AIP operator runbooks

Use these runbooks for the first response to a common getaip-server, connector-fleet, or connector-host incident. Select the symptom below, contain new effects, use the recorded checks to identify the failing owner, and stop when the eviden

SECTION	Deploy and Operate
TYPE	Guide
PROTOCOL	AIP 1.0
SOURCE	docs/guides/operator-runbooks.md

The procedures apply to source revision `d7cce13d1d555644d04a4d73c66c95b113737635`. They do not define deployment-specific alert thresholds, database failover commands, provider operations, or incident ownership. Use Observe and recover AIP work for the exact query flow and evidence hierarchy.

Select the runbook

Symptom	Start here	Immediate containment
Core `/ready` returns `503`	Core daemon is not ready	Stop new traffic to the unready replica
Worker cycles stop or action leases do not progress	Runtime worker or action lease is stalled	Pause affected mutations; preserve action IDs
Runtime storage fails or a database failover begins	Runtime database is unavailable	Stop new mutations for that state owner
Fleet snapshot fails or no route is available	Fleet maintenance or route resolution fails	Keep affected bindings disabled or traffic paused
Host `/ready` reports an invalid lease	Connector-host lease is lost	Remove the replica from new traffic; keep its state
Remote queue or event ingress reaches capacity	A bounded admission queue is saturated	Apply backpressure at the authenticated edge
A transaction is `outcome_unknown` or a duplicate is suspected	Provider outcome is uncertain	Stop retries for the original mutation
Callback or remote delegation is dead-lettered	Delivery retry budget is exhausted	Do not rerun the source action
A new connector rollout regresses	Connector rollout must be rolled back	Stop new assignments to the affected revision
Signature or replay validation alerts	Trust or replay boundary is failing	Reject the traffic; preserve redacted evidence

Before any runbook, retain the action, transaction, approval, delegation, callback, message, tenant, and connector-route identifiers that exist. Never copy credentials or governed payloads into an incident record.

Core daemon is not ready

Trigger: `/health` succeeds but core `/ready` returns `503`, or the load balancer has removed a `getaip-server` replica.

Checks:

1. Read `gateway.ready`, storage, and local connector details.
2. Read `supervisor.runtime_worker_running`, `nats_required`,

`nats_listener_running`, `last_worker_error`, and the last worker success.

3. Inspect every required local module. Do not treat an optional fleet snapshot

as part of the top-level readiness calculation.

4. Confirm that the process still uses the intended runtime store, trust files,

and authenticated edge.

Contain: keep the unready replica out of traffic. If all replicas share the same failing state owner, pause new actions instead of sending them to a fresh empty runtime.

Recover: restore read/write access to the configured store, the required NATS listener, or the required local module. The runtime worker supervisor restarts a failed child after a bounded delay. Repeated restart with the same error means the dependency remains broken.

Verify: require core `/ready` to return `200`, observe a new worker success and advancing worker cycles, then read one known action from the durable store.

Do not: switch a clustered runtime to memory or a new file directory merely to obtain a ready response. That creates a second action and idempotency owner.

Runtime worker or action lease is stalled

Trigger: queued actions age, running actions stop updating, or `runtime_worker_running` is false.

Checks:

1. List the affected queued and running actions.
2. For each action, compare `updated_at`, retry `next_attempt_at`, lease

owner, lease ID, and lease expiry.

3. Distinguish a future retry window, pending approval, and provider

reconciliation from a worker stall.

4. Compare worker cycles, claimed-action activity, last success, and last error.
5. Check runtime-store latency and clock health before increasing concurrency.

Contain: pause new mutations for the affected capability when lease ownership or provider outcome is uncertain. Keep read paths available.

Recover: restore the worker dependency and let the existing worker loop claim only actions whose retry window has opened and whose lease is absent or expired. The storage backend must fence the claim atomically.

Verify: the original action receives a new durable update or reaches a terminal state. Confirm that no second action ID was introduced.

Do not: clear a valid lease, edit its expiry, or copy the queued payload to a new action. A stale worker may not settle work after it loses the lease.

Runtime database is unavailable

Trigger: gateway storage readiness fails, durable reads fail, or the runtime PostgreSQL owner enters failover.

Checks:

1. Identify whether the deployment uses PostgreSQL or one dedicated local

storage directory.

2. Confirm which replicas share the logical runtime and which database role

they use.

3. Record active actions, leases, approvals, transactions, callbacks,

delegations, replay claims, and provider operation references from the last trustworthy view.

