Nikxius Inc.

NIKXIUS RESEARCH · SEPTEMBER 2026

The Agent Action Contract

Authority, native state, dispatch and recovery for consequential operations

REPORT
NIKXIUS-ACTION-CONTRACT-2026

DATE
2026-09-15

VERSION
1.0

CLASSIFICATION
PUBLIC

ABSTRACT

A technical framework and current Nikxius implementation mapping. The framework is a synthesis of established engineering concepts, not an interoperable standard.

Contents

Scope

An agent action contract connects permission to a specific native effect and defines how that effect will be accounted for. It is useful whenever an agent or conventional automation can change consequential external state.

This note distinguishes a **conceptual framework** from the **current Nikxius Runtime API.** The framework is not a new standard, credential format or claim of invention. It combines least privilege, conditional updates, durable execution identity, idempotency, explicit outcome states and recovery procedures.

Use an existing native stack if it satisfies this contract economically. Policy engines, identity systems, orchestration and native APIs already provide substantial pieces. The relevant test is their combined behavior for one operation.

The contract record

A complete conceptual contract answers the following questions. A deployment can distribute the fields across existing systems; they must still refer to the same operation.

Element	Required meaning
Principal and delegator	Verified identities, accepted issuer, scope of delegated authority and accountable owner
Operation	Exact type, version, target, parameter set and externally meaningful effect
Native baseline	Trusted resource identity, version and operation-specific state assertions
Limits	Expiry, quantity, concurrency and task-level capacity; behavior across sub-agents and retries
Policy and review	Applicable policy snapshot and any approval bound to the exact command
Dispatch	Frozen semantic command, native request, attempt identity and durable possible-send record
Outcome	Separate native commitment, later observation and integrity classifications
Recovery	Accepted evidence, safe retry rules, reconciliation owner and permitted compensation

Element	Required meaning
Evidence	Retained records, provenance, signing trust and known omissions

This is not a proposal to accept arbitrary caller-supplied values for every field. In particular, a client must not choose its verified principal, tenant, trusted resource identity or accepted policy simply by placing them in JSON.

Authority is narrower than identity

Authentication establishes the principal accepted by the service. Authorization determines whether that principal may request this effect in this context. Native credentials determine what the writer can physically cause. These are different boundaries.

In the current Runtime, identity registration determines tenant scope. A task grant binds the registered workload to an enrolled operation, exact target, permitted effect and expiry. Policy may require additional human authorization; zero approvals is a valid configured policy outcome. The Node holds the protected native credential, and the agent must lack an alternate write path.

OWASP recommends downstream authorization and complete mediation rather than relying on the LLM to decide whether it is allowed to act. This does not replace prompt-injection defenses or model evaluation; it constrains the effects of a bad proposal through a separately enforced boundary.[1]

Bind native preconditions, not cached reasoning

Resolve the native object independently. Check the resource identity as well as its display name. Freeze the properties that define the allowed baseline, and use native conditional-update semantics to refuse stale mutations.

Kubernetes documents resource-version preconditions for detecting lost updates. Current documentation also defines supported resource-version ordering under specific version and resource-type conditions. The Nikxius adapter uses equality against its captured baseline rather than inferring commitment from numeric ordering. A later resource version alone is not evidence that this particular operation performed the transition.[2]

A precondition can protect only what it checks. An image mutation can be bounded without establishing database-schema compatibility. Native admission components and other controllers may change behavior outside the checked invariant set. Those assumptions belong in enrollment and security review.

A current Runtime proposal

The following is an illustrative request shape for the private customer Runtime. It is not sent to nikxius.com, and its synthetic values are not a usable grant, operation or image artifact.

```
{
  "operation": "kubernetes.deployment.rollback_image.v1",
  "target": {
    "cluster": "evaluation-cluster",
    "namespace": "checkout",
    "deployment": "checkout-api",
    "container": "api"
  },
  "input": {
    "image": "registry.example.com/checkout-
api@sha256:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
    "compensatesOperationId": "01993100-0000-7000-8000-000000000001",
    "expectedResourceVersion": "4812"
  },
  "grantRef": "01993100-0000-7000-8000-000000000002",
  "idempotencyKey": "checkout-rollback-evaluation-001"
}
```

The rollback grant and enrollment must permit that digest and prior-operation relationship. The referenced operation must be committed in the same tenant, on the same target and UID. Its before-image must equal the requested rollback image; its after-image must match the current baseline being reversed. The runtime does not infer permission from arbitrary rollout history or run a generic `rollout undo`.

The API independently validates all fields. A raw native URL, shell command, manifest, patch, caller principal or tenant field is not accepted as a substitute for the registered contract. Optional expected-state fields are assertions checked against native state, not trusted facts.

Commit point and durable identity

Before possible native dispatch, the Runtime records the frozen command, its native request binding and attempt identity. This reduces the crash window in which software can send an effect without a durable local operation identity. It does not create an atomic transaction between PostgreSQL and Kubernetes.

The proposal's idempotency key is scoped to tenant and actor. Repeating the identical request under that key returns the same durable operation. Changing its input or grant reference under the same key is refused. A new key does not replenish a root task's consumed or uncertain capacity.

The semantic command hash and native dispatch hash serve different purposes. Review binds the command and dispatch being approved; the writer must honor that exact binding. Recording a hash proves neither mental intent nor native commitment.

After a durable dispatch claim exists, a restarted worker must regard the native request as potentially sent. A worker lease can coordinate local ownership but cannot retract a request already in the network or accepted remotely. A lost PostgreSQL COMMIT response similarly does not establish that the database transaction rolled back.

