Recover the Cal.diy connector

Use this runbook when a Cal.diy host is unavailable, not ready, rejecting traffic, losing its lease, returning uncertain mutation outcomes, or retaining provider events for later delivery. It helps you identify the failing boundary and reco

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

Run these checks through the operator network. Keep action submission and provider webhook traffic on their authenticated routes. This runbook assumes the one-account deployment described in Deploy the Cal.diy connector.

Contain risk before diagnosis

Pause new Cal.diy mutations or disable the affected tenant binding when any accepted mutation has an unknown outcome, durable storage is unavailable, or the replica cannot renew its lease. Leave read traffic enabled only when its provider and tenant boundaries remain verified.

Do not kill the process, clear PostgreSQL, rotate credentials, create a new action ID, or move a pinned action as an initial response. First retain:

- UTC incident start and last known-ready time;
- connector type, version, instance, replica, and artifact digest;
- tenant, external account, configuration, and credential revisions;
- lease sequence and expiry from the lifecycle view;
- action ID, idempotency key reference, provider operation ID, and provider

request ID where present;

- `/ready` response and two `/metrics` snapshots with timestamps;
- gateway assignment, action result, events, receipts, and redacted logs;
- Cal.diy audit evidence and central event observations where applicable.

These identities determine whether recovery may resume existing work. A new healthy replica does not make an old uncertain mutation safe to repeat.

Read five views in order

Use all five views. No single view proves the outcome of an action.

The read-only examples require `curl`, `jq`, and an operator-only observation origin in `CAL_HOST_OBSERVE_ORIGIN`.

View	What it answers	What it does not prove
`/health`	Can the host process answer its static AIP liveness route?	Lease, storage, provider, or tenant readiness
`/ready`	Are drain, lease, storage, and the Cal.diy profile probe ready now?	Other capabilities or one action's outcome
`/metrics`	Is local readiness, load, rejection, or lease-recovery state changing?	Durable action or webhook-batch identity
Lifecycle and gateway	Is the replica eligible and where is accepted work pinned?	Whether Cal.diy committed an external effect
Action and provider evidence	What happened to this action at AIP and Cal.diy?	General host readiness for new work

Capture the three host surfaces without sending an action:

```
SH
curl --silent --show-error "$CAL_HOST_OBSERVE_ORIGIN/health" | jq .
curl --silent --show-error \
  --write-out '\nHTTP %{http_code}\n' \
  "$CAL_HOST_OBSERVE_ORIGIN/ready"
curl --silent --show-error "$CAL_HOST_OBSERVE_ORIGIN/metrics"
```

The `/ready` probe calls the authenticated Cal.diy profile endpoint every time. Set probe frequency and timeout within the reviewed provider and network budgets. Do not use aggressive polling as a substitute for metrics.

Interpret readiness precisely

`/health` returns success while the common process router is alive. `/ready` returns HTTP `200` only when all four predicates are true:

Readiness field	Ready value	Failure boundary
`draining`	`false`	Planned or partial shutdown
`lease_valid`	`true`	Lifecycle path, clock, registry, or expired lease
`runtime.ready`	`true`	PostgreSQL read/write readiness
`connector.ready`	`true`	Authenticated Cal.diy profile request

The `runtime.durability` field identifies the configured store class. A production PostgreSQL host should report `durable_shared`. The nested recovery summary is captured during startup:

Recovery field	Meaning
`recovered_at`	Time at which runtime recovery began
`pending_approvals`	Existing approvals retained for continuation
`recoverable_transactions`	Transactions retained for reconciliation
`queued_results`	Queued actions completed during startup recovery
`delegation_results`	Delegations completed during startup recovery
`callback_deliveries`	Callback deliveries resumed during recovery

The summary is startup evidence, not a live queue gauge. If queued-action or delegation recovery reports an error, the process fails before registration rather than advertising damaged state as ready.

Cal.diy connector readiness performs `cap:cal_diy:profile.get` against the configured account. Success proves that the selected credential reached that profile endpoint at that moment. It does not validate booking permissions, webhook delivery, every operation-specific scope, or a mutation result.

