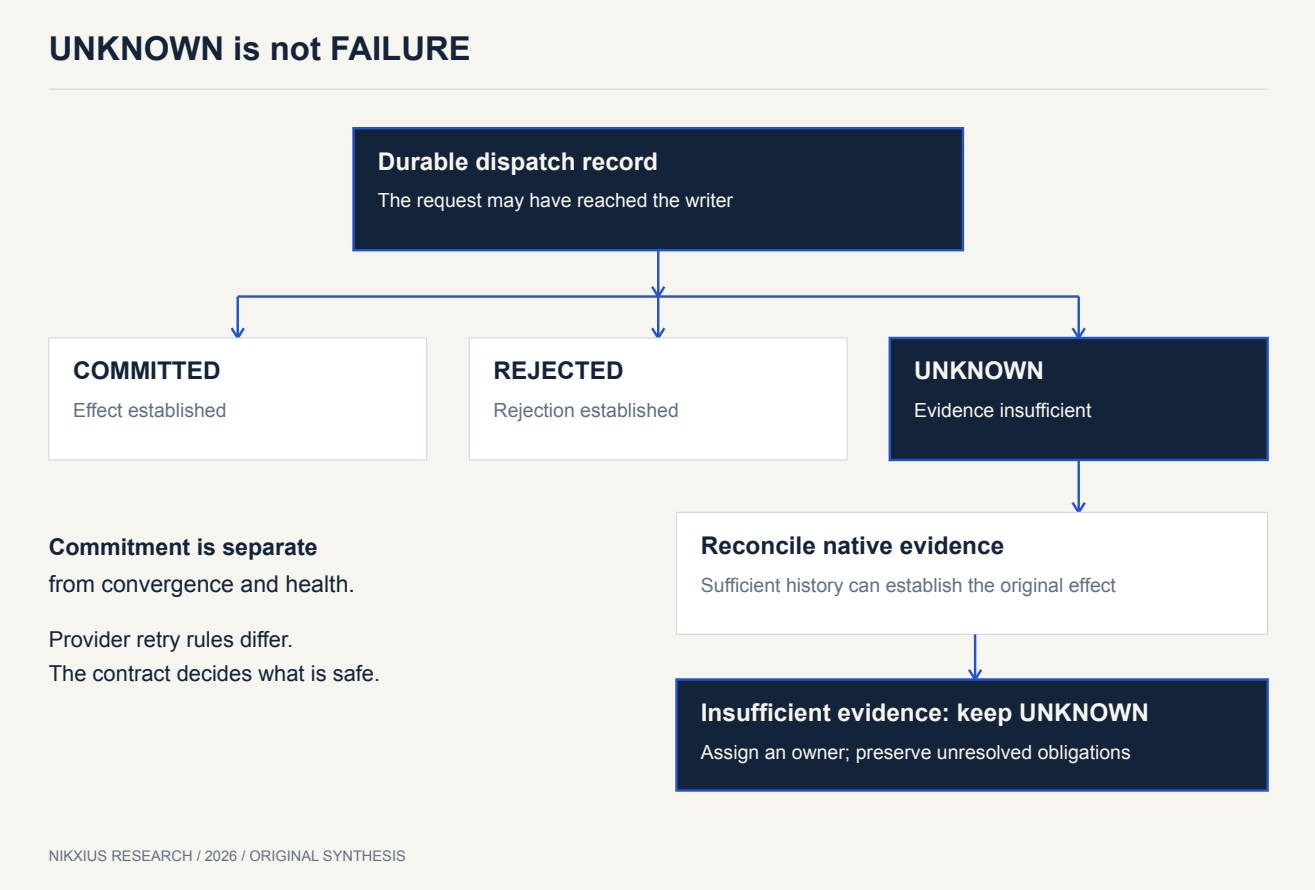

Figure 5. After a recorded dispatch, authoritative evidence may establish commitment or rejection, or leave the outcome unknown. Reconciliation can resolve an unknown only with sufficient evidence; insufficient history leaves it unknown. Commitment is separate from healthy rollout.

Three distinctions prevent much confusion:

- **Admitted is not dispatched.** A local decision or durable record does not prove a request reached the writer.
- **Committed is not converged.** A desired-state update can be accepted while controllers or applications remain unhealthy.
- **Currently matching is not necessarily attributable.** The requested value may be present for a reason unrelated to the operation being investigated.

"Unknown" should have consequences. It may retain a concurrency reservation, prevent a task budget from being reused, assign investigation to an operator, or block compensation until the first effect is understood. The correct consequence depends on the native contract and the cost of duplicate or conflicting effects.

A system should also distinguish unknown from known partial effect. If evidence establishes that one part committed and another failed, that is more information than “unknown.” Preserving that detail allows a recovery procedure to address what actually happened rather than repeat an entire workflow.

Observability remains valuable throughout this process. It helps gather and correlate evidence. The distinction is that an operational state machine must decide which conclusions that evidence supports and retain a pending obligation when it supports none.

13. Recovery is part of the product, not a retry button

Recovery has at least three different meanings. A workflow process resumes after its worker crashes. An operation’s external effect is reconciled after a response is lost. A new operation compensates for a known undesirable effect. These are related, but they are not interchangeable.

