Connector fleet qualification and failure matrix

Use this procedure to evaluate routing, lifecycle, trust, persistence, and failure recovery across a complete isolated connector fleet. The source-owned matrix exercises real AIP processes and PostgreSQL state against controlled provider be

SECTION	Qualification and Evidence
TYPE	Qualification evidence
PROTOCOL	AIP 1.0
SOURCE	docs/testing/connector-fleet-qualification.md

This page describes the procedure at source revision `97be86e9efedf07ecf1783b03800f683f107fb04`. It does not record a run for that revision. A result becomes evidence only after the exact source and generated artifacts are identified and the retained report is reviewed.

Understand the destructive scope

The script owns a fixed Compose project and removes that project's containers, networks, and volumes before and after the run. It also removes and recreates the selected evidence directory.

Before continuing:

- use a dedicated Docker host or confirm that no other run uses the Compose

project `aip-connector-fleet-qualification`;

- choose a new, dedicated evidence path that contains no valuable files;
- reserve enough time, CPU, memory, and disk for local Rust image builds and

several PostgreSQL-backed processes;

- record the full Git commit, tree state, Docker and Compose versions, host

architecture, and UTC start time;

- do not place provider credentials or production data in this topology.

The preparation step writes `deploy/connector-fleet/.env.qualification` with mode `0600`. It derives the source identity from tracked and unignored files, so a dirty worktree produces a different `workspace-sha256` identity. Preserve both the Git state and derived identity in the report.

Know the isolated topology

Component	Role in the matrix
TLS edge	Terminates the internal HTTPS routes used by the qualification client
Two AIP gateways	Prove routing through either data-plane replica
Lifecycle service	Handles signed connector registration, heartbeat, drain, and offline transitions
PostgreSQL	Holds the registry plus separately owned runtime and controlled-provider databases
Three connector instances	Represent two controlled connector types across two tenants
Four connector replicas	Provide one two-replica instance and two single-replica instances
Qualification client	Sends signed native calls and catalog queries with tenant-specific identities
Controlled providers	Expose deterministic reads, mutations, approvals, and audit facts

The Compose network is internal. Runtime services use distinct identities, signing material, credential policies, and PostgreSQL roles. Connector hosts do not receive the catalog-administrator credential.

Review the baseline

The baseline runner must complete eight operations before the failure matrix starts:

Baseline operation	Required observation
Four signed native calls	Both tenants, both controlled connector types, and both gateways return completed actions
Tenant capability query	The permitted capability appears in the bounded catalog
Signed catalog query for the allowed tenant	The second controlled capability is visible
Signed catalog query for the isolated tenant	The same capability is absent
Signed orchestration plan	Verification returns `verified` with snapshot fencing enabled

Its machine-readable summary must state `passed`, `cases: 8`, `gateways: 2`, `connector_instances: 3`, and `connector_replicas: 4`. These values establish the expected topology; they are not a capacity or product result.

Review every failure family

The outer matrix performs the following cases in order. A later success does not erase an earlier missing or failed case.

Case	Fault or transition	Acceptance invariant
Baseline and registry topology	Start the isolated stack and inspect live leases	The eight baseline operations pass and exactly four replicas are ready with unexpired leases
Credential revision revocation	Revoke the connector's current credential revision	The call fails closed with the expected credential-revision error
Credential policy restore	Atomically restore the permitted revision	The same bounded read completes
Tenant-provider isolation	Ask one tenant for another tenant's provider record	The request fails closed
Gateway replica failover	Stop the first gateway	The same signed read completes through the second gateway
Graceful connector failover	Stop one connector replica cleanly	The registry marks it offline and routing continues through its peer
Lease-expiry fencing	Hard-kill the remaining replica and disable automatic restart	After lease expiry, routing fails closed instead of using the dead replica
Host recovery	Restart the hard-killed replica	Health and signed routing recover