Interpret the eight host metrics

The pinned host emits exactly these unlabeled Prometheus metrics:

Metric	Operational use	Important limit
`aip_connector_host_ready`	Combined local lease, drain, storage, and connector state	Reads cached flags; correlate with `/ready` and registry state
`aip_connector_host_storage_ready`	Last durable-storage probe result	Does not identify a damaged record
`aip_connector_host_in_flight`	Actions executing in this process	Does not include provider work after a lost response
`aip_connector_host_requests_total`	Native messages observed by the host	Does not classify success by capability
`aip_connector_host_rejected_total`	Native messages that ended in a host or connector rejection	Requires error-code correlation
`aip_connector_host_heartbeat_failures_total`	Failed heartbeat renewal or lease-recovery cycles	One increase is not proof of lease expiry
`aip_connector_host_lease_recovery_attempts_total`	Re-registration attempts after lease expiry	Includes attempts that do not restore readiness
`aip_connector_host_lease_recoveries_total`	Successful expired-lease recoveries	Compare with attempts and the current lease

The source emits no Cal.diy-specific latency, provider-status, credential, dispatch-ledger, webhook-replay, or event-outbox metric. Use AIP action and event records, provider request IDs, redacted logs, and deployment-specific read-only diagnostics for those views. Do not infer a successful booking from `aip_connector_host_ready`.

Define alert duration and ratios from the deployment SLO. At minimum, alert on these relationships rather than isolated samples:

- readiness remains `0` outside a planned drain;
- storage readiness is `0` or `/ready` reports a storage failure;
- heartbeat failures grow as the current lease approaches expiry;
- lease-recovery attempts grow without a successful recovery;
- in-flight work remains at the configured concurrency bound;
- rejection growth diverges from request growth;
- provider readiness fails while process and storage health remain good;
- an accepted webhook event is absent from central observation beyond its

delivery objective.

Classify the failing boundary

Observation	Boundary	Immediate decision
No process response	Process, node, ingress, or network	Preserve the endpoint and database; follow hard-crash recovery
`/health` works and `draining` is true	Planned or interrupted shutdown	Stop new assignments and finish drain or investigate its failure
`lease_valid` is false	Lifecycle network, TLS, DID, clock, or admission	Keep mutations paused and allow fenced lease recovery after repair
`runtime.ready` is false	PostgreSQL or durable state	Isolate the replica; never replace the database with an empty store
`connector.ready` is false	Provider origin, CA, credential, account, or Cal.diy availability	Repair the provider boundary and verify a profile read
Rejections rise with high in-flight work	Local capacity	Preserve action IDs and let bounded backpressure work
Rejections rise with low load	Schema, signature, route, credential revision, approval, or connector error	Group by exact error code before changing capacity
Mutation returns `outcome_unknown`	Provider dispatch evidence	Reconcile; do not resubmit or compensate yet
Webhook says `durably_queued`	Central event delivery	Restore central ingress and observe the retained batch

Recover process or lease loss

If the process is still running, repair lifecycle DNS, TLS, CA trust, pinned DID, clock, or service availability before restarting it. The host retains its last committed lease until expiry. Once the lease has expired and local storage and provider probes are ready, the heartbeat worker performs signed re-registration automatically.

An ambiguous lifecycle response is retried with the same request identity and registration bytes. A definite admission or configuration rejection starts a new request only after the mismatch is corrected. Do not extend the lease or set registry status by hand.

After a hard crash:

1. prove that no process still owns the replica key and endpoint;
2. preserve the same PostgreSQL database, artifact, instance, and replica;
3. wait for clean offline state or lease expiry before re-registering that

replica identity;

4. start the reviewed process and let durable recovery finish;
5. stop if recovery reports queued-action or delegation errors;
6. compare the recovery summary, new lease sequence, retained assignments, and

original action IDs;

7. verify a non-mutating profile read before restoring mutation traffic.

Use a new replica identity for a real replacement. Never run two processes with the same replica ID or signing key.

Recover storage or provider readiness

When `runtime.ready` is false, pause the affected binding before the next heartbeat makes the replica ineligible. Restore connectivity and the exact database. Check capacity, certificates, database role, schema, and read/write health. Do not initialize a new database to make readiness green; that removes the state needed to fence retries and recover callbacks, actions, and events.

