CrewAI health, observability, and recovery

Use this runbook when a CrewAI replica is unhealthy, not ready, losing its lease, or holding an Action without trustworthy terminal evidence. Preserve the host runtime and sidecar journal before changing the deployment.

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

Four states matter independently: process liveness, fleet routing readiness, sidecar durability, and provider outcome. A green probe at one boundary does not prove the others.

Start with the four boundaries

Boundary	Primary evidence	Healthy meaning	Excluded meaning
Rust process	Host `GET /health`	Router responds with AIP host identity	Storage, sidecar, or lease readiness
Fleet replica	Host `GET /ready`	Not draining, valid lease, storage ready, connector ready	Successful CrewAI model or tool call
Python sidecar	Authenticated sidecar `GET /health`	Runtime loaded, journal owned, registry and policy available	Provider credentials and external tools work
Action outcome	Host checkpoints, sidecar record and events, provider evidence	Evidence agrees on a terminal result	Automatic safety of a replacement Action

Check the boundaries in that order. A failed liveness probe makes later local probes unavailable, but it does not erase durable evidence.

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. If it fails, inspect process termination, listener binding, ingress routing, resource limits, and container state.

Do not restart repeatedly before preserving PostgreSQL checkpoints and the sidecar volume. An interrupted Action can retain external effects even when the host is gone.

Read routing readiness

The host `/ready` endpoint returns HTTP `200` only when all four 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 CrewAI 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": "CrewAI sidecar is durable and exposes configured crews"
  }
}
```

The exact connector detail includes the configured crew count. Use the fields, not the example string, as the stable diagnostic contract.

Remove a not-ready replica from new assignments. Preserve existing Action identity until its runtime and sidecar state are reconciled.

Read sidecar health

The Rust connector authenticates to sidecar `/health` and requires:

- `status` equal to `ready`;
- `durable_journal` equal to `true`;
- every configured host crew ID in `crews`;
- every descriptor-admitted operation in `operations`.

The sidecar response also reports:

Field	Operational use
`active_runs`	Current records still marked running
`retained_runs`	Journal occupancy
`max_retained_runs`	Admission capacity ceiling
`stream_checkpoint_events`	Persistence interval for streamed chunks
`training_directory`	Whether protected training storage is configured

The host rejects a sidecar with a narrower operation policy than its admitted descriptors. Extra sidecar operations do not become AIP capabilities unless a descriptor admits them.

Sidecar health does not call a model, tool, knowledge store, evaluator, or crew task. Use a bounded qualification Action for those claims.

Monitor the 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
`aip_connector_host_lease_recovery_attempts_total`	Counter	Re-registration attempts after lease expiry
`aip_connector_host_lease_recoveries_total`	Counter	Successful lease recoveries

Scrape each replica with deployment labels added outside the process. Alert on readiness loss, storage loss, sustained in-flight saturation, rejection growth, heartbeat failures, and recovery attempts without matching recoveries.

The sidecar has no `/metrics` route at the reviewed revision. Collect its health fields, process resources, structured logs, journal occupancy, and Action-event evidence separately.

Correlate logs and durable evidence

Use these identifiers on every query:

- tenant, connector type, version, instance, and replica;
- host artifact and sidecar image digests;
- AIP Action ID and idempotency key digest;
- crew ID and operation;
- sidecar event sequence and host checkpoint;
- provider, model, tool, and artifact request identifiers when available.

The sidecar logs admission, start, completion, cancellation, archive, and failure with Action ID, crew ID, and operation. Failure logging includes a provider exception traceback, which may contain governed data.

Restrict sidecar logs and redact them before sharing. The durable failed event contains only the exception class and a generic diagnostic message.

Recover a sidecar readiness failure

1. stop new assignments to the host replica;
2. read host `/ready` and authenticated sidecar `/health` once;
3. record image digests, configuration revision, crew descriptors, and enabled

operations;

4. preserve `runs.json`, its lock context, CrewAI home, and training artifacts;
5. compare sidecar crews and operations with the host descriptor set;
6. repair the missing registry, policy, secret, path, permission, or capacity

input;

7. restart only one sidecar against the preserved volume;
8. reconcile every restored uncertain record;
9. verify sidecar health, host readiness, and registry lease before routing.

Do not bypass the journal lock. A second writer can invalidate the durable request fence even if both processes appear healthy.

Recover a storage failure

When host readiness reports runtime storage failure:

1. remove the replica from new routing;
2. keep the host and sidecar evidence available for inspection;
3. verify PostgreSQL endpoint, credential file, TLS, permissions, capacity, and

schema availability;

4. restore readable and writable durable storage without replacing Action

identities;

5. confirm the runtime recovery report has no queued or delegation errors;
6. require `/ready` to return `200` before reassignment.

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

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 the exact 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 TLS, DID, endpoint, 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 create a second process with the same replica ID while the first may still hold a valid lease.

Recover an interrupted Action

A restored sidecar record that was `running` becomes terminal `failed` with `uncertain_outcome: true`. The task is not resumed.

Use this decision order:

Evidence	Action
Host and sidecar both show completed	Verify output and provider effects, then retain the result
Sidecar shows running and process is healthy	Continue bounded status or event observation
Restart failure reports uncertainty	Reconcile provider, tool, model, and artifact effects
Sidecar reports cancelled for thread-backed work	Assume provider work may continue
Journal record is absent but host shows dispatch	Escalate as lost sidecar identity; do not recreate blindly
Exact repeat returns a collision	Preserve the original record and correct changed request material

Never archive a record to make a retry fit. Archive only a terminal record after the replay-retention window and an explicit operator decision.

Drain and restore routing

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

After repair, require:

- host liveness and readiness;
- authenticated durable sidecar health;
- a valid registry lease and exact replica identity;
- stable storage and heartbeat metrics;
- reconciled non-terminal and uncertain Actions;
- a bounded qualification result for affected provider behavior.

Restore route weight gradually. Keep the previous artifact and state available until the observation window closes.

Related documentation

- Deploy the CrewAI connector
- CrewAI connector configuration
- CrewAI idempotency, cancellation, and process fencing
- Errors and retry decisions
- Metrics reference