Troubleshoot the Chatwoot connector

Use this runbook to identify the boundary that failed and choose the narrowest safe next action. It covers process startup, connector-host readiness, signed Action handling, Chatwoot provider calls, webhook ingress, and durable Event delive

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/chatwoot/troubleshooting/README.md

Pause new mutations whenever provider outcome is uncertain. Preserve the original Action ID, idempotency key, account ID, route assignment, remote status, webhook delivery ID, and durable state before changing a deployment.

Identify the error surface

Start with the surface that produced the evidence. The same HTTP status can represent different trust and recovery boundaries.

Surface	Evidence shape	First trust check
Process startup	Error text and non-zero exit	Compare configuration, secret-file ownership, storage, and admission
Pre-envelope host rejection	Unsigned compact JSON	Trust only the controlled TLS path; no valid envelope exists to sign
Connector-host response	Signed AIP error envelope	Verify signature, sender, recipient, and correlation
Durable Action	`ActionResult`, state, and events	Read it even when the original HTTP exchange failed
Chatwoot webhook	Compact `{ "error": { "code": "..." } }`	Correlate the authenticated ingress request and delivery ID
Event delivery	Local Event and outbox state	Separate accepted ingress from central publication

Automate on the exact code, category, retry flag, `uncertain_outcome`, and durable state. Message text is diagnostic and is not a stable interface.

Choose the first check

Symptom	First check	Safe first action
Capability is absent	Authenticated tenant catalogue and enabled binding	Correct discovery or admission; do not call a host directly
Host returns `401`	Host trust code versus `connector.chatwoot.authentication`	Repair only the failed authentication boundary
Host returns `403`	Route, credential revision, approval, or provider code	Re-resolve the exact authority boundary; do not bypass it
`/health` works and `/ready` fails	Drain, lease, storage, and connector fields	Follow the failing readiness owner
Read returns a provider or transport failure	Published retry support and Action budget	Retry only the same eligible read through policy
Mutation has no definitive provider result	`uncertain_outcome` and existing provider state	Reconcile before another mutation
Same idempotency key is rejected	Existing key record and input hash	Reuse, wait, reconcile, or stop according to its state
Webhook returns `400` or `401`	Exact compact code and retained raw request	Correct that delivery without generating a replacement event
Webhook returns `503`	Replay, Event-log, and outbox storage	Restore storage, then redeliver the same request
Webhook succeeds but central Event is absent	Accepted local Event and outbox batch	Recover publication; do not replay the provider action

Resolve startup failures

Startup failures occur before a protocol error can be returned. Compare the first process error with the configured base URL, account ID, token file, response limit, allowed-operation set, webhook secret, PostgreSQL state, central Event endpoint, manifest, and immutable host identity.

Correct one owning boundary at a time. Require `/ready` and route a safe account read through `aipd` after a repair. A direct Chatwoot request does not prove the AIP tenant, route, approval, or signing boundary.

Follow readiness to its owner

`/health` is a process-liveness response. `/ready` succeeds only while the host is not draining, its lease is valid, durable storage is ready, and the authenticated Chatwoot account probe succeeds.

Failing field or observation	Owner	Decision
`draining`	Host lifecycle	Let in-flight work settle and replace or restore the replica
`lease_valid`	Registry heartbeat and fencing	Recover the same admitted identity or replace it through admission
`storage_ready`	Host PostgreSQL state	Restore durable state before accepting Actions or webhooks
`connector_ready`	Chatwoot account probe	Correct account, token, base URL, or provider availability
Ready host is not selected	Binding, route, or central registry	Re-resolve the admitted route; do not edit the assignment
`connector_host.capacity`	Replica-local concurrency	Honor bounded backoff through the original Action policy

Use the operations runbook for containment, heartbeat recovery, replacement, and rollback. Readiness does not prove every Chatwoot operation or webhook delivery path.

Look up Chatwoot connector codes

The table covers every explicit `connector.chatwoot.*` code at the reviewed revision. All connector failures set automatic `retryable` to false.

Code	Meaning	Decision
`connector.chatwoot.authentication`	Chatwoot returned `401` or `403`	Correct account, token, scope, or provider access
`connector.chatwoot.configuration`	URL, client, metadata, or connector configuration is invalid	Correct the named configuration boundary
`connector.chatwoot.invalid_action`	Capability, operation, input, or composite rule is invalid	Correct the Action; do not retry unchanged
`connector.chatwoot.invalid_credential`	Token material is empty or structurally invalid	Rotate the owner-only token file and replace the process
`connector.chatwoot.invalid_webhook`	Webhook signature, timestamp, body, or mapping input is invalid	Preserve the raw delivery and correct authentication or format
`connector.chatwoot.invalid_webhook_delivery`	Typed ingestion payload cannot be decoded	Correct the host-to-connector delivery object
`connector.chatwoot.missing_idempotency_key`	A catalogue mutation has no key	Reissue only as the same reviewed Action with a stable key
`connector.chatwoot.remote_rejected`	Chatwoot returned another non-success status	Correct request or provider state; inspect mutation outcome
`connector.chatwoot.remote_temporary`	Chatwoot returned `429` or `5xx`	An invocation can be uncertain; do not infer retry safety
`connector.chatwoot.transport`	Provider transport failed	An invocation can be uncertain; reconcile before replay
`connector.chatwoot.response_too_large`	Provider response exceeded the configured bound	Review the operation and limit; an invocation can be uncertain
`connector.chatwoot.replay_store`	Webhook replay storage failed	Restore storage and retain the original delivery
`connector.chatwoot.webhook_replay`	The typed replay policy rejected a delivery	Inspect the retained delivery and replay state
`connector.chatwoot.cancelled`	Invocation was cancelled before a result settled	Treat provider outcome as uncertain and read durable state

