# Qualify the Cal.diy connector

Use this page to decide whether one exact Cal.diy connector artifact has enough evidence for a bounded release claim. It separates source-defined tests, deterministic provider fixtures, standalone-host behavior, and actual Cal.diy observati

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

**Current exact-artifact result: not established by this page.** The pinned source contains qualification tests and procedures, but this documentation review did not execute them. The retained isolated-live result is historical, uses different AIP source bytes and a bundled topology, and cannot qualify the current standalone host.

## Use evidence terms consistently

| Term | Required evidence |
|---|---|
| Implemented | The behavior and its contract exist in the identified source |
| Conformant | The exact artifact passed the applicable conformance scenarios with retained results |
| Controlled-qualified | The exact artifact passed a named deterministic topology and failure matrix |
| Isolated-live verified | The exact artifact produced reviewed observations against an identified Cal.diy deployment |
| Production-observed | Deployment-specific records show the behavior under the stated production boundary |

Source presence proves only implementation. A test function proves that a gate exists, not that it passed for the release under review.

## Bind every claim to one identity set

A qualification result is invalid unless its evidence binds all applicable values:

| Identity class | Required values |
|---|---|
| AIP source | Commit, tree or complete dirty-source digest, and clean or dirty status |
| Connector artifact | Immutable image ID, host component, version, source label, and entrypoint |
| Contract | Manifest digest, 81-capability catalog digest, schemas, and implementation support map |
| Cal.diy upstream | Commit, source tree, image ID, database migration set, and relevant API configuration |
| Harness | Harness source digest, runner image, configuration digest, and test selection |
| Runtime | Architecture, container engine, database versions, trust policy, and topology |
| Run | UTC start and end, unique run ID, operator or CI identity, and evidence-root digest |

A mutable tag, branch name, filename, or successful command exit is not an artifact identity. If the source was dirty, retain enough bytes to reconstruct it; a digest alone distinguishes the input but does not make it reproducible.

## Evaluate four independent gates

| Gate | Source-owned entry point | What it can prove | What it cannot prove |
|---|---|---|---|
| Q1: connector contracts | `connector_contract.rs` and `frozen_conformance.rs` | Deterministic connector semantics through public Rust boundaries | Container, fleet, or external Cal.diy behavior |
| Q2: standalone image | `verify-product-images.sh` | Exact image identity, runtime user and entrypoint, labels, inspect, history, and SBOM | Signature, vulnerability policy, admission, or provider behavior |
| Q3: controlled fleet | `verify-product-fleet.sh` | Standalone registration, routing, profile operations, fail-closed behavior, and recovery against fixtures | Actual Cal.diy compatibility or all 81 operations |
| Q4: isolated live | `live_cal_diy.rs` or `examples/cal-diy-qualification/qualify.sh` | Only the operations and topology observed against the identified Cal.diy deployment | Unobserved operations, arbitrary deployment, scale, or production readiness |

Record a result for each required gate. Q1 cannot replace Q3, and Q3 cannot replace Q4. A release may omit Q4 only when its claim explicitly excludes external-product verification.

## Review deterministic connector evidence

The provider-boundary contract suite defines 11 integration tests. Its covered boundaries include:

- correlated, versioned booking creation and exact idempotent reuse;
- collision and ambiguous-provider failure fencing;
- webhook secret injection, destination binding, and response redaction;
- path, query, and body partitioning for calendar connection events;
- schema rejection before dispatch and canonical input hashing;
- private-link routing, reservation schema closure, and response-size bounds.

The frozen conformance driver evaluates identity and credentials, schemas, idempotency, retry, errors and uncertain outcomes, approvals, transactions, audit and redaction, webhook security, and restart or reconnect. Cancellation races and streaming are explicitly not applicable because the published Cal.diy surface claims neither. The source test requires a passing report with 13 checks.

Retain the actual test report, exact test binary or build identity, stdout and stderr, start and end times, and process status. Do not report the source-level 13-check assertion as an observed pass when no result artifact exists.