Temporal provides a mature example of durable process execution. Its documentation also recommends idempotent Activities, because a durable orchestration history does not itself make an external effect safely repeatable. A system using Temporal should respect that contract rather than claim that an orchestration engine supplies universal exactly-once external execution.[34] [31]

Reconciliation asks what can be established about the original operation. It may use a provider’s idempotency record, a native object version, a continuous event history or another authoritative acknowledgment. Its output can be confirmed commitment, known rejection, identified partial effect or continued uncertainty. It should not manufacture a fresh logical operation merely to make the queue appear empty.

Compensation is different. Rolling back an image changes the system again. Refunding a payment is another financial event. Restoring a previous configuration may fail or create new consequences. Each compensation needs its own authorization, preconditions and outcome record. The original operation should remain in history.

Even the word “reversible” deserves scrutiny. EY’s trust-layer discussion uses signed, traced and reversible actions as an architectural aspiration. A system designer must translate that aspiration into specific native behavior. An email cannot be made unread by a database rollback. A trade cannot be assumed cancellable after execution. An earlier container image does not undo a schema migration.[10]

Our synthesis is that the unresolved obligation can be more valuable than the trace. A trace answers what the software recorded doing. An obligation record answers what the organization still needs to determine, who owns that work, and which actions remain unsafe while it is unresolved. Whether that deserves a separate product depends on how much integration and operational work it removes.

14. Kubernetes makes the problem concrete

Kubernetes is a useful example because its native controls are substantial and its state transitions are explicit. It is not an argument that every cluster needs another control layer. The API, RBAC, admission mechanisms, controllers and delivery workflows should be the baseline against which an additional service is evaluated.[36] [37] [38]

Imagine an SRE agent investigating a degraded Deployment after an image update. It proposes changing one container to an explicitly permitted prior immutable image. A broad architecture gives the agent a write-capable Kubernetes credential and asks it to behave. A bounded architecture lets the agent propose a typed operation while a controlled writer checks the exact action and relevant native state.

The narrower architecture still needs answers to hard questions. Which cluster and namespace? Which Deployment UID, not just name? Which container? Which complete image digest? Which observed resource version? Which earlier operation is being compensated? Is this caller's authority still valid? What should happen if someone else edits the Deployment during review?

Kubernetes supports native concurrency semantics, including updates conditional on a resource version to detect lost updates. Its current API documentation also specifies conditions for comparing resource versions on supported server versions; clients must respect those type and version boundaries. Nikxius uses the captured version as an equality precondition rather than inferring an operation's success from version ordering. Native authorization and admission remain in force when an intermediate service dispatches the request.[36] [39]

There are several separate outcomes to observe. The API may accept a desired image change. The Deployment controller may begin a rollout. Pods may become ready. The application may or may not work correctly. Kubernetes Deployment documentation explains rollout progress, conditions and rollback behavior; none turns an accepted specification update into universal proof of application recovery.[40]

For a precise rollback contract, the previous image is not merely "something in rollout history." The operation should identify the permitted baseline and what it reverses. In Nikxius's current implementation, the explicit rollback profile references a previously committed operation in the same tenant, checks the same target and UID, and requires reversed before/after image relationships. The rollback is a new authorized operation with its own outcome. This is a product fact about the implemented profile, not a claim that Kubernetes requires that design.[39]

The failure cases teach more than the happy path:

Case	Required distinction
Deployment was replaced under the same name	The resource identity changed; an earlier authorization must not silently transfer
Another actor edited the Deployment	Revalidate relevant state and use native concurrency semantics
Grant was revoked before dispatch	Refuse under the documented revalidation policy
Reply was lost after the API may have committed	Preserve possible dispatch and an unknown commit outcome
Same proposal arrives again	Preserve its logical identity; reject changed parameters under the same key
Current image matches after a history gap	Present convergence may be known while original causation remains unknown
Image update committed but rollout is unhealthy	Record commitment separately from degradation; do not label the mutation unexecuted
A compensating rollback is proposed	Apply a new contract rather than erasing the earlier operation

The first public Nikxius replay demonstrates a local exact-image change and response-loss recovery, plus separate boundary checks. It is not a recording of a customer rollback deployment. The distinction matters when assessing what the implementation has actually demonstrated.41

Teams already using GitOps must also decide who owns desired state. An out-of-band image mutation can be overwritten by reconciliation from Git. The correct solution may be to change a reviewed desired-state source rather than patch the cluster directly. A runtime should not manufacture a second writer where the customer's operating model intentionally has one.

15. A crowded market, with different boundaries

The emerging market is not empty space between models and systems of record. Identity vendors, cloud platforms, security products, workflow engines and application platforms already meet there.