Case	Fault or transition	Acceptance invariant
Idempotent mutation replay	Complete one governed mutation, hard-kill its host, then replay the same key	The provider retains one effect, one approval request, and one creation audit event
Synchronous crash windows	Kill the selected host at each of five durable boundaries, then retry the same identifiers	Every retry completes with exactly one provider effect and one durable result chain
Shared database outage	Stop PostgreSQL while registry and runtime paths are active	Calls fail closed
Database recovery	Restart PostgreSQL without replacing the AIP processes	Gateways, lifecycle service, hosts, and signed routing recover
Admission revocation	Revoke the signed connector admission package	Routing through the revoked package fails closed

Verify the five crash windows

Each crash case reuses one stable action ID, transaction ID, idempotency key, instance, and replica. The controller waits for a durable observation marker, sends `SIGKILL`, restarts the process, and retries the same logical operation.

Crash boundary	What has happened before the kill
`before_intent_persistence`	No durable execution intent is promised
`intent_persisted`	The runtime intent is durable
`provider_effect_committed`	The controlled provider has committed its effect
`provider_response_received`	The provider response has reached the connector
`result_persisted`	The terminal AIP result is durable

After every recovery, independent provider and runtime queries must find exactly one of each required record:

- provider effect;
- approval request;
- provider creation audit event;
- settled idempotency result;
- terminal action result;
- transaction receipt chain.

This is the decisive exactly-once-effect assertion. A completed HTTP response without those independent counts is insufficient.

Run the source-owned matrix

From the exact repository root, assign a fresh evidence directory and run:

```
SH
export AIP_FLEET_EVIDENCE_DIR="$PWD/.aipd-fleet-qualification/run-manual"
deploy/connector-fleet/verify-failure-matrix.sh
```

The script calls `prepare.sh` and builds locally identified images. It then initializes the isolated trust and database state, executes the baseline and failure matrix, captures diagnostics on both success and failure, and tears down the dedicated Compose project with its volumes.

Do not point `AIP_FLEET_EVIDENCE_DIR` at the repository root, a shared artifact directory, or any path that must survive `rm -rf`. Do not run two matrices concurrently because the Compose project and generated environment file are shared.

Inspect the retained evidence

The selected directory should contain:

Artifact	Review purpose
`summary.json`	Final script status and completion time
`failure-matrix.jsonl`	Ordered case, status, detail, and UTC record for the matrix
`qualification/summary.json`	Baseline topology and case totals
`qualification/operations.jsonl`	Per-operation tenant, gateway, time, status, and response reference
Per-case JSON and stderr files	Native result or fail-closed observation for each probe
Crash checkpoint and retry files	Durable stage observation, interrupted outcome, attempts, and recovered result
Route snapshots	Assignment, lease, capacity, active work, health, and circuit state around recovery
`registry-final-state.json`	Final replicas, routes, and admission counters
`docker-events.jsonl`	Container lifecycle ordering
`compose-ps.jsonl` and `compose.log`	Final process state and timestamped service diagnostics
`compose-down.log`	Dedicated-project teardown result

The script writes `summary.json` even when it fails. A file's presence is not a pass. Accept the run only when all of these conditions hold:

1. the final summary says `passed`;
2. the ordered matrix contains every expected case and the final `complete`

record;

3. the baseline contains all eight expected operations;
4. every crash window has its checkpoint, retry history, recovered result, and

independent one-record assertions;

5. all referenced artifacts exist, are redacted, and have retained digests;
6. the source, local image IDs, environment, and time interval are recorded;
7. no earlier failure was hidden by a later recovery.

Preserve CI evidence before expiry

The reviewed GitHub CI workflow runs this matrix together with migration-image and product-image verification and retains the three evidence directories for 14 days. The reviewed Gitea workflow declares 30 days. Copy evidence required for a release decision into a protected, immutable release record before the shorter retention window expires.

CI artifacts still require review. Confirm that the uploaded source commit is the commit under evaluation and that the matrix did not produce an incomplete but superficially green report.

State the result narrowly

A complete run can support this form of claim:

> The identified AIP artifacts passed the controlled connector-fleet failure
>
> matrix in the recorded isolated topology at the stated time.

It cannot establish external product behavior, the six public connector catalogues, fleet-scale SLOs, