Operate and recover the Dify connector

Use this runbook to locate a Dify connector failure before changing state. Inspect process, readiness, lease, storage, credential, Action, and provider evidence in that order.

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

When a mutation has an uncertain provider outcome, stop related mutations. Reconcile durable Action and provider state before any retry or corrective action.

Read signals in order

Order	Surface	Primary fact
1	`GET /health`	The host HTTP process responds
2	`GET /ready`	This replica can accept routed Actions now
3	`GET /metrics`	Pressure, rejection, or lease recovery is changing
4	Connector registry	Version, route, replica, and lease are authoritative
5	Durable Action and profile state	Work and remote task identity survive locally
6	Dify provider reads	The configured app or workspace state exists
7	Stream or cancellation evidence	Completion or remote-stop certainty is known

Do not use a later signal to overwrite an earlier fact. A live process can be not ready, and a ready replica cannot determine an older mutation outcome.

Treat liveness as process-only

`GET /health` returns HTTP `200` while the host router can serve:

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

This route does not probe PostgreSQL, Dify, credentials, admission, lease, task correlation, or stream publication.

Read composite readiness

`GET /ready` runs a bounded storage check and live connector health probe. It also reads the drain flag and current lease expiry.

HTTP `200` requires:

- the replica is not draining;
- the lifecycle lease is valid;

- durable runtime storage is readable and writable;
- every configured Dify credential passes its provider probe.

Any false condition returns HTTP `503`. Inspect `draining`, `lease_valid`, `runtime.ready`, and `connector.ready` independently.

The `runtime.recovery` object describes bounded startup recovery. It is not a live task-store, approval, callback, or provider-work backlog.

Interpret Dify product health

The connector probes every configured credential:

Credential	Provider request
App	`GET /v1/parameters`
Knowledge	`GET /v1/datasets?limit=1`

Checks run with concurrency bounded at 32. One failed credential makes the whole connector health probe fail.

The common host bounds the complete probe by its configured health timeout, which defaults to 2,000 milliseconds.

A ready result proves route reachability and bearer-key acceptance for all configured descriptors at that moment. It does not verify app mode, mutation permissions, streaming, cancellation, indexing, or pipeline execution.

The healthy detail reports app and Knowledge credential counts. A failure does not expose a structured per-credential status list.

Use host metrics

`GET /metrics` renders eight Prometheus families:

Metric	Type	Operational meaning
`aip_connector_host_ready`	Gauge	Cached provider and storage flags plus live lease and drain state
`aip_connector_host_storage_ready`	Gauge	Last stored durable-storage probe result
`aip_connector_host_in_flight`	Gauge	Actions executing in this process
`aip_connector_host_requests_total`	Counter	Native message requests received
`aip_connector_host_rejected_total`	Counter	Instrumented host rejections
`aip_connector_host_heartbeat_failures_total`	Counter	Failed lease renewal or recovery attempts
`aip_connector_host_lease_recovery_attempts_total`	Counter	Re-registration attempts after expiry
`aip_connector_host_lease_recoveries_total`	Counter	Re-registrations that applied a new lease

The metrics have no application labels and reset on process restart. Add bounded scrape-target metadata for replica, version, zone, and connector type.

The ready gauge does not run a provider probe. `/ready` and the heartbeat loop update cached flags.

Account for observability gaps

The reviewed host has no public Dify-specific metric for:

- provider route, status, or latency distribution;

- individual credential health;
- response-bound or invalid-stream failures;
- emitted SSE events or callback backlog;
- active durable task mappings;
- cancellation attempts or remote-stop confirmations;
- Dify indexing, workflow, model, or pipeline state.

Some successful operation envelopes include Dify `x-request-id`. Typed connector failures currently leave `provider_request_id` empty.

Retain the AIP Action ID and any output request ID before redaction. Do not infer an empty task store or healthy provider route from a normal scrape.

Understand lease behavior

The heartbeat loop probes Dify and durable storage, records readiness flags, reports in-flight work, and renews or recovers the replica lease.

An unhealthy heartbeat moves the registry replica offline. A later healthy cycle can restore it under a valid lease or re-register after expiry.

Lease recovery preserves one request ID across ambiguous control-plane failures. Definitive admission or configuration rejection clears that pending request before a corrected attempt.

Rising heartbeat failures without recovery indicate control-plane, identity, sequence, admission, storage, provider, or network failure. They do not isolate the Dify boundary by themselves.

Interpret Dify failure codes

Code	Category	Retry rule	Uncertain mutation condition
`connector.dify.authentication`	Auth	Never	No
`connector.dify.remote_temporary`	Temporary	Safe reads only	Invocation mutation
`connector.dify.remote_rejected`	Permanent	Never	No
`connector.dify.transport`	Transport	Safe reads only	Mutation after connection began
`connector.dify.invalid_stream`	Connector	Safe reads only	Invocation mutation
`connector.dify.response_too_large`	Permanent	Never	Invocation mutation
`connector.dify.cancelled`	Temporary	Never	Mutation without confirmed remote stop
`connector.dify.task_store`	Temporary	Never	Invocation mutation
`connector.dify.invalid_credential`	Auth	Never	No
`connector.dify.configuration_or_input`	Permanent	Never	No

Temporary classification does not make a mutation replay-safe. The catalogue's read safety controls the `retryable` field for temporary, transport, and invalid-stream failures.