The first distinction is **identity versus effect authority**. An identity product can establish and govern principals without owning every native mutation. Conversely, a narrow execution runtime should not need to become the enterprise identity provider. Microsoft Entra Agent ID and existing workload-identity practices make integration a more credible starting point than replacement.20

The second is **policy decision versus enforcement path.** OPA intentionally supplies decisions to callers. AWS AgentCore combines a gateway boundary with policy enforcement. Microsoft's Agent Control Specification, inspected during the search review, describes a draft control contract whose host is responsible for enforcement. These are different boundaries, not evidence that one side lacks "real governance."[30] [21] [42]

The third is **agent security versus operation completion.** Security vendors increasingly describe enforcement inside the execution loop. Airia uses the phrase "AI agent execution control"; Snyk describes governance inside that loop. OWASP's agent-security guidance addresses exact-action approval, replay protection and other controls. Nikxius cannot credibly claim to have discovered pre-action enforcement or exact-action authorization.[43] [44] [45]

The fourth is **durable process execution versus native effect semantics.** Temporal is a strong example of durable orchestration with explicit Activity requirements. A recovery-capable action implementation must work with those requirements rather than treating orchestration as merely logging.[31]

The fifth is **generic action infrastructure versus a deeply specified operation.** Arcade describes an actions runtime between agents and business systems. Palantir integrates semantics, permissions and action types. A new vendor should be compared with those real capabilities, not an imagined market consisting only of model-output filters.[46] [8] [15]

Even outcome-aware recovery language is already present in the market. Cohesivity's July 2026 technical article describes write timeouts as potentially unknown, stable operation identifiers, durable records and reconciliation before retry. That supports the relevance of the problem while removing any basis for claiming that the vocabulary or basic mechanisms are unoccupied.[47]

The plausible differentiation is therefore operational and must be demonstrated: fewer bespoke integrations for a specific action, stronger agreement between authority and native semantics, less manual exception work, or clearer unresolved-state ownership. None follows from naming a category. Native composition remains the hardest and often best alternative.

16. Observability, authorization, execution and recovery

These functions are complementary. Positioning them as mutually exclusive products creates false comparisons.

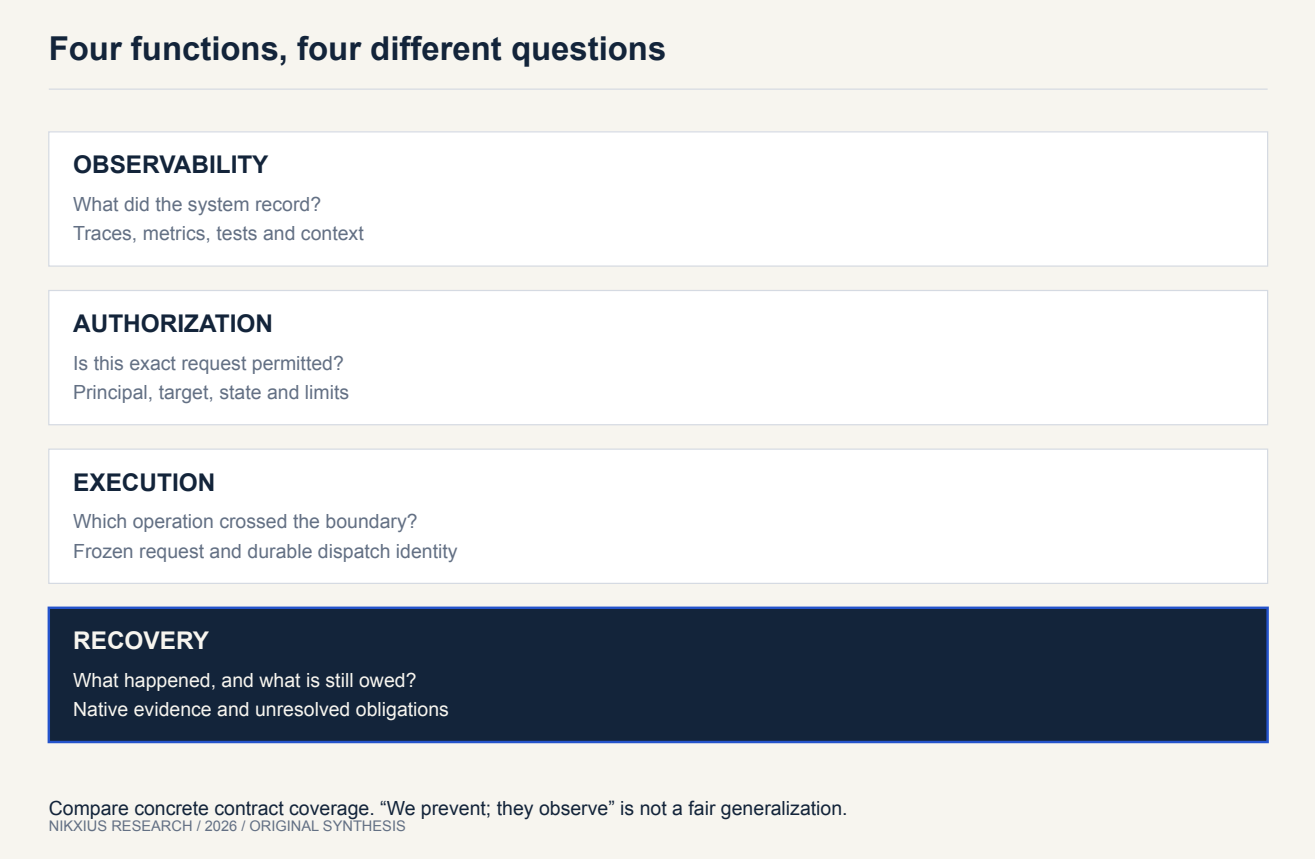

Figure 6. Observability describes what was recorded; authorization decides what is permitted; execution mediates and dispatches the exact native operation; recovery establishes outcomes or retains unresolved obligations. One product may implement several functions.

