Live product end-to-end qualification

Use this procedure to prove that one identified AIP build can interact safely with one identified deployment of Cal.diy, Hermes Agent, Chatwoot, Dify, CrewAI, or Twenty. A valid campaign exercises a real product boundary in an isolated envi

SECTION	Qualification and Evidence
TYPE	Qualification evidence
PROTOCOL	AIP 1.0
SOURCE	docs/testing/live-product-e2e.md

This page describes how to design and review a campaign for GetAIP `2.0.0` source revision `d7cce13d1d555644d04a4d73c66c95b113737635`. The protected controlled product-fleet gate is not an isolated-live external-provider campaign.

Know what the source can run

The pinned source does not contain a general live suite for all six products. Use only the entry points in this table, or record a separately reviewed campaign driver as an external artifact.

Product	Source-owned product-facing entry point	Evidence boundary
Cal.diy	Complete isolated Docker qualification and a narrower ignored reservation test	The complete campaign covers governed booking, replay, webhook, restart, and persistence; the ignored test covers reservation, replay, and release only
Hermes Agent	Deployed-stack operator and delegation binary	Exercises MCP and native approval paths against identified Hermes endpoints and a controlled downstream AIP capability
Chatwoot	No dedicated live-product driver at the pinned revision	Deterministic connector tests do not prove a live Chatwoot deployment
Dify	No dedicated live-product driver at the pinned revision	Deterministic connector tests do not prove a live Dify deployment
CrewAI	Ignored exact-upstream sidecar test	Proves the Rust connector and real CrewAI library boundary; the supplied qualification registry uses a deterministic network-free language model
Twenty	No dedicated live-product driver at the pinned revision	Deterministic connector tests do not prove a live Twenty deployment

The optional five-product fleet verifier uses controlled upstream behavior and does not include Twenty. It is useful AIP integration evidence, but it is not live-provider evidence for any product.

Define the claim before the run

Write one sentence that names the product, provider deployment, connector artifact, AIP artifact, campaign revision, and operations in scope. Do not use the broader phrase “the connector is qualified” when the run covers only a subset of operations.

For example:

> The identified AIP and connector artifacts completed the documented Cal.diy
>
> booking campaign against the identified isolated provider deployment, with

cleanup confirmed, at the recorded time.

The campaign must not expand its claim to another provider version, another connector artifact, untested operations, production safety, or portable performance.

Complete the safety and identity preflight

Do not begin a mutating step until every item below has an owner and a recorded value.

Check	Required record
AIP identity	Full source revision, dirty-tree state, build or image digest, configuration digest, and manifest digest
Connector identity	Connector artifact digest, exact upstream pin, capability-set digest, and declared operation surface
Provider identity	Product version or source revision, deployment or image digest, enabled feature flags, and migration state
Isolation	Dedicated non-production tenant, account, workspace, application, crew, or calendar with no production data
Mutation budget	Exact objects the run may create, update, cancel, or delete, plus a cleanup owner and deadline
Principals	Separate requester and approver identities where approval is tested; least-privilege product identity
Credentials	Protected file or secret-store references; record only names, scopes, expiry, and fingerprints
Network	Endpoint names, trust roots, proxy path, redirect policy, and relevant allowlists without secret values
Time	UTC clock source and measured skew for signatures, deadlines, leases, and replay windows
Evidence	New owner-only directory, redaction rules, retention period, and reviewer

Confirm that every planned mutation is reversible. If a product operation has no reliable cleanup path, use a disposable deployment that can be destroyed as a unit. Never discover this constraint after writing shared customer data.

Run one complete campaign

Use the same sequence for every product, adapting only the product operations.

1. Record all identities and prove that the isolated target is ready.
2. Run the connector's deterministic tests and the applicable conformance

checks. Retain their results separately from live evidence.

3. Perform authenticated, non-mutating discovery and baseline reads.
4. Submit one governed mutation through AIP, including approval when required.
5. Repeat the same action, idempotency key, or transaction identity and verify

that it does not create an additional external effect.

6. Exercise the supported cancellation, timeout, or recovery boundary without

inventing behavior the connector does not declare.

7. When webhooks are supported, verify the raw-body signature, freshness,

duplicate suppression, and rejection of invalid authentication.

8. Restart the relevant AIP or connector process and verify durable status,

provider references, and recovery behavior.

9. Remove or revert every created provider object and verify its absence or terminal cleanup state through an independent read.
10. Redact, hash, inventory, and review the retained evidence before assigning a result.

Record unsupported operations as `NOT_RUN`; do not replace them with similar operations or silently omit them.

Apply the product-specific minimum

A product campaign is complete only when it covers the applicable minimum in this table. The connector documentation defines the exact inputs, exclusions, and product-specific cleanup steps.