Contain and capture evidence

Drain or remove the route when identity, storage, provider origin, credential, manifest, or task-state ownership is uncertain.

Retain:

- host, version, instance, replica, tenant, and external-account identity;
- timestamped `/health`, `/ready`, and `/metrics` observations;
- registry route, status, lease sequence, and expiry;
- configuration and credential revisions without secret bytes;
- Action ID, capability, input hash, approval, and idempotency-key hash;
- Dify status, event name, task ID, and request ID when available;
- final provider reads used for reconciliation.

Never retain API keys, signing seeds, database URLs, or unrestricted customer payloads in incident records.

Classify the failing boundary

Observation	Likely boundary	Next evidence
`/health` unavailable	Process, listener, edge, or network	Supervisor and bounded startup error
`/health` is `200`, `/ready` is `503`	Drain, lease, storage, or Dify health	Individual readiness fields
`runtime.ready` is false	PostgreSQL, schema, role, or timeout	Durable storage probe and migration state
`connector.ready` is false	Origin, TLS, credentials, or Dify	Descriptor set and bounded health error
`lease_valid` is false	Heartbeat, control plane, sequence, or admission	Registry and recovery counters
In-flight equals capacity	Local saturation	Action duration and route pressure
Rejections increase	Host validation, trust, readiness, approval, or capacity	Signed protocol error code
`invalid_stream` appears	Provider SSE, bounds, or callback emission	Last valid event and cleanup certainty
`task_store` appears	Profile-state durability	Storage health and action mapping ownership
Cancellation is uncertain	Missing task identity or failed stop	Durable mapping and provider task state

Recover storage and task state

1. Keep the replica out of routing.
2. Verify the configured database URL file, role, schema, and reachability.
3. Preserve durable Actions, idempotency records, and profile state.
4. Restore one coherent storage authority for all instance replicas.
5. Inspect bounded startup recovery and task-store access.
6. Require `runtime.ready: true` before registration or routing.
7. Reconcile every mutation whose dispatch boundary is unknown.

Do not delete task mappings or idempotency records to force readiness.

Recover provider health or credentials

1. Confirm the fixed Dify origin and descriptor hashes.

2. Identify the failing credential revision without printing its key.
3. Distinguish TLS, DNS, timeout, `401`, `403`, rate limit, and server failure.
4. Rotate the owner-only key file through a replacement replica when required.
5. Re-run readiness and the matching read-only canary.
6. Reconcile earlier mutations before enabling writes.

Product keys and descriptors are startup-owned. Replacing file bytes does not prove that a running process adopted them.

Recover lease or control-plane state

1. Compare local instance, replica, version, manifest, artifact, and sequence

with the registry.

2. Verify the fixed control endpoint, trusted DID, TLS, and registry state.
3. Preserve a pending recovery request across ambiguous transport failure.
4. Correct definitive admission or configuration rejection first.
5. Require a valid ready lease and correct route before traffic resumes.

Never start a second live process with the same replica ID while lease or pinned Action ownership remains unresolved.

Recover capacity pressure

`connector_host.capacity` is returned before provider dispatch and is marked retryable by the host. Confirm durable Action state before repeating the same correlated request.

Reduce pressure or add an admitted replica before raising the in-flight limit. A larger limit does not increase Dify capacity or preserve latency.

Recover stream failure

Preserve the last valid event, task mapping, response-byte count, and cleanup result. A malformed or oversized stream can occur after Dify admitted a task.

If a durable task ID exists, verify the mode-specific stop result. If no task ID was observed, retain an uncertain outcome and reconcile provider state.

Do not convert premature close into completion or replay the model execution under a new Action ID.

Recover an uncertain mutation

1. Preserve the original Action ID, key, approval, and canonical input.
2. Read durable Action and task-correlation state.
3. Query the authoritative provider resource when an operation exposes one.
4. Correlate Dify request, task, workflow-run, document, or batch identity.
5. Accept matching state, retain ambiguity, or issue a separately approved

corrective intent.

An idempotency key fences AIP replay identity. It does not prove provider-side deduplication or absence of effects.

Verify recovery

Require:

- `/health` is reachable;
- `/ready` returns `200` with valid lease, no drain, and both probes ready;
- registry identity and route match the intended replica;
- ready and storage gauges agree with the latest readiness probe;
- recovery counters stop increasing unexpectedly;
- exact manifest counts and read-only canaries pass;
- task-store access and streaming callback ownership are intact;
- every uncertain mutation has an explicit reconciliation decision;
- no durable state or secret was deleted to obtain recovery.

Observe at least one lease renewal and a normal provider window. These checks establish operational recovery, not complete capability qualification.

Related documentation

- Deploy the Dify connector
- Dify connector configuration
- Dify streaming, cancellation, and binary responses
- Observe and recover
- Metrics reference