Function	Useful question	Important limit
Observability	What did the system record, and where did behavior diverge?	A trace alone does not establish that an effect was authorized or completed
Authorization	May this principal request this operation in this context?	Permission is not evidence that dispatch or commitment occurred
Execution control	Which exact operation crossed the controlled writer boundary?	A controlled request can still fail or have an unknown external outcome
Recovery	What can native evidence establish, and what still needs action?	Insufficient history may prevent a conclusive result

The same vendor may implement all four. The useful comparison is coverage of a concrete contract, including bypasses and failures. “We prevent; they observe” is not an honest general description of today’s agent-security landscape.

The separation also improves operator interfaces. A screen should not use one green “success” label for authenticated, authorized, submitted, committed, converged and healthy. Nor should a timeout automatically become a red “failed” label if the remote effect is unknown. Clear state dimensions make escalation and review more reliable.

Evidence follows the same logic. Palantir’s action-log documentation describes a successful-submission log and points to other mechanisms for broader coverage. That is a reminder to inspect the exact record being used, not grounds to claim the entire platform lacks failure visibility.48

17. What platform engineering should do next

Start with one operation that is ready to automate but remains blocked by authority or outcome handling. A broad inventory of all possible agents can be useful elsewhere, but it does not replace tracing one real write path end to end.

Write the action contract in language the platform owner, security reviewer and application owner can all inspect. Identify the native commit point. Specify exactly what a successful API response establishes. Name the observations required for convergence and the additional tests needed for application health. Record the cases where no definitive conclusion is available.

Then build a native baseline. Use existing identity, least privilege, protected delivery, admission policy, native preconditions, workflow durability and audit retention wherever they fit. Record the integration effort and operating burden. If the baseline is adequate, preserve it.

For an additional runtime, test the same cases against the same baseline. Include refusal outside scope, changed parameters under a repeated request key, revocation, concurrent edits, a worker crash, a lost native response, history loss and an unhealthy rollout after a committed mutation. The comparison should measure recovery behavior and operator work, not only time to make a successful API call.

Finally, decide where operational responsibility lives. Who owns a stuck reservation? Who can authorize a new compensating action? What evidence allows an incident to close? What happens if the operator believes the system is healthy but original commitment cannot be established? Those decisions should exist before a production incident forces them.

The production-readiness checklist is a practical starting point. Passing it is a review outcome, not a certification that an autonomous system is safe.

18. What security engineering should ask

The first question is not whether the agent is trustworthy in general. It is which effects the agent can cause through every available route. Review the runtime's credential, native permissions and control over delegation, but also inspect alternate paths such as token creation, Secret access, shell execution, CI automation and privileged helper services.

Next, inspect what crosses the trust boundary. Does the proposer supply an asserted identity that the runtime treats as verified? Can it choose its own policy version, resource identity or prior operation? Can a new session reset a limit that was meant to apply to a root task? Can a tool output cause the next tool invocation to exceed the original authority?

Treat enforcement and business correctness separately. A narrow operation can be faithfully enforced and still be an inappropriate remediation. Policy quality, data quality, model evaluation and application-level judgment remain necessary. No single allow/deny boundary proves the desired business outcome.

Review evidence under a stated trust model. A Node-signed observation is evidence about what that trusted Node recorded. It is not an independent statement from the Kubernetes API server unless the native system supplied such a statement. Verify what is authenticated, what is retained, what can be omitted and what a compromised trusted component could falsify.

Finally, insist on an honest failure model. A system that preserves unknown states may create operational work, but it is easier to assess than a system that silently turns every timeout into a safe-looking verdict. Security review should examine how uncertainty constrains subsequent authority, not merely whether logs are retained.

19. What AI platform teams should keep outside the model

The agent can reason about which operation to propose. The enforcement path should not depend on the agent agreeing that its own action is allowed. OWASP's downstream-mediation guidance supports this separation.19

Keep externally consequential parameters typed and inspectable. Avoid exposing a generic shell when a small operation interface is sufficient. Preserve stable request identity across workflow retries. Carry the outcome vocabulary back to the planner so it can distinguish denied, accepted, committed, degraded and unknown instead of treating every tool response as an unstructured narrative.

This does not require pretending models are deterministic. Bain's banking architecture draws the useful distinction between probabilistic orchestration and deterministic tools or APIs. In practice, deterministic controls mean repeatable validation and explicit state transitions; they do not mean networks never fail or external systems always reach the desired state.49

Evaluation should include whether the agent responds correctly to refusal and uncertainty. Does it create a new request key to circumvent a denial? Does it switch tools to find a broader credential? Does it claim a rollback succeeded when it only received queue acceptance? Does it keep retrying after the contract says that the original effect remains unresolved?

These are workflow tests as much as model tests. A model can propose a safe next step only if the tool interface gives it truthful state, bounded options and a clear distinction between new authority and the continuation of an existing operation.