These tests exercise behavior families and selected routes. They do not prove that every provider path in the 81-operation catalog was invoked.

## Review the standalone image gate

The source-owned image verifier builds `aip-host-cal-diy` through the allowlisted common host Dockerfile and executes its help surface under read-only, capability-dropped, no-new-privileges controls. It then requires:

- an immutable `sha256` image ID;
- runtime user `10001:10001`;
- entrypoint `/usr/local/bin/aip-connector-host`;
- exact source-revision, version, and component labels;

- retained image inspect, full history, and SPDX JSON SBOM artifacts.

Review those artifacts for the Cal.diy image rather than relying on the shared six-host summary. An SBOM is an inventory, not a vulnerability scan, license decision, signature, or provenance attestation. Supply those release controls through their own evidence roles.

## Review the controlled standalone-fleet gate

The product-fleet procedure uses deterministic provider fixtures. For Cal.diy, the baseline requires:

- exactly 81 admitted capabilities for the identified connector version;
- one gateway-routed `cap:cal_diy:profile.get` result;
- one approval-governed `cap:cal_diy:profile.update` using a stable action ID

and idempotency key;

- provider audit entries for `GET /v2/me` and `PATCH /v2/me`, plus shared audit

assertions that authorization and idempotency markers are present;

- rejection of a deliberately invalid fixture credential.

The outer matrix also includes the Cal.diy host in graceful restart, `SIGKILL`, expired-lease fail-closed, restart recovery, gateway outage, shared PostgreSQL outage, and database recovery cases. It captures registry, database, image, process, event, response, stderr, and timestamped log evidence on both success and failure.

This gate exercises two Cal.diy capability IDs against a controlled fixture. It does not exercise the other 79 capability IDs, Cal.diy webhook ingress, a real Cal.diy deployment, provider version drift, or production traffic. Call it controlled standalone-fleet qualification, not live Cal.diy qualification.

## Review isolated-live evidence

The ignored `live_cal_diy.rs` test defines a small reversible gate against an operator-supplied deployment. It checks authenticated health, creates one future slot reservation, repeats the same key and compares the stored result, then deletes the reservation with a new key. Because the test is ignored and environment-gated, it contributes no result until an exact run is retained.

The larger source-owned campaign under `examples/cal-diy-qualification` deploys an actual pinned self-hosted Cal.diy revision and exercises discovery, availability, booking plan and approval, one booking, idempotent replay, webhook cases, and restart recovery. Its AIP component uses the legacy bundled daemon, not `aip-host-cal-diy`. A new standalone-host release claim therefore needs a new live campaign through the fleet topology.

For every live campaign, prove that the provider is the identified Cal.diy source and image, not a deterministic fixture with compatible routes. Use reversible future-dated data, dedicated accounts, independent requester and approver identities, isolated networks, and a verified cleanup record.

## Interpret the historical result narrowly

The retained 13 July 2026 JSON result records historical status `passed`. Its bytes have SHA-256 digest `507c2136aa695a5e34efddd6fe5dc6b50e46de7d898c5ad39c3d5f7ebbbb82b3`. It identifies an actual Cal.diy commit and image and retains observations for 81-capability admission, availability, approval-governed booking, one-effect idempotent replay, webhook authentication and duplicate handling, and restart recovery.

That result does not apply to the current source revision because:

- its AIP build used a different base commit plus uncommitted source identified

only by digest;

- the unavailable changed bytes prevent exact reconstruction;
- its AIP topology was the bundled daemon rather than the standalone host;
- its retained artifact lacks the current standard's cleanup evidence;
- it observed only the campaign's selected capability paths.

Preserve the historical label and date. Do not rewrite it as a current pass. The full identity and limitation record belongs to Cal.diy isolated-live qualification.