The state model

Keep these current Runtime dimensions separate:

Dimension	Values	Interpretation
Lifecycle	PROPOSED, AWAITING_APPROVAL, REVALIDATING, ADMITTED, DISPATCH_RECORDED, PENDING, UNKNOWN, RESOLVED, DENIED	Coordinates the local operation lifecycle
Commit	not_dispatched, committed, rejected, unknown	What can be established about the declared native effect
Observation	not_applicable, pending, converged, degraded, superseded, unobserved	Later native/controller state, separate from commitment
Reservation	none, reserved, consumed, released	Task/resource capacity accounting
Integrity	unchecked, matched, mismatch	Whether inspected invariants match the admitted contract

`RESOLVED` does not mean business success. A no-change operation may resolve without a patch or consumed commit unit. A committed image update can have a degraded rollout. An invariant mismatch after commitment is a committed integrity incident; it must not be rewritten as an unexecuted action.

Kubernetes Deployment conditions describe rollout progress and failure, but an accepted specification update is not sufficient evidence of application health.[3]

UNKNOWN and reconciliation

A request timeout describes lost communication, not necessarily a failed mutation. Preserve the operation and its possible effect. Do not create a fresh logical operation simply to obtain a cleaner response.

The general framework permits retry only under the writer's documented contract. Stripe, for example, recommends same-key/same-parameter retries for network failures while describing certain server-error outcomes as indeterminate. There is no universal rule that all unknown effects are safely repeatable or that none can ever be retried.[4]

The current Nikxius Kubernetes path makes a narrower choice: after possible dispatch, reconciliation **reads**. It does not blindly reissue the mutation. A valid native response or sufficient retained continuous native history can establish the original transition. A matching current image, inherited annotation, missing object or relist after a history gap cannot establish original causation alone.

Kubernetes explicitly documents limited watch history and the need to relist after an unavailable historical version. Relisting restores a current view; it does not recreate a transition that is no longer available.[2]

When evidence remains insufficient, UNKNOWN can remain indefinitely. The operation keeps its relevant task/resource reservation and needs an operational owner. There is no current "mark successful" or "reset unknown" API. An operator's confidence is not a substitute for the missing evidence.

Revocation, concurrency and break-glass

Revalidate grant, policy, review and native baseline before dispatch under the documented lifecycle. A changed command may require a fresh review. Revoking authority prevents newly permitted work; it cannot necessarily cancel an already accepted native request or release an uncertain reservation safely.

The resource slot prevents conflicting enrolled work within the runtime's scope. It does not create a cross-system transaction or automatically exclude every external controller. Native permissions and operational ownership must establish the larger boundary.

Break-glass access belongs to the customer's incident process. It should have an independent authorization path and retained evidence. Its use must not rewrite the original operation as never executed or force an unknown commit into success. New emergency work may change the resource while leaving the earlier attribution unresolved.

Compensation is another action

A rollback is not deletion of history. It needs a fresh operation, grant, native preconditions, review where required and an outcome of its own. It can fail or remain unknown.

An image reversal does not undo schema changes, restore lost data or establish healthy dependencies. The contract must identify what the compensation changes and what it cannot restore. That limit applies even when the earlier operation was correctly authorized.

API and integration boundary

These endpoints belong to the customer-hosted Runtime, not the public website:

Endpoint	Contract
POST /v1/operations	Durable proposal; 202 for creation, 200 for identical replay; not completion
GET /v1/operations/:id	Operation, approvals, attempts, observations and events
POST /v1/operations/:id/approve	Exact-command review with current version/hash binding
POST /v1/operations/:id/reconcile	Queue read-only reconciliation; body `{}`
GET /v1/operations/:id/evidence	Export recorded lifecycle and assurance limits
POST /v1/grants/:id/revoke	Record revocation; do not recall an accepted native request

The existing private Node client is a convenience wrapper, not a separately published public SDK or a client-side enforcement boundary. It does not make effect semantics interchangeable. Transport uncertainty must preserve the original immutable request/key and inspect durable state.

Evidence and validation

Evidence should retain verified identity context, accepted grant, policy, exact-command binding, required review, dispatch record, native observations, state transitions and unresolved limitations. Optional Node signing attributes those records to an accepted key. It does not independently establish native truth, complete capture or application correctness.

The public verification record is dated 12 September 2026 and reports local conformance, including disposable Kubernetes tests and synthetic identity fixtures. It is not evidence of customer production acceptance. The recorded website walkthrough demonstrates `set_image`, not a production rollback deployment.[5]

Evaluate the contract with refusal outside scope, stale UID/version, changed payload under an existing key, revoked authority, concurrent edits, a worker crash, lost response, retained-history recovery, history loss and an unhealthy rollout after commitment. Compare the same cases against the native stack. Measure operator work and unresolved obligations as well as successful dispatch.

Read the full research report, Runtime scope, security boundary and bounded rollback example.

---

1. OWASP, LLM06:2025 Excessive Agency, 2025, mitigation guidance including downstream authorization and complete mediation.↩
2. Kubernetes project, Kubernetes API Concepts, “Updates to existing resources,” “Resource versions,” and watch-history semantics. Original publication date unknown; mutable documentation reviewed 15 September 2026.↩ ↩
3. Kubernetes project, Deployments, rollback and Deployment-status sections. Original publication date unknown; reviewed 15 September 2026.↩
4. Stripe, Advanced error handling, network errors, idempotency and indeterminate server-error results. Original publication date unknown; reviewed 15 September 2026.↩
5. Nikxius, Runtime verification record, local conformance recorded 12 September 2026. Product evidence with explicit environment and assurance limits.↩