Troubleshoot Cal.diy

Use this runbook to trace a Cal.diy failure to the signed host, pinned route, credential, provider request, durable mutation, or webhook boundary. The result should be one explicit decision: correct, wait, retry a read within policy, reconc

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/cal-diy/troubleshooting/README.md

Pause new mutations whenever provider outcome is uncertain. Preserve the original action ID, idempotency key reference, route assignment, provider operation ID, provider request ID, and durable state before changing the deployment.

Trust the right error shape

Determine where the failure appeared before interpreting its fields:

Surface	Evidence shape	Trust check
Connector-host AIP route	Signed error envelope containing `ProtocolError`	Verify host signature, sender, recipient, and request correlation
Action lifecycle	`ActionResult` with `status: failed` and an error	Read the durable action even when HTTP succeeded
Cal.diy webhook ingress	Compact `{ "error": { "code": "..." } }`	Use authenticated TLS and host evidence; this is not a signed AIP envelope
Process startup	Error text and non-zero exit	Compare exact configuration and file ownership; no protocol code exists yet

Automate on `code`, category, `retryable`, `uncertain_outcome`, and durable state. Do not match message text or trust an unsigned pre-envelope response as authenticated action state.

Choose the first check

Symptom	First check	Safe first action
Capability is absent	Authenticated tenant catalog and enabled binding	Correct discovery or admission; do not call a host directly
Host returns `401`	`connector_host.authentication` versus `connector.cal_diy.authentication`	Repair the identified trust layer without broadening authority
Host returns `403`	Signed route, credential-revision, approval, or provider code	Re-resolve the exact route or policy; do not bypass it
`/health` works and `/ready` fails	`draining`, `lease_valid`, runtime, and connector fields	Follow the failing readiness boundary
Read receives `429`, `5xx`, or transport failure	Capability retry support, retry hint, and budget	Retry only the same read through normal policy
Mutation receives any ambiguous response	`uncertain_outcome` and provider operation	Reconcile before another mutation or compensation
Same idempotency key is rejected	In-flight, collision, completed, or uncertain ledger state	Wait, reuse, correct, or reconcile as the state requires
Booking plan says slot unavailable	Requested start and fresh slot evidence	Choose another slot and create a new plan
Webhook receives `400`, `401`, or `503`	Compact webhook code and retained raw request evidence	Correct delivery or retry the exact accepted-safe delivery

Diagnose authentication in order

Several layers can return an authentication-looking error. Start at the first failed boundary:

1. `connector_host.authentication` means the request did not match the pinned

gateway principal, DID, signature, or trust domain. Correct the signed gateway path.

2. `connector_host.route_mismatch` means the signed assignment does not match

this type, version, instance, replica, endpoint, manifest, tenant, binding, or fence. Discard it and resolve a current route.

3. `connector_host.credential_revision_denied` or

`connector_host.credential_revision_unavailable` means the pinned revision is denied or cannot be read. Restore the reviewed policy; do not select a different credential silently.

4. `connector.cal_diy.invalid_credential` means local credential material is

empty or structurally invalid before a useful provider exchange.

5. `connector.cal_diy.authentication` means Cal.diy returned HTTP `401` or

`403`. Verify the intended account, credential mode, scope, expiry, and provider-side status.

The standalone `aip-host-cal-diy` pins one account and one deployment credential. The optional library credential-routing mode adds `connector.cal_diy.credential_resolution`. In that mode, inspect the verified actor, tenant membership, tenant-to-account binding, credential issuer, tenant, operation scope, expiry, resolved material, and account match in that order. Do not look for library routing configuration in a standalone host.

After any credential repair, require `/ready` and route a non-mutating `cap:cal_diy:profile.get` through `aipd`. A successful direct provider request does not prove the AIP tenant and route boundary.

Separate discovery from account routing

An absent capability and a wrong Cal.diy account are different failures:

Observation	Owner	Decision
Tenant catalog omits the capability	Registry version, implementation support, or binding	Correct signed admission or use a published capability
Catalog contains it but no ready replica exists	Lease, health, topology, or capacity	Restore an admitted replica; keep the action at the gateway
Host reports route mismatch	Pinned assignment versus host identity	Re-resolve; never edit the signed route
Provider result belongs to an unexpected account	Instance, external account, or credential ownership	Stop traffic and treat it as an isolation incident
Optional library route has no account binding	Verified tenant mapping	Add the reviewed mapping; do not fall back to a default account

The standalone host does not dynamically choose among accounts. Changing its account ID or provider credential is a deployment change, not request recovery.

Interpret provider responses by operation type

The client retains a redacted provider code, HTTP status, request ID, and parsed `Retry-After`. It does not expose the raw provider error body.

Provider observation	Connector code	Read decision	Mutation decision
HTTP `401` or `403`	`connector.cal_diy.authentication`	Correct credential or scope	Definitive rejection; correct authority before a new reviewed attempt
HTTP `409` or `412`	`connector.cal_diy.state_conflict`	Refresh resource state	Definitive rejection; re-plan from current state
HTTP `429`	`connector.cal_diy.rate_limited`	Honor bounded policy and `Retry-After`	Ledger records not committed; do not auto-retry because mutation retry support is false
HTTP `500`–`599`	`connector.cal_diy.remote_temporary`	Bounded read retry may be eligible	Outcome is uncertain; reconcile
Transport failure	`connector.cal_diy.transport`	Bounded read retry may be eligible	Outcome is uncertain; reconcile
Response exceeds configured bound	`connector.cal_diy.response_too_large`	Review the operation and safe limit	Outcome is uncertain after dispatch; reconcile
Successful response is invalid JSON or schema	`connector.cal_diy.invalid_request` or `connector.cal_diy.invalid_provider_output`	Stop and inspect upstream compatibility	Outcome is uncertain for a mutation; reconcile
Other HTTP `400`–`499`, except `408`	`connector.cal_diy.remote_rejected`	Correct request or state	Definitive rejection; use a new plan when corrected
HTTP `408` or a `2xx` body with `status: error`	`connector.cal_diy.remote_rejected`	Inspect provider evidence before retry	Source treats the dispatch as uncertain; reconcile

For a mutation, `retryable: true` on rate limiting does not override the connector's published implementation support, which enables automatic retry only for reads. A definitive rejection can establish `not_committed`, but a new attempt still requires the same operation, idempotency key, and input plus the owning action or transaction policy.

Look up Cal.diy connector codes

The following groups cover every explicit `connector.cal_diy.*` code at the pinned revision. The generic `connector.cal_diy` fallback represents an unexpected non-typed connector error.

Input, policy, and planning

Code	Category and retry	Decision
`connector.cal_diy.invalid_action`	Permanent, false	Correct unsupported capability, schema, path field, or connector-owned webhook rule
`connector.cal_diy.invalid_request`	Permanent, false	Correct local URL or input construction, or investigate invalid provider JSON
`connector.cal_diy.policy`	Policy, false	Supply required idempotency or an approved webhook secret reference
`connector.cal_diy.transaction_required`	Policy, false	Use plan and commit for booking create, reschedule, or cancel
`connector.cal_diy.slot_unavailable`	Policy, false	Refresh slots and plan another start time
`connector.cal_diy.idempotency_collision`	Policy, false	Stop; the key owns a different operation or input

Credential and provider

Code	Category and retry	Decision
`connector.cal_diy.credential_resolution`	Auth, false	Repair the optional verified tenant and credential-handle chain
`connector.cal_diy.invalid_credential`	Auth, false	Replace invalid local material through credential rotation
`connector.cal_diy.authentication`	Auth, false	Correct provider credential, scope, account, or provider access
`connector.cal_diy.state_conflict`	Policy, false	Refresh provider state and re-plan
`connector.cal_diy.rate_limited`	Temporary, conditional	Reads may honor `Retry-After`; mutations require ledger-state review
`connector.cal_diy.remote_temporary`	Temporary, conditional	Retry a read within policy; reconcile a mutation
`connector.cal_diy.remote_rejected`	Permanent, false	Use status and mutation certainty to correct or reconcile
`connector.cal_diy.transport`	Transport, conditional	Retry a read within policy; reconcile a mutation
`connector.cal_diy.response_too_large`	Transport, false	Review the response contract and deployment limit
`connector.cal_diy.invalid_provider_output`	Connector, false	Retain status and request ID; inspect upstream drift
`connector.cal_diy.client`	Connector, false	Diagnose the wrapped client boundary and retain redacted evidence

Durable mutation and reconciliation

Code	Category and retry	Decision
`connector.cal_diy.in_flight`	Temporary, true	Wait for the current owner; do not create another action
`connector.cal_diy.outcome_unknown`	Temporary, false	Reconcile the returned provider operation ID
`connector.cal_diy.cancelled`	Temporary, false	Check `uncertain_outcome`; cancellation after dispatch requires reconciliation
`connector.cal_diy.state`	Temporary, false	Restore PostgreSQL and inspect the existing record
`connector.cal_diy.pre_dispatch_state`	Temporary, false, 120-second hint	Dispatch did not begin, but claim cleanup failed; wait and reconcile before reuse
`connector.cal_diy.post_dispatch_state`	Temporary, false, uncertain	Settlement failed after the dispatch boundary; use checkpoint and provider evidence
`connector.cal_diy.reconciliation_not_found`	Permanent, false	Verify provider operation ID, account scope, retained database, and route

Webhook and fallback

Code	Category and retry	Decision
`connector.cal_diy.webhook_auth`	Auth, false	Correct subscription, secret reference, and exact raw-body signature
`connector.cal_diy.webhook_policy`	Policy, false	Correct version, timestamp, or replay condition
`connector.cal_diy`	Connector, false	Retain the typed source and escalate an unexpected connector failure

Category and retry describe the emitted error. The mutation ledger remains the authority for replay safety.

Decide whether to retry

Apply this order. Stop at the first unmet condition:

1. Verify the signed response or read the durable action result.
2. Determine whether the operation is a read or mutation.
3. For a mutation, read its provider operation and reconciliation state. If the

state is not terminal `not_committed`, do not retry.

4. Require capability and implementation retry support. At this revision,

Cal.diy publishes retry implementation support only for reads.

5. Require `retryable` not to be false and respect the action's attempt,

elapsed-time, and backoff budgets.

6. Let the runtime retry the same read action according to policy; do not

bypass the gateway with a provider call.

`connector.cal_diy.in_flight` means another attempt owns the key, not that a second call is needed. `connector.cal_diy.idempotency_collision` means the key was reused with different input and must not be repaired by changing stored state.

Resolve an unknown mutation outcome

When `uncertain_outcome` is true or the code is `connector.cal_diy.outcome_unknown` or `connector.cal_diy.post_dispatch_state`:

1. freeze new mutations for the affected resource;
2. retain the original action, transaction, idempotency, provider operation,

request, and route identifiers;

3. read the existing action and transaction state;
4. invoke the normal reconciliation path with the exact provider operation ID;
5. correlate Cal.diy audit or resource state and any embedded AIP booking

metadata;

6. accept `committed` or `not_committed` only from durable trusted evidence;
7. keep the result nonterminal when evidence remains ambiguous;
8. compensate only after the committed effect is identified and the

capability contract permits compensation.

A stale pre-dispatch `claimed` record can become `not_committed`. A stale `dispatching` record becomes `uncertain`. Waiting 120 seconds is therefore not by itself permission to replay a dispatched mutation.

Troubleshoot webhook ingress

The standalone route returns compact HTTP errors:

Code	HTTP	Decision
`webhook.missing_header`	400	Preserve the request and restore both required headers
`webhook.invalid_utf8`	400	Correct the sender or proxy; signature verification requires exact UTF-8 bytes
`webhook.authentication_failed`	401	Check subscription mapping, secret, signature encoding, and raw-body preservation
`webhook.invalid_delivery`	400	Check version, timestamp window, JSON payload, and replay evidence
`webhook.storage_unavailable`	503	Restore replay, event-log, or outbox storage; retry the exact delivery
`webhook.invalid_mapping`	500	Retain the delivery and escalate an impossible connector mapping

A successful response can report `durably_queued` rather than central acknowledgement. Recover central publication from the retained outbox; do not ask Cal.diy to generate a new event. Follow Webhook security and replay for the exact HMAC, timestamp, duplicate, and delivery contract.

Resolve startup failures

Startup failures precede the AIP error surface. Inspect the first process error and compare it with Cal.diy configuration. Common causes include:

- an invalid base URL or account ID;
- a mixed or incomplete authentication mode;
- a secret file that is unreadable or exceeds its configured bound;
- an invalid webhook secret mapping;
- a missing central event endpoint for enabled ingress;
- a database or manifest-discovery failure; or
- a registry identity mismatch.

Correct one owning boundary at a time. Re-run readiness and the gateway-routed profile read after every change; do not combine credential, account, artifact, and registry changes into one diagnostic restart.

Verify the resolution

Close the incident only when:

- the signed error or startup failure no longer occurs for the corrected path;
- `/ready`, the lifecycle lease, and tenant discovery agree;
- a gateway-routed profile read reaches the intended account;
- the original mutation is terminal or remains explicitly paused as uncertain;
- the exact accepted webhook event appears centrally when ingress was affected;
- no credential, raw provider body, webhook secret, or personal data entered

logs or the incident record.

Escalate with:

- the source revision plus artifact and manifest digests;
- the tenant-safe identity set and timestamps;
- the signed error plus action and transaction state;
- provider operation and request IDs;
- remote status, retry hint, and uncertainty flag;
- readiness fields and metric snapshots; and
- webhook event IDs and redacted logs.

Related documentation

- Errors and retry decisions
- [Cal.diy authentication and tenant routing](../getting-started/authentication-and-tenant-routing.md)
- Cal.diy configuration
- Recover the Cal.diy connector
- Cal.diy webhook security and replay