Testing and evidence index

Use this index to choose the smallest procedure that can support your claim and to find the evidence you must retain. Start with the claim, not with whichever test command is easiest to run.

SECTION	Qualification and Evidence
TYPE	Documentation
PROTOCOL	AIP 1.0
SOURCE	docs/testing/README.md

The pinned GetAIP `2.0.0` source contains tests, conformance suites, qualification scripts, and CI definitions. The retained release record shows that the six protected contexts and their named controlled gates passed for revision `d7cce13d1d555644d04a4d73c66c95b113737635`. It does not extend that result to a rebuilt image, external provider, or target deployment.

Choose evidence by the claim

Question you need to answer	Start here	Evidence boundary
What does the pinned source implement?	Implementation status	Code presence is not an executed result
Does one schema instance, envelope, manifest, connector, or MCP surface satisfy its contract?	Conformance and qualification	Run the complete applicable matrix and assert its expected checks
Do registry, routing, host, lease, revocation, outage, and recovery paths work together?	Connector fleet qualification	Controlled multiprocess evidence is not product-provider evidence
Does the registry meet its declared fleet-size thresholds?	Fleet scale SLO	Results apply only to the recorded environment and dataset
Which upstream product revision must a connector campaign use?	Pinned upstream baselines	A source pin does not qualify a built provider deployment
How should an external product campaign be prepared and retained?	Live product E2E	Each product, credential policy, topology, and image needs its own evidence
What did the older Cal.diy campaign prove?	Cal.diy isolated-live evidence	The result covers only the artifacts named in that report
What did the older Hermes Agent campaign prove?	Hermes Agent isolated-live evidence	The result covers only the two named endpoints and their dependencies
Was a release built, signed, attested, and verified?	Release artifacts	Workflow source is not publication evidence

If the claim spans several rows, retain evidence for every row. A product campaign does not replace protocol conformance, and a release signature does not replace deployment recovery or capacity testing.

Follow the evidence progression

Use the layers in this order so a later failure does not hide a cheaper, more local defect:

1. validate schemas and typed invariants;
2. run deterministic unit, property, fixture, and recovery tests;
3. run every applicable core, profile, and connector conformance suite;
4. exercise process, storage, routing, restart, and failure boundaries;

5. run isolated-live campaigns against pinned external products or peers;
6. build and verify immutable release artifacts;
7. qualify the target deployment's security, recovery, fault, and capacity

requirements.

The layers are cumulative only when they refer to the same source and build lineage. If an image is rebuilt after conformance, identify the new digest and rerun every affected layer.

Use the source-owned procedures deliberately

Procedure family	Source-owned entry point	What its source is designed to cover
Source release gate	`cargo run -p xtask -- release-check`	Format, boundaries, semantic version checks, schemas, lint, tests, dependencies, fuzz compilation, and API docs
Core diagnostic	`getaip conformance run`	One built-in message body or one supplied native envelope
MCP client diagnostic	`getaip mcp conformance`	Four external-client checks only
Connector harness	`run_connector_conformance`	Atomic manifest admission and twelve conditional behavior families
PostgreSQL recovery	`aip-storage-postgres` recovery test	Durable runtime recovery behavior
Registry and control plane	PostgreSQL registry and control-plane tests	Schema, roles, routing, lifecycle, concurrency, and reconnect behavior
Fleet failure matrix	`verify-failure-matrix.sh`	Multiprocess trust, routing, failover, revocation, database outage, and durable execution boundaries
Image boundaries	Migration and product image verification scripts	Labels, users, entrypoints, component separation, image IDs, histories, and SBOMs
Controlled product fleet	`verify-product-fleet.sh`	A 45-case configured product-host matrix with controlled upstream behavior, restart, failover, and database isolation
Scale gate	Registry scale test using `aip-fleet-slo-v1`	Fixed numeric seeding, latency, throughput, memory, and database-growth thresholds
Cal.diy isolated-live	`examples/cal-diy-qualification/qualify.sh`	One pinned Cal.diy deployment and reversible booking lifecycle

Read the detailed page before running a destructive or external procedure. Some scripts create and remove dedicated containers and volumes, require owner-only secret files, or mutate an isolated product account.

The controlled product-fleet script ran in protected Gitea Actions run `4242` for the `v2.0.0` release and its 45-case matrix passed. It remains controlled integration evidence; every product, including Twenty, requires separate external-provider evidence for a live-provider claim.

Retain a complete evidence package

An acceptable package identifies both the subject and the observation. Retain at least:

- full source commit, tree state, and build or image digest;
- protocol, profile, manifest, schema, connector, and upstream revisions in

scope;

- exact commands, arguments, tool versions, and expected check IDs;
- configuration class, topology, trust boundaries, and credential revision

references without secret values;

- UTC start and finish times plus the final status;

- raw reports, logs, query results, request traces, and lifecycle identifiers

required to reproduce each assertion;

- a digest and size for every retained artifact;
- failed, skipped, blocked, and not-applicable checks, not only successes;
- limitations, environmental conditions, and required follow-up work.

Use explicit states such as `PASS`, `FAIL`, `BLOCKED`, `NOT_RUN`, and `NOT_APPLICABLE`. Record why a check is not applicable and tie that reason to the published contract. Never use an empty report as proof; both shared report aggregators consider an empty check list successful.

Protect evidence without weakening it

Evidence must be useful to an evaluator and safe to retain:

- replace credentials with opaque revision or fingerprint references;
- redact tokens, cookies, authorization headers, webhook secrets, personal

data, prompts, and provider payload fields outside the assertion;

- preserve hashes and correlation IDs when they are needed to prove identity

or ordering;

- keep raw artifacts access-controlled and publish only the bounded redacted

view;

- record the redaction method and verify that it did not change the asserted

fact.

Do not paste customer data or secret material into a Markdown report. A useful report points to retained evidence by immutable identifier.

Apply retention and expiry rules

The reviewed GitHub CI definition retains fleet-scale logs for 30 days and fleet failure and image evidence for 14 days. Local fleet scripts write to bounded `.getaip-server-*` directories; the controlled product-fleet script creates a new UTC-named run directory. These are operational defaults, not a sufficient release-retention policy.

Copy evidence needed for a release decision into the protected release record before CI expiry. Keep immutable release digests, signature verification, attestations, and required qualification reports for at least the supported lifetime of that release.

Rerun affected evidence after a change to source, dependencies, upstream revision, image, credentials policy, schema, topology, trust configuration, storage, network boundary, or capacity target. Mark the older result historical instead of silently editing its artifact identity.

Review an evidence package

Before accepting a claim, verify all of the following:

- the artifact under review is the artifact named in the report;
- the expected suites and check IDs are complete and nonempty;
- every applicable failure or omission is visible;
- every referenced artifact resolves and matches its recorded digest;

- controlled fixtures are not described as external-provider runs;
- historical results are not assigned to a later build;
- the claim uses `implemented`, `conforming`, `qualified`, or `production-ready` only at the evidence level actually reached.

If any identity or required artifact is missing, the strongest defensible result is `not established`, not an inferred pass.

Related documentation

- Implementation status
- Conformance and qualification
- Connector documentation
- Release artifacts