When `connector.ready` is false, verify the configured provider origin, CA, account ID, credential mode, credential file, and credential revision. The profile health request is read-only, but repeated failure can cause an unhealthy heartbeat and loss of routing eligibility. Repair or rotate through the reviewed credential procedure, then require `/ready`, a current registry lease, and one gateway-routed profile read.

Do not change instance or account identity to mask a provider failure.

Recover capacity and rejected traffic

At the local concurrency bound, new actions receive retryable `connector_host.capacity` with HTTP `429` and a short retry hint. Let the gateway and client honor backpressure with the original action identity. Scale only by admitting a distinct replica under the same instance and durable coordination boundary.

`aip_connector_host_rejected_total` also increases for invalid envelopes, signature or route failures, not-ready state, credential-revision denial, approval import failure, and connector execution errors. Capacity is not the right response when in-flight work is low. Correlate the signed AIP error with Errors and retry decisions.

A retryable error describes transport or scheduling policy. It does not prove that a Cal.diy mutation was not committed.

Recover a Cal.diy mutation

Cal.diy mutations use a durable, account-scoped dispatch ledger. A claim lease defaults to 120 seconds, but time alone does not authorize a retry after the provider-dispatch boundary.

Dispatch state	Meaning	Safe decision
`claimed`	The idempotency key is owned before provider dispatch	Wait while current; after expiry reconciliation can prove `not_committed`
`dispatching`	The provider boundary may have been crossed	Wait while current; after expiry treat it as `uncertain`
`completed`	A successful provider response and output are stored	Reuse the stored result for the same operation, key, and input
`not_committed`	Durable evidence says provider dispatch did not commit	The same operation, key, and input may acquire a new claim
`uncertain`	Dispatch occurred or may have occurred without terminal proof	Reconcile with provider and transaction evidence; do not retry

The connector also marks a mutation uncertain when cancellation happens after dispatch, a successful provider response fails output validation, or a transport or remote outcome is ambiguous. A definitive rejection or failure that guarantees no dispatch is recorded as `not_committed`.

Reconciliation returns the stored provider request and status evidence. A trusted durable transaction outcome cursor can establish committed or not-committed terminal state. Without that evidence, `uncertain` remains nonterminal and the same provider operation ID is returned as the cursor.

Inspect Cal.diy for the embedded AIP booking metadata or other operation-owned evidence where the capability contract provides it. Do not edit the ledger or invent a terminal result.

Recover webhook event delivery

An authenticated webhook is normalized into deterministic AIP event IDs, stored in the durable event log, and then accepted into the connector event outbox. The acknowledgement reports either `centrally_acknowledged` or `durably_queued`.

The outbox retains at most 10,000 pending batches, with 1 to 100 events per batch. A background worker claims batches for 30 seconds, scans up to 100 at a time, and retries transient central-delivery failure with bounded exponential backoff. It requires a ready replica and current lease before publication.

After restart, the worker scans the same PostgreSQL-backed profile state and publishes the exact retained event IDs. A crash after central acknowledgement but before local deletion can resend those IDs; central event storage deduplicates them. Do not ask Cal.diy to recreate an event that the host already accepted.

There is no public pending-outbox route or metric in this revision. Use the webhook acknowledgement, central event presence, retained logs, and controlled read-only storage diagnostics. Preserve webhook replay state with the outbox; deleting either state can turn recovery into duplicate ingestion. See Webhook security and replay for the signature, replay, and retention boundary.

Verify recovery

Restore mutation traffic only when every applicable check passes:

1. `/health` and `/ready` succeed and all four readiness predicates agree;
2. the lifecycle view shows the intended replica with a current lease;
3. metric growth has stabilized and available capacity is positive;
4. the same action ID reaches its justified durable state;
5. an uncertain mutation has terminal provider evidence or remains paused;
6. central AIP observes the exact accepted webhook event IDs;
7. a gateway-routed profile read reaches the intended Cal.diy account;
8. the incident record retains every identity, transition, error, and operator

decision.

If the views disagree, keep the mutation boundary paused. Escalate with the database retained and include the source revision, artifact and manifest digests, lease history, readiness body, metric snapshots, action and provider identifiers, event IDs, and redacted logs.

Related documentation

- Deploy the Cal.diy connector
- Operate the connector host lifecycle
- Observe and recover AIP
- Metrics reference
- Errors and retry decisions