## Account for all 81 capabilities

Maintain one generated or machine-validated coverage row per published capability:

| Field | Required value |
|---|---|
| Capability ID | Exact admitted identifier |
| Contract evidence | Schema, method, API version, route, risk, approval, retry, and transaction mapping |
| Deterministic evidence | Test case and result artifact, or explicit gap |
| Standalone-fleet evidence | Case and artifact, or `not_run` |
| Isolated-live evidence | Provider revision, case and artifact, or `not_run` |
| Failure evidence | Auth, validation, rejection, uncertainty, replay, or recovery cases exercised |
| Side-effect control | Idempotency, approval, plan and commit, compensation, and cleanup as applicable |
| Review decision | Accepted scope, limitation, owner, and review time |

Every row needs contract evidence. Live mutation of all 81 operations is not a default requirement and may be unsafe. Select reversible live cases by unique risk, transport, state, idempotency, approval, and provider behavior, then publish every unobserved capability as a gap. Never turn sampling into an all-capabilities-live claim.

## Retain a complete evidence set

A new qualification record should contain:

1. immutable source, image, manifest, upstream, harness, dependency, topology,

and run identities;

2. a gate manifest naming every required case and its expected invariant;
3. raw exit records plus structured per-case results and redacted stderr;
4. captured admission, catalog, implementation-support, and route snapshots;
5. image inspect, history, SBOM, and separate supply-chain evidence;
6. provider request and audit correlations without credentials or raw personal

data;

7. durable action, approval, transaction, idempotency, reconciliation,

lifecycle, event, and webhook observations where applicable;

8. failure and recovery ordering, including container and database events;
9. a generated 81-row capability coverage matrix;
10. checksums for every retained artifact and one evidence-root checksum;
11. verified cleanup, residual-resource inventory, and any retained exception;

12. a final summary that states scope, result, exclusions, reviewer, and time.

A summary file's presence is not a pass. Review all referenced artifacts and ensure no earlier failure was hidden by later recovery.

## Assign results without ambiguity

| Result | Use when |
|---|---|
| `passed` | Every required invariant passed for the exact identity set, artifacts are complete and redacted, gaps match the claim, and cleanup was verified |
| `failed` | Any required invariant failed, identity mismatched, evidence was corrupted or leaked, or cleanup violated the campaign contract |
| `blocked` | A required dependency or authority was unavailable before the invariant could be evaluated |
| `not_run` | The gate was not executed for this identity set |
| `historical` | A prior result is retained for a different source, artifact, topology, or evidence standard |

Do not collapse `blocked`, `not_run`, or `historical` into `passed`. A gate can pass while the broader release claim remains unestablished.

## Publish only the supported claim

Acceptable result language names the evidence boundary:

> At the recorded time, the identified AIP and Cal.diy artifacts passed the
>
> listed deterministic, standalone-fleet, and selected isolated-live cases in
>
> the retained topology. The coverage matrix lists all unobserved capabilities
>
> and excluded production properties.

Do not publish `fully compatible`, `all 81 operations live verified`, `exactly-once`, or `production-ready` unless the retained campaign defines and proves each phrase under the stated deployment.

## Reviewer checklist

- [ ] Every source, artifact, manifest, upstream, harness, and run identity is

immutable and mutually consistent.

- [ ] Required gates are explicit; no fixture result is labeled live.
- [ ] The 81-row coverage matrix matches the admitted catalog.
- [ ] Deterministic reports and standalone-fleet failure evidence are complete.
- [ ] Live evidence names the real Cal.diy revision, topology, data, and

reversible cleanup.

- [ ] Mutation results include approval, idempotency, provider effect, and

uncertainty or reconciliation evidence.

- [ ] Image inventory is not presented as signature, scan, or provenance.
- [ ] Artifacts are redacted, checksummed, access-controlled, and retained for

the claim lifetime.

- [ ] Cleanup and residual resources are independently recorded.
- [ ] The final wording states time, scope, result, gaps, and exclusions.

## Related documentation

- Conformance and qualification
- Connector fleet qualification
- Live-product qualification
- Cal.diy capability index
- Release artifacts and verification