20. What not to build by default

A useful architecture does not need an independent product at every box in the diagram.

Do not create a second identity directory merely to label principals as agents. Do not create a generic policy engine if an existing one can express and evaluate the required decisions. Do not replace working orchestration simply because the proposer is now an LLM. Do not add a broad connector catalog before knowing which operations require distinct native guarantees.

Do not make approval volume the proxy for safety. An additional click can have little value if the approver cannot see the exact operation or the request changes afterward. Do not confuse replaying a workflow with safely retrying its effects. Do not make “undo” a universal capability where the native system only supports a new compensating action.

Do not position a signed certificate as proof of correctness. Cryptographic evidence is useful when it strengthens integrity, attribution and review of a well-defined lifecycle. It does not repair missing authorization, create native history that was never retained or prove that an application’s recovery succeeded.

Do not replace successful conventional automation just to create an agent deployment. Google SRE’s published principles explicitly preserve automation that already meets business needs.6

The general rule is economical: build the smallest missing operational contract. If a policy change, narrower tool, native precondition or clearer runbook closes the gap, use it.

21. The thesis Nikxius is testing

Nikxius is execution control and recovery infrastructure for production automation. Its customer-controlled Runtime mediates supported operations, binds authority to an exact action and native state, records dispatch, reconciles available outcome evidence and retains unresolved state when evidence is insufficient.

The first evaluation workflow is an agent-proposed Kubernetes Deployment rollback to an explicitly permitted prior image. Platform engineers and SREs are the primary users; security engineering reviews the authority boundary. The company is not claiming to govern every agent or every tool invocation.

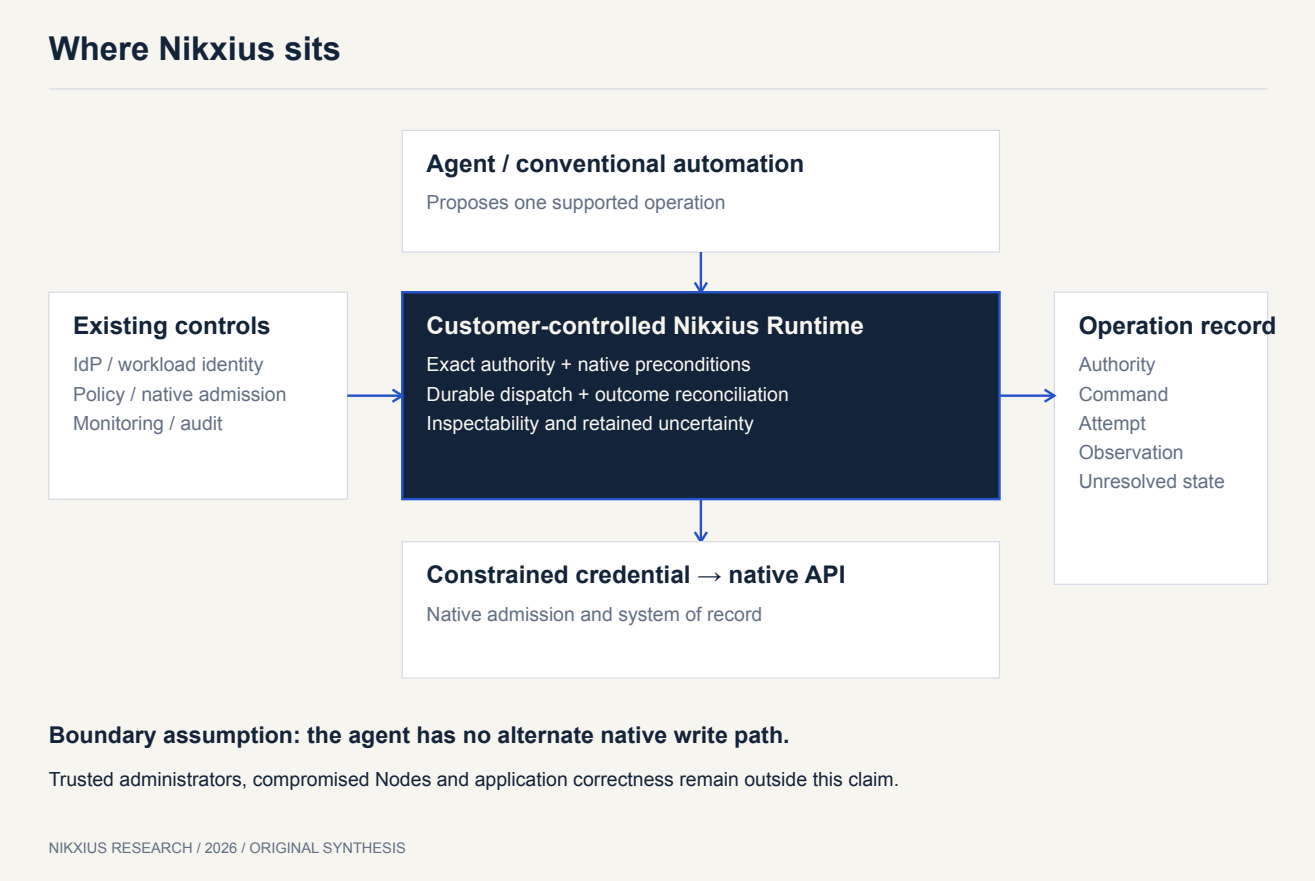

Figure 7. An agent proposes a supported action to the customer-controlled Nikxius Runtime. The Runtime uses a constrained credential to dispatch through the native API. Existing identity, policy, monitoring and audit systems remain alongside it. The boundary depends on controlling alternate write paths.

The current Runtime's explicit rollback profile references the prior committed operation and checks the relationship between old and new images on the same native target. Its lifecycle, commit and observation states remain separate. A resolved operation is not automatically a successful or healthy application outcome. Reconciliation after possible dispatch reads native evidence rather than blindly sending another mutation.39

The local verification record dated 12 September 2026 reports 232 distinct Runtime tests, including 27 native Kubernetes cases, and three recorded demonstration scenarios. These are historical local conformance results. The native environment was disposable Kubernetes 1.35.0, with synthetic identity and MFA fixtures. Database restart and mocked-provider tests have their own stated boundaries. The website replay is not a customer deployment, external security review or production certification.41

The security claim depends on deployment: the customer Node holds the protected write credential, while the agent lacks an alternate native write path. The Runtime does not automatically discover every bypass or protect against a compromised trusted Node or privileged administrator. Signed records attribute recorded observations to an accepted Node key; they do not independently prove native truth or completeness.[39]

The commercial thesis remains a hypothesis. A team should evaluate whether this boundary removes enough bespoke authority and recovery work to justify another component. The proposed evaluation is one supported workflow in one named nonproduction environment over four weeks, with an explicit comparison against the current stack. Scope, responsibilities and fees require a separate agreement.

If the customer's existing stack satisfies the complete contract economically, it should use that stack. Future operation families should be added only when native semantics and customer evidence justify them. Payments remain a possible future family and part of Nikxius's research history; they are not the default identity of the company.

22. Open questions and claims this research cannot support

The literature supports the importance of operational controls. It does not establish a universal priority ranking among integration, cost, data quality, authorization, recovery and workflow redesign. Different organizations will encounter different bottlenecks.

It also does not establish a standalone budget for execution control and recovery. Engineering ownership is a plausible starting point: PwC's Responsible AI survey reports that 56% of 310 US executives assign primary leadership to first-line IT, engineering, data and AI teams. That broad category does not prove that every Head of Platform Engineering owns the purchase or that security and risk are secondary in every organization.[9]

We do not yet know whether buyers will prefer broad platforms, native composition or narrow operation runtimes as the category matures. Existing vendors can deepen their execution and recovery capabilities. Standardization can reduce differentiation in receipts and policy interfaces. NIST's agent standards initiative reflects active work on authentication, identity, protocols and evaluation; it is not a mandate to purchase an independent control layer.[50]

The most important unresolved technical questions concern evidence continuity and deployment boundaries. What native history is economically retainable? Which effects can be conclusively attributed after failure? How should permanently unknown outcomes constrain future operations? How can customers verify that alternate write paths are closed without creating another large integration project?

There are also economic questions. How much review can a bounded contract remove? How frequently do ambiguous outcomes occur in a real workflow? Does one team's implementation generalize to another environment, or does every deployment become consulting? Which state or integration becomes valuable enough that customers prefer a maintained product?

These questions are tests of the thesis, not reasons to fill the gaps with market-size estimates. The research does not prove product-market fit, an unavoidable new platform category or a defensible monopoly for Nikxius.

23. The operational standard to aim for

The useful end state is a production operation that remains understandable before, during and after execution.

Before dispatch, a reviewer can identify the responsible principal, exact action, native target, current preconditions and limits. During execution, the system preserves the admitted operation and its possible effect across worker failure. Afterward, the operator can distinguish what the native system established from what remains unknown, and knows who owns the next step.

The reasoning that selected an operation can remain probabilistic. The authority boundary and lifecycle should be explicit enough to test. External effects will still fail, arrive late or become difficult to attribute. Good execution infrastructure makes those limits operationally visible instead of disguising them as success.

That is the engineering discipline this report argues for: **every consequential agent action should have a defensible execution contract, whether the organization assembles it from existing systems or adopts a dedicated runtime.**

Continue with the Agent Action Contract technical note, the Kubernetes rollback example, or the production-readiness checklist.

Sources

References link to original public publications. Undated documentation is labeled as such; it was reviewed by the 15 September 2026 cutoff. Publisher statements and product documentation are attributed claims, not independent certification of deployed behavior. The full reviewed-corpus bibliography and editorial coding notes are available in the source library.