`connector.chatwoot.webhook_replay` exists in the typed error mapping, but the ordinary duplicate webhook path does not construct it. That path reproduces a deterministic Event ID so durable Event storage can deduplicate the delivery.

Look up connector-host codes

The host can reject a request before Chatwoot is called. Product-neutral trust, route, credential-revision, approval, replay, and execution codes are defined in the global error reference. These compact or immediate host codes often identify the first diagnostic branch.

Code	HTTP	Trust and recovery decision
`schema.envelope.invalid`	400	Correct the JSON against the Envelope schema; the compact response is unsigned
`envelope.decode.invalid`	400	Correct typed Envelope decoding; the compact response is unsigned
`connector_host.not_ready`	503	Wait for a ready, leased, non-draining host through normal routing
`connector_host.capacity`	429	Honor the 100 ms hint only within Action retry policy
`connector_host.response_signing`	500	Isolate the response and read central durable state; authenticity is incomplete

A signed host code such as `connector_host.authentication`, `connector_host.route_mismatch`, or `connector_host.credential_revision_denied` identifies a host policy boundary, not a Chatwoot credential result. Keep the original signed envelope when escalating.

Look up webhook HTTP codes

The standalone ingress route returns a compact error rather than an AIP error envelope.

Code	HTTP	Decision
`webhook.missing_header`	400	Restore the delivery, timestamp, and signature headers
`webhook.invalid_timestamp`	400	Supply a canonical signed decimal timestamp
`webhook.invalid_utf8`	400	Preserve exact bytes and correct the sender or proxy encoding
`webhook.authentication_failed`	401	Verify secret selection, timestamp window, HMAC, raw body, and JSON delivery
`webhook.invalid_delivery`	400	Correct the typed delivery after preserving the original evidence
`webhook.invalid_mapping`	500	Retain the accepted payload and escalate the impossible Event mapping
`webhook.storage_unavailable`	503	Restore replay, Event-log, or outbox storage and redeliver the same request

An HTTP `200` can report a locally accepted Event whose central publication is still queued. Recover the durable outbox instead of asking Chatwoot to create a new webhook or repeating the Action that caused it.

Decide whether to retry

Apply these checks in order and stop at the first unmet condition:

1. verify the signed response or read the durable Action result;
2. classify the operation as a read, mutation, or webhook delivery;
3. for a mutation, establish provider outcome from authoritative state;
4. require published implementation retry support;
5. require `retryable` not to be false;
6. keep the same Action, input, and idempotency key;
7. respect attempt, elapsed-time, and backoff limits.

At this revision, the conversation-status composite and the 62 catalogue reads publish retry support. Connector-mapped failures still set `retryable` to false. The host capacity and not-ready codes can carry `retryable: true`, but that hint cannot override operation safety or the Action budget.

Treat uncertain outcomes as nonterminal

`remote_temporary`, `transport`, `response_too_large`, and `cancelled` can mark an invocation outcome uncertain. Stop new mutations for the affected resource, retain the original identifiers, read Chatwoot and AIP state, and decide whether the intended effect is present before any compensation or replacement Action.

The connector does not publish transaction or reconciliation support. Recovery therefore uses the canonical provider resource reads and the original durable Action evidence. A temporary category or missing provider request ID is never proof that a mutation did not happen.

Collect minimum evidence

Retain a bounded incident bundle before changing configuration or state:

- UTC interval, deployment revision, instance ID, replica ID, and account ID;

- Action ID, correlation ID, capability ID, route assignment, and key reference;
- exact code, category, retry flag, uncertainty flag, and remote status;
- readiness snapshot and the eight process-local host metrics;
- redacted provider resource state for an uncertain mutation;
- webhook delivery ID, timestamp, response code, Event ID, and outbox state;
- the first startup or heartbeat failure without credentials or raw payloads.

Route configuration defects to the connector owner and lease or capacity defects to the fleet operator. Route provider-access defects to the Chatwoot account owner, webhook authentication defects to the ingress owner, and uncertain mutations to the Action owner. Escalate cross-tenant output, signature failures, route misbinding, credential exposure, and impossible Event mappings as security or integrity incidents.

Verify recovery

Close the incident only when the corrected path is proven through the normal gateway route, readiness and lease agree, durable Action or Event state is terminal, and no uncertain mutation remains unresolved. Record the evidence and remove temporary containment explicitly.

Related documentation

- Health, observability, and recovery
- Configuration
- Mutation approval and idempotency
- Webhook security, loop prevention, and replay
- Errors and retry decisions