Product	Minimum live assertions
Cal.diy	Ready health; availability read; mutation-free booking plan; independent approval; one booking; identical idempotent replay; signed webhook acceptance and replay rejection; restart recovery; booking or reservation cleanup
Hermes Agent	Endpoint health and discovery; streaming result; independent approval; governed AIP tool lifecycle; scoped delegation; out-of-scope rejection; cancellation or terminal status; reconnect or restart observation
Chatwoot	Account-scoped read; reversible conversation, private-note, status, or handoff mutation; idempotent replay; authenticated webhook; event-loop prevention; object or state cleanup
Dify	Application parameter discovery; applicable synchronous and streaming execution; cancellation; human-input or pause state when supported; task recovery; applicable knowledge mode; cleanup of disposable inputs and tasks
CrewAI	Sidecar authentication; exact registered crew discovery; real CrewAI crew execution; stream and terminal status where enabled; cancellation; replay behavior; restart fencing; one authoritative writer; separate identification of any external model provider
Twenty	OpenAPI-derived core-record read; reversible record lifecycle; metadata read; webhook HMAC and nonce replay rejection; credential redaction; proof that the connector did not use the excluded GraphQL or legacy API-key paths; record cleanup

If the product cannot supply one applicable assertion, report the gap and stop at `BLOCKED` or `FAIL`. A deterministic substitute cannot fill a live-product gap.

Use only existing source-owned entry points

Run commands from the root of the pinned AIP checkout. The examples below show the available interfaces; operators must supply their own protected values and must satisfy the preflight before execution.

Complete Cal.diy campaign

```
SH
examples/cal-diy-qualification/qualify.sh
```

The script owns an isolated Compose project and evidence directory. Review its destructive scope and configured paths before running it.

Narrow Cal.diy reservation probe

```
SH
AIP_E2E_CAL_DIY_URL="https://isolated-cal.example/api" \
AIP_E2E_CAL_DIY_API_KEY="$CAL_DIY_TEST_API_KEY" \
AIP_E2E_CAL_DIY_EVENT_TYPE_ID="42" \
AIP_E2E_CAL_DIY_SLOT_START="2050-01-01T10:00:00Z" \
cargo test -p aip-connector-cal-diy \
  live_cal_diy_reservation_round_trip_uses_the_aip_connector \
  -- --ignored --nocapture
```

This probe checks ready health, one reservation, identical replay output, and release. It does not replace the complete Cal.diy campaign.

Exact-upstream CrewAI sidecar probe

```
SH
AIP_E2E_CREWAI_URL="http://127.0.0.1:8090" \
AIP_E2E_CREWAI_CREW_ID="support-qualification" \
AIP_E2E_CREWAI_BEARER_TOKEN="$CREWAI_TEST_TOKEN" \
cargo test -p aip-connector-crewai \
  exact_upstream_sidecar_executes_a_real_crewai_crew \
  -- --ignored --nocapture
```

The supplied upstream image installs CrewAI from the exact recorded source checkout. Its qualification registry uses a deterministic, network-free language model. The result therefore proves a real CrewAI library mapping, not a live external model-provider integration.

Hermes deployed-stack probe

```
SH
GETAIP_SERVER_MCP_ACCESS_TOKEN="$MCP_TEST_TOKEN" \
GETAIP_SERVER_NATIVE_BEARER_TOKEN="$APPROVER_TEST_TOKEN" \
cargo run -p aip-connector-hermes-agent \
  --bin getaip-server-hermes-operator-smoke -- \
  --mcp-url "https://isolated-aip.example/mcp" \
  --server-url "https://isolated-aip.example" \
  --endpoint-id "hermes-qualification"
```

Repeat `--endpoint-id` to check more than one deployed Hermes endpoint. This driver requires different MCP and native approval credentials and emits a JSON report. Redact the report before retention.

There is no equivalent source-owned live command for Chatwoot, Dify, or Twenty at the pinned revision. A campaign for those products needs a separately reviewed driver and must retain its source revision with the evidence.

Retain a reviewable evidence package

Store raw evidence in an access-controlled location and publish only a redacted report. The retained package must include:

- a manifest of AIP, connector, upstream, provider, configuration, and

campaign identities;

- the written claim, scope, operator, reviewer, UTC start, and UTC end;
- a step ledger with command or request identity, expected invariant, result,

and evidence path;

- redacted requests, responses, stream records, webhooks, and provider reads;
- approval, action, transaction, idempotency, provider-reference, and audit

correlation identifiers;

- before-and-after provider state plus cleanup verification;

- process restart, cancellation, timeout, and recovery observations;
- stdout and stderr, exit status, environment-name inventory, and redaction

log;

- a checksum manifest covering every retained artifact;
- the final result and every unresolved limitation.

Do not retain bearer tokens, API keys, cookie values, signing secrets, raw customer data, or complete secret-bearing environment dumps.

Assign the result

Use exactly one result state.

Result	Meaning
`PASS`	Every applicable planned assertion passed, evidence is complete, and cleanup was independently confirmed
`FAIL`	An assertion was exercised and produced a contradictory or unsafe result
`BLOCKED`	A prerequisite, safe mutation, cleanup path, or required observation was unavailable
`NOT_RUN`	No campaign was attempted for the identified artifact set

A skipped ignored test is `NOT_RUN`, not `PASS`. A successful process exit is insufficient when evidence is incomplete. Cleanup failure makes the campaign a `FAIL`, even when all product operations previously succeeded.

Interpret existing reports narrowly

The retained Cal.diy report and Hermes Agent report describe historical campaigns and identify the artifacts they evaluated. They do not qualify the current pinned AIP revision by inheritance. Use them as report-shape examples and as evidence only for their recorded identities and scope.

Related documentation

- Testing and qualification
- Connector upstream baselines
- Conformance reference
- Implementation status
- Connector documentation