1. McKinsey & Company, Building the foundations for agentic AI at scale (2026-04). Pages 4 and 9: architecture principles and governance. Publisher guidance or documented behavior; not independent product validation.↩ ↩

2. BCG, Enterprise AI Control Plane: The CIO's Guide to Governing and Accelerating AI Agents (2026-08-14). Technical Components; Lessons Learned. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩ ↩

3. Bain & Company, Will Agentic AI Disrupt SaaS? (2025-09-23). Bottlenecks and the need for common syntax; Strategic priorities. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

4. Oliver Wyman, How AI for enterprise legacy IT runs operations at scale (2026-05-20). Legacy IT operating model; bounded Run IT priorities. Publisher guidance or documented behavior; not independent product validation.↩

5. Bain & Company, The Three Layers of an Agentic AI Platform (2026-04-02). Application/orchestration, analytics and data layers. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

6. Google, AI in SRE: Where and how Google is deploying agentic AI to improve operations (2026-05-29). Design considerations; incident investigation. Publisher guidance or documented behavior; not independent product validation.↩ ↩

7. Bain & Company, Your AI Budget Is Growing. Your Returns Aren't. Here's Why. (2026-06-01). Figures 2 and 4; survey n=951. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

8. Palantir, Overview • Ontology (undated; reviewed 15 September 2026). Ontology building; action types and functions. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

9. PwC, PwC's 2025 Responsible AI survey: From policy to practice (2025-10-30). Governance ownership; agent governance; survey methodology, n=310US leaders. Publisher guidance or documented behavior; not independent product validation.↩ ↩

10. EY / EY-Parthenon, Why agentic AI governance needs real-time trust layers (2026-06-08). Decision rights, monitoring and escalation; EY-hosted article originally published on CIO.com. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

11. KPMG, Agentic AI advantage: Unlocking next-level value (2025-10). Page 23: identity, runtime containment and kill switches. Publisher guidance or documented behavior; not independent product validation.↩

12. Accenture, Strengthening AI agent security with identity management (2025-01-20). Challenges with AI agent identity and recommended controls. Publisher guidance or documented behavior; not independent product validation.↩ ↩

13. KPMG, Invisible access, visible risk (2025-12-19). Embedding NHI governance into enterprise identity strategy. Publisher guidance or documented behavior; not independent product validation.↩ ↩

14. PwC, Agents of change - When AI agents lose their way: How to manage the insider threat (undated; reviewed 15 September 2026). Pages 2–4: adapt incident response, preparation and recovery. Publisher guidance or documented behavior; not independent product validation.↩ ↩

15. Palantir, Action types • Permissions (undated; reviewed 15 September 2026). Read-time enforcement; submission criteria; side-effect permissions. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩ ↩

16. KPMG, Global AI Pulse Q2 2026 (2026-06). Pages 18–19 and 22: cost-response chart, visibility and methodology; n=2,145. Publisher guidance or documented behavior; not independent product validation.↩ ↩

17. Oliver Wyman, What 200 senior executives told us about agentic AI: The Oliver Wyman 2026 Parallel Surveys on CEOs and Tech Leaders on Agentic AI and IT (2026-09-08). Companion survey PDF, pages 6–7,13–14; separate samples of 130 technology leaders and 70CEOs/executive-committee members. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

18. Oliver Wyman, Why government AI success starts with simpler operations (2026-07-22). Simplifying institutional friction before automation. Publisher guidance or documented behavior; not independent product validation.↩ ↩

19. OWASP, LLM 06:2025 Excessive Agency - OWASP Gen AI Security Project (2025). Prevention and Mitigation Strategies, especially complete mediation. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

20. Microsoft, Authorization in Microsoft Entra Agent ID | Microsoft Learn (undated; reviewed 15 September 2026). Microsoft Graph permissions for agent IDs; delegated and application permissions. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

21. AWS, Policy in Amazon Bedrock AgentCore: Control Agent Interactions - Amazon Bedrock AgentCore (undated; reviewed 15 September 2026). Gateway policy boundary; key benefits and features. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

22. Palantir, Action types • Submission criteria (undated; reviewed 15 September 2026). Submission criteria, including scoped-token cautions. Publisher guidance or documented behavior; not independent product validation.↩ ↩

23. Stripe, Advanced error handling | Stripe Documentation (undated; reviewed 15 September 2026). Network errors, server errors and idempotency. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

24. Palantir, Action types • Side effects • Webhooks (undated; reviewed 15 September 2026). Writeback versus side-effect webhooks. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

25. PCAOB, AS 1105: Audit Evidence | PCAOB (undated; reviewed 15 September 2026). AS 1105 paragraphs.04–.08: sufficiency, relevance and reliability. Publisher guidance or documented behavior; not independent product validation.↩

26. IETF / RFC Editor, RFC 9943: An Architecture for Trustworthy and Transparent Digital Supply Chains | RFC Editor (2026-06). Sections 5.1.1,7 and 9.2: registration policy, validation and accuracy of statements. Publisher guidance or documented behavior; not independent product validation.↩

27. AWS, Temporal policies - Amazon Bedrock AgentCore (undated; reviewed 15 September 2026). Temporal policy sessions and Security considerations. Publisher guidance or documented behavior; not independent product validation.↩ ↩

28. Accenture, Agentic Commerce and the Future of Payments (2026-05-27). The protocols challenge; Own the control layer. Publisher guidance or documented behavior; not independent product validation.↩

29. Google AP2 project, Agent Authorization Framework - AP 2 - Agent Payments Protocol Documentation (undated; reviewed 15 September 2026). Mandate Structure; Action Authorization and Mandate Receipt result. Publisher guidance or documented behavior; not independent product validation.↩