4. Determine whether the database authority has fenced the former primary.

Contain: stop admitting new mutations for this logical runtime. Keep old replicas from writing until the database failover procedure establishes one primary owner.

Recover: complete the deployment's database failover or restore the exact local state directory. Start one reviewed `getaip-server` replica against that state owner and allow runtime recovery to inspect queued actions, approvals, transactions, callbacks, and delegations. Reconcile unknown provider outcomes before restoring full concurrency.

Verify: core readiness passes, action and idempotency history remains queryable, lease ownership is unique, and retained provider references still match their transactions.

Rollback: if the new database owner is not authoritative, stop the new writer before returning to a previously fenced primary. Never run two writable histories and attempt to merge them through action replay.

Fleet maintenance or route resolution fails

Trigger: the optional fleet snapshot reports repeated failures, tenant capability discovery omits an expected route, or new actions return no usable connector assignment.

Checks:

1. Inspect `fleet.worker_running`, consecutive failures, last error, last

success, and the last aggregate summary.

2. Check the registry data-plane connection without substituting an

administrator credential.

3. For new work, verify the tenant binding, active connector type and version,

ready non-expired replica, remaining capacity, and topology constraint.

4. For existing work, read the durable assignment for the original action.

Current discovery describes only routes available for new assignments.

Contain: keep the affected binding disabled or pause new actions. Existing assignments remain pinned and must not be moved to the replica that currently looks healthiest.

Recover: restore the registry data path. Let the fleet maintenance worker expire stale replica leases in bounded cycles and refresh its summary. Restore or register replicas through their signed lifecycle path.

Verify: the fleet snapshot records a later successful cycle, tenant discovery returns the intended capability, and a new non-mutating action obtains one assignment. Verify old ambiguous actions on their original routes.

Do not: write registry status or assignment rows manually. Fleet failures do not make the top-level core readiness response fail at this revision.

Connector-host lease is lost

Trigger: a standalone host is alive but `/ready` reports `lease_valid: false`, the registry marks the replica expired, or connector event publication stops after lifecycle-control failure.

Checks:

1. Read host `draining`, lease validity, runtime storage readiness, connector

readiness, and recovery summary.

2. Confirm whether the lifecycle control plane is reachable and still trusts

the admitted replica identity.

3. Compare the last lease sequence and expiry with the registry view.
4. Record active assignments and unresolved provider work.

Contain: remove the host from new traffic. Do not discard its durable runtime state or provider operation references. An invalid lease also blocks the common host's event-outbox publication.

Recover: when storage and connector health are ready, the heartbeat worker attempts lease recovery by re-registering the same immutable identity. Restore the lifecycle path and allow that idempotent recovery to finish. If admission rejects the identity, correct the package, configuration, or capacity conflict instead of weakening the check.

Verify: host `/ready` returns `200`, the registry shows a current lease and health revision, and retained work remains on the same logical replica.

Fail over: use a distinct replica identity for planned replacement. Reuse the failed identity only after a clean offline transition or lease expiry and only with the same durable recovery boundary.

A bounded admission queue is saturated

Trigger: remote dispatch reports admission-capacity errors, remote queued work or bytes stay near their configured bounds, or connector-event ingress rejects work because no permit is available.

Checks:

1. Distinguish global remote queue, per-tenant queue, queued-byte, queue-age,

and active-dispatch limits.

2. Compare active and queued tenants to identify whether one tenant is

consuming its own bound or the shared scheduler is saturated.

3. Check provider latency, registry latency, host capacity, and external rate

limits.

4. For event ingress, compare configured capacity with available permits and

receiver latency.

Contain: honor the typed retry guidance at the authenticated edge and apply tenant-scoped backpressure. Do not allow an unbounded upstream queue to hide the rejection.

Recover: restore the slow dependency or reduce accepted traffic. Change a limit only after measuring memory, request size, provider concurrency, tenant fairness, and downstream rate limits.

Verify: queued work and bytes decrease, permits become available, the oldest admitted request stays within the intended queue-age bound, and other tenants make progress.

Do not: treat process-local remote queue metrics as the durable action queue. A client retry still needs the original action and idempotency identity.

Provider outcome is uncertain

Trigger: a transaction is `outcome_unknown`, the connection failed after a provider request, or two external effects may represent one intended action.

Checks:

1. Freeze automatic and manual retry for the action and idempotency key.

2. Read the transaction, provider operation reference, reconciliation cursor,

action input hash, tenant, external account, and connector assignment.

3. Query provider facts through an authorized, provider-specific reconciliation

operation.

4. Compare receipts and audit events with provider timestamps and identifiers.

Contain: block another commit and keep the original connector route available. An unavailable response is not evidence that the provider rejected the mutation.

Recover: record the authoritative provider result in the existing transaction through reconciliation. Resume, fail, or start a separately governed compensation only after the outcome is known.

Verify: the transaction reaches a reconciled terminal status and the provider shows one intended effect. If the provider offers no reliable lookup, escalate for human resolution without guessing.

Do not: compensate and retry in parallel. Compensation is not a substitute for identifying the original outcome.

Delivery retry budget is exhausted

Trigger: a callback delivery or remote delegation reaches `dead_lettered`.

Checks:

1. Confirm the source action's terminal state.
2. For callbacks, inspect target, profile, attempts, last error,

dead-letter reason, and receipt chain.

3. Check DNS, TLS, allowlist, receiver authentication, signature validation,

response status, and receiver idempotency.

4. For remote delegation, correlate the parent action, child identity, peer

observations, action events, and audit records.

Contain: keep the source action terminal and preserve the delivery record. Do not rerun the action to regenerate a message.

Recover: restore the destination first. Pending callback deliveries and expired callback leases recover through the worker. Dead-lettered callback and delegation records are terminal for automatic recovery, and this revision exposes no public replay command for either outbox. Use an approved, deployment-specific redelivery procedure or escalate.

Verify: the receiver acknowledges exactly one message with the intended delivery identity and the incident record links that evidence to the original action.

Connector rollout must be rolled back

Trigger: a newly admitted connector version, configuration revision, or replica causes health, correctness, or provider failures.

Checks:

1. Identify the exact package revision, artifact digest, connector version,

instance, binding policy revision, and affected replicas.

2. Separate new unassigned work from actions already pinned to the revision.

3. Record active assignments, outcome-unknown actions, delivery outboxes, and

provider jobs.

4. Confirm that the previous artifact and admission package remain trusted.

Contain: pause affected mutations and apply a signed binding revision that prevents new assignments to the failed path. Do not revoke a whole package unless the incident requires that wider security boundary. If the artifact is no longer trusted, stop further provider execution and reconcile retained work from independent provider evidence.

Recover: express rollback as a newer desired-state generation. Start distinct replicas for the known-good immutable version, verify their registration and readiness, and enable new assignments through a reviewed binding. Begin drain on the failed-version replicas and keep their processes and durable state available while pinned work finishes or reconciles.

Verify: discovery sends new work only to the intended version, old assignments are terminal or reconciled, and active assignment counts reach zero before the failed processes stop and their replicas go offline.

Rollback the rollback: create another newer generation. Do not decrement a generation, reuse a replica identity for another digest, or erase the execution journal.

Trust or replay boundary is failing

Trigger: signatures fail, an unexpected signer appears, replay claims increase, or a valid-looking request is rejected for time or identity context.

Checks:

1. Preserve the message ID, signer DID, claimed sender, recipient, timestamp,

authenticated edge identity, tenant, and redacted typed error.

2. Verify clock synchronization and the configured acceptance window.
3. Read the durable replay claim before resending the same message.
4. Compare the signer-to-principal binding, trusted identity revision, scopes,

and recent key rotation.

5. Determine whether the message was altered, duplicated, delayed, or signed by

a key outside the current trust set.

Contain: reject the affected traffic and isolate a suspected key or edge. Keep unrelated tenants inside their own verified trust boundaries.

Recover: correct clock or trust data through the reviewed configuration owner. Rotate or revoke a compromised signer, then verify the boundary with a fresh signed, non-mutating request. Retry a rejected mutation only after its durable action state proves that no accepted or ambiguous effect exists.

Verify: signatures bind to the expected principal, replay claims remain durable across restart, and no unsigned exception was introduced.

Do not: disable signature, replay, tenant, or authorization checks to restore availability.

Retain evidence and escalate

Retain:

- source revision, artifact digest, deployment revision, and configuration

class;

- action, approval, transaction, delegation, callback, receipt, and route

identifiers;

- redacted typed errors, provider correlation IDs, event cursors, and

timestamps;

- readiness and metrics snapshots from before and after recovery;
- the containment, recovery, verification, and rollback decisions;
- the approving operator and the owner accepting any residual uncertainty.

Escalate when the provider outcome cannot be established, the authoritative state owner is disputed, a dead-lettered delivery needs a new effect, or a trust key may be compromised. Also escalate if recovery would change an existing action identity or route.

Related documentation

- Errors and retry decisions
- Metrics reference
- Operate the connector-host lifecycle
- Orchestrate and roll back connector hosts
- Deploy AIP in production