Twenty health, observability, and recovery

Use this runbook when a Twenty replica is unhealthy, not ready, losing its lease, holding an Action without trustworthy terminal evidence, or failing to publish a verified webhook event.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/twenty/operations/health-observability-and-recovery.md

Preserve PostgreSQL, provider, registry, Action, replay, event, and outbox evidence before changing the deployment. A green signal at one boundary does not prove the others.

Start with the four boundaries

Boundary	Primary evidence	Healthy meaning	Excluded meaning
Host process	`GET /health`	Router responds with static AIP host identity	Storage, lease, or provider readiness
Fleet replica	`GET /ready`	Not draining; lease, storage, and connector probe pass	Every Twenty operation works
Provider Action	Durable Action, checkpoint, result, receipts, provider request ID	Evidence agrees on one terminal intent	Downstream automation completed
Webhook event	Replay scope, local event, outbox batch, central event ID	One delivery is durably owned	Every provider event will arrive

Check process and routing boundaries before tenant traffic. Investigate Action and webhook evidence by their original identities rather than submitting replacement work.

Read host liveness

The host `/health` response is static:

```JSON
{
  "status": "ok",
  "protocol": "AIP",
  "version": "1.0",
  "service": "aip-connector-host"
}
```

Use it only to determine whether the process and HTTP router respond. Failure points first to process termination, listener binding, ingress, resource limits, or container state.

Do not use `/health` as a provider, PostgreSQL, lease, credential, operation, webhook, or artifact check. The generic image healthcheck probes this static route.

Read routing readiness

The host `/ready` endpoint returns HTTP `200` only when all conditions are true:

1. the replica is not draining;
2. its registry lease remains valid;

3. durable runtime storage is readable and writable;
4. the Twenty connector health probe succeeds within its deadline.

Otherwise it returns HTTP `503` with structured state:

```
JSON
{
  "status": "not_ready",
  "lease_valid": false,
  "draining": false,
  "runtime": {
    "durability": "durable",
    "ready": true,
    "detail": "durable runtime storage is readable and writable"
  },
  "connector": {
    "ready": true,
    "detail": "Twenty authenticated workspace API is reachable"
  }
}
```

The response also includes the runtime recovery report. Use each field rather than reducing all readiness loss to one provider alarm.

Remove a not-ready replica from new assignments. Preserve existing Action and webhook identities until durable state is reconciled.

Understand the provider health probe

The connector sends authenticated `GET /rest/open-api/core` to its fixed origin. A successful HTTP status reports ready with the detail shown above.

The probe verifies:

- provider DNS and network path at that moment;
- TLS and fixed origin routing;
- current API token acceptance for the core OpenAPI route;
- provider response status within the health timeout.

It does not decode the OpenAPI body or verify workspace data. It does not test record mutation, metadata permission, provider schema, idempotency, webhook registration, event delivery, response-size handling, or the pinned provider revision.

A credential can pass the probe while lacking permission for another admitted operation. Diagnose the exact Action separately.

Monitor host metrics

Host `/metrics` exposes eight Prometheus text metrics without labels:

Metric	Type	Interpretation
`aip_connector_host_ready`	Gauge	Cached conjunction of lease, storage, connector, and drain state
`aip_connector_host_storage_ready`	Gauge	Cached durable runtime storage result
`aip_connector_host_in_flight`	Gauge	Actions currently held by the host limiter
`aip_connector_host_requests_total`	Counter	Native AIP requests observed
`aip_connector_host_rejected_total`	Counter	Requests rejected before or during dispatch
`aip_connector_host_heartbeat_failures_total`	Counter	Failed heartbeat or lease-recovery cycles

Metric	Type	Interpretation
`aip_connector_host_lease_recovery_attempts_total`	Counter	Re-registration attempts after lease expiry
`aip_connector_host_lease_recoveries_total`	Counter	Successful lease recoveries

Add instance, replica, tenant, region, zone, artifact, and deployment labels in the scrape configuration rather than changing the metric names.

Alert on readiness loss, storage loss, sustained in-flight saturation, rejection growth, heartbeat failures, and recovery attempts without matching recoveries.

The reviewed Twenty host exposes no connector-specific metrics for provider status, webhook nonce count, local events, pending outbox batches, or central acknowledgements. Collect those through bounded durable-store diagnostics, structured deployment telemetry, and exact event evidence.

Correlate durable evidence

Use these identities for every investigation:

- connector type, version, instance, replica, artifact, and configuration revision;
- tenant, membership, external account, workspace, and credential revision;
- AIP Action ID, capability, canonical input hash, and idempotency-key digest;
- approval ID, decision ID, and policy hash for mutations;
- execution checkpoint, terminal status, result, error, and receipts;
- provider HTTP status and bounded request ID;
- webhook ID, nonce digest, deterministic event ID, replay scope, and event time;
- local event identity, outbox batch identity, and central delivery outcome.

Do not log provider token, webhook secret, raw webhook payload, full idempotency key, or unredacted record and metadata bodies.

The connector exposes a bounded provider error allowlist. Absence of an unlisted provider field is expected and is not proof that the provider omitted it.

Recover a provider readiness failure

1. remove the replica from new assignments;
2. retain `/ready`, metric, lease, artifact, configuration, and credential state;
3. verify fixed origin, DNS, network policy, TLS, and redirect behavior;
4. verify token file ownership, current credential revision, and workspace binding without printing the token;
5. check authenticated core OpenAPI status through the deployment's controlled path;
6. compare provider maintenance and rate-limit evidence;
7. repair the fixed dependency without changing tenant or instance identity;
8. require `/ready` to return `200` before restoring route weight.