30. Open Policy Agent, Philosophy | Open Policy Agent (undated; reviewed 15 September 2026). What is OPA?; policy decoupling and document model. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩

31. Temporal, Activity Definition | Temporal Documentation (undated; reviewed 15 September 2026). Idempotency; worker completion acknowledgement and native keys. Publisher guidance or documented behavior; not independent product validation.↩ ↩ ↩ ↩ ↩

32. Modern Treasury, Approval Rules Overview (undated; reviewed 15 September 2026). Approval rule definitions, sequencing, reviewer and update controls. Publisher guidance or documented behavior; not independent product validation.↩

33. GitHub, Deployments and environments - GitHub Docs (undated; reviewed 15 September 2026). Deployment protection rules, required reviewers, administrator bypass and environment secrets. Publisher guidance or documented behavior; not independent product validation.↩

34. Temporal, What is a Temporal Retry Policy? | Temporal Documentation (undated; reviewed 15 September 2026). Default Activity and Workflow retry behavior. Publisher guidance or documented behavior; not independent product validation.↩ ↩

35. Palantir, AIP Evals • Overview (undated; reviewed 15 September 2026). AIP Evals overview and core concepts. Publisher guidance or documented behavior; not independent product validation.↩

36. Kubernetes project, Kubernetes API Concepts | Kubernetes (undated; reviewed 15 September 2026). Resource versions; conditional updates and lost-update detection. Follow the documented version and resource-type conditions. Relevant sections of first-party material reviewed by 15 September 2026.↩ ↩

37. Kubernetes project, Using RBAC Authorization | Kubernetes (undated; reviewed 15 September 2026). Role and ClusterRole; native resource/verb permissions. Relevant sections of first-party material reviewed by 15 September 2026.↩

38. Kubernetes project, Admission Control in Kubernetes | Kubernetes (undated; reviewed 15 September 2026). Admission control phases; authentication/authorization and mutation/validation. Relevant sections of first-party material reviewed by 15 September 2026.↩

39. Nikxius, Current Runtime contract and supported operation scope (undated; reviewed 15 September 2026). Current supported-operation matrix, state dimensions, recovery and deployment assumptions. Product documentation, not independent validation. See also the Kubernetes rollback and security documentation.↩ ↩ ↩ ↩

40. Kubernetes project, Deployments | Kubernetes (undated; reviewed 15 September 2026). Rollout progress, status conditions and rolling back a Deployment. Relevant sections of first-party material reviewed by 15 September 2026.↩

41. Nikxius, Historical local Runtime verification record (undated; reviewed 15 September 2026). Local conformance compiled 12 September 2026; underlying record timestamp 2026-09-11T23:10:57.822419+00:00. 232 distinct Runtime tests, 27 native Kubernetes cases and three demonstration scenarios; disposable Kubernetes 1.35.0 and synthetic identity fixtures. Historical local evidence, not customer production or certification.↩ ↩

42. Microsoft, agent-governance-toolkit/policy-engine/spec/SPECIFICATION.md at main · microsoft/agent-governance-toolkit · GitHub (undated; reviewed 15 September 2026). Model and invariants; host integration boundary. Draft specification, engine 0.4.0-alpha.3 and manifest 0.4.0-alpha.1. Relevant sections of first-party material reviewed by 15 September 2026.↩

43. Airia, AI Agent Execution Control: How to Move Beyond Output Monitoring for Agentic AI Security | Airia (2026-07-07). The Execution Control Model; pre-execution policy and review. Vendor-reported positioning. Relevant sections of first-party material reviewed by 15 September 2026.↩

44. Snyk, The New Security Control Point: Governing AI Agents Inside the Execution Loop | Snyk (2026-06-23). Applying governance in real time. Feature described as Open Preview. Relevant sections of first-party material reviewed by 15 September 2026.↩

45. OWASP, AI Agent Security - OWASP Cheat Sheet Series (undated; reviewed 15 September 2026). High-impact action controls: exact-action approval, short-lived artifacts and replay protection. Relevant sections of first-party material reviewed by 15 September 2026.↩

46. Arcade, Arcade: The Actions Runtime for Enterprise AI Agents (undated; reviewed 15 September 2026). Enforce, Execute and Govern; custom policy hooks. Publisher guidance or documented behavior; not independent product validation.↩

47. Cohesivity, AI Agent Failure Recovery: Retries, Checkpoints, and Approval — cohesivity (2026-07-29). Logical operation IDs, unknown outcomes, reconcile-before-retry and separate compensation. Technical article, not independent product validation. Relevant sections of first-party material reviewed by 15 September 2026.↩

48. Palantir, Action types • Action log (undated; reviewed 15 September 2026). When to use the action log; successful submissions and other record facilities. Publisher guidance or documented behavior; not independent product validation.↩

49. Bain & Company, What It Takes to Build the AI-Native Modern Bank (2026-07-22). Modern technology and data stack: deterministic/probabilistic boundary. Publisher guidance or documented behavior; not independent product validation.↩

50. NIST, AI Agent Standards Initiative | NIST (2026-02-17). Strategic Pillars; voluntary guidelines, protocols and identity research. Publisher guidance or documented behavior; not independent product validation.↩