Do not prove recovery with a mutation. Use the same bounded read or health path until routing readiness is restored.

Recover a runtime storage failure

When `/ready` reports `runtime.ready: false`:

1. stop new assignments and webhook registration changes;
2. preserve host and PostgreSQL state before restart or failover;
3. verify database endpoint, URL file, TLS, role, capacity, locks, and schema availability;
4. inspect the structured runtime recovery report;
5. preserve queued Action, idempotency, checkpoint, replay, event, and outbox identities;
6. restore readable and writable durable storage without replacing logical IDs;
7. require storage and connector readiness before reassignment.

Host registration is blocked when durable recovery reports queued-Action or delegation errors. Investigate those records instead of forcing the replica online.

Do not replace PostgreSQL with an empty database to clear a failure. That can discard idempotency, replay, event, and outbox ownership while external effects remain.

Recover an expired lease

The heartbeat loop probes connector and storage health before renewal. An expired lease is re-registered only while both are ready.

For an ambiguous control-plane failure, the host retains one stable recovery request identity and payload for idempotent replay. Admission or configuration rejection clears that pending identity because no lease committed.

If lease recovery fails:

1. keep the replica out of new assignments;
2. compare attempt, recovery, and heartbeat-failure counters;
3. verify control-plane endpoint, TLS, DID, and registry identity;
4. check for a live conflicting replica or immutable identity mismatch;
5. repair the control-plane boundary;
6. observe one successful recovery and a valid lease in `/ready`.

Do not start a second process with the same replica ID while the first may hold a valid lease.

Recover an interrupted or uncertain Action

Read the original Action, result, error, receipts, idempotency record, and provider request identity before deciding.

Evidence	Response
Read failed temporarily	Retry the same read intent according to policy
Mutation has no provider dispatch evidence	Classify the exact failure before deciding on a new intent
Completed result and expected provider state agree	Retain the result and verification read
Action is pending or approval-held	Continue the original lifecycle
Mutation reports uncertain outcome	Inspect actual provider state; do not replace the Action or key
Provider state proves intended effect	Record reconciliation evidence without replay
Provider state proves no effect after bounded review	Create a new governed intent only under recovery policy
State remains ambiguous	Escalate and preserve the mutation fence

`record.create`, `record.create_many`, and `record.update` advertise retry safety with the original key. That contract does not authorize a new Action, new key, changed body, or automatic retry after an unresolved outcome.

The other eleven mutations advertise unsafe retry. All mutation response or post-connection failures can require provider-state reconciliation.

The connector exposes no reconciliation operation. Use authorized bounded reads and external provider evidence while retaining the original AIP chain.

Recover webhook ingress

Classify the public response first:

Response	Primary boundary
`400 webhook.missing_header`	Proxy or provider omitted an exact required header
`401 webhook.authentication_failed`	HMAC, time, nonce, JSON, workspace, or payload validation
`503 webhook.storage_unavailable`	Replay, event-log, or durable-outbox state
`200 centrally_acknowledged`	Local acceptance and central acknowledgement completed
`200 durably_queued`	Local outbox owns delivery; central acknowledgement is pending

For repeated `401`, verify exact raw-body preservation, clock, secret revision, workspace, registration, and nonce reuse without exposing secret or body.

For `503`, preserve replay scope, nonce admission state, event log, profile state, database, and outbox before changing the host. Replay admission occurs before payload persistence, so the same nonce may already be fenced.

Do not ask the provider to resend the same nonce after a late local failure. Determine whether the deterministic event or durable batch exists and recover from the owned stage.

Recover central event publication

A `durably_queued` response means the exact normalized batch exists in local profile state or was already acknowledged. The background worker owns delivery.

1. search local event log by deterministic event ID;
2. locate the durable outbox batch derived from channel and event data;
3. verify event endpoint, route authorization, TLS, central ingress, and host signing identity;
4. inspect attempts, next attempt, lease owner, lease expiry, and bounded last error;
5. restore central reachability without changing event or batch identity;
6. confirm central observation of the same event ID;
7. confirm the durable batch drains according to publisher policy.

A crash after central acknowledgement but before local deletion can resend the same event ID. Central event storage owns deduplication of that identity.

Do not edit an event ID, create a new batch, or clear the outbox merely to make the queue appear empty.

Drain and restore routing

Graceful shutdown marks the replica draining, waits for in-flight work within the configured deadline, stops the listener, and attempts an offline registry transition.

Before stopping, preserve:

- nonterminal and uncertain Action identities;
- provider request and verification evidence;
- replay and local event state;
- pending durable event batches;
- lease, heartbeat, metrics, and recovery state.

After repair, require:

- static host liveness and structured readiness;
- authenticated provider health against the intended workspace;
- readable and writable PostgreSQL with clean recovery report;
- valid lease and exact replica identity;
- stable metrics during an observation window;
- reconciled uncertain Actions;
- recovered webhook events and outbox batches;
- a bounded qualification result for the affected behavior.

Restore route weight gradually. Keep the previous artifact and database recovery point available until the incident observation window closes.

Related documentation

- Deploy the Twenty connector
- Twenty connector configuration
- Twenty webhook security, replay, and events
- Errors and retry decisions
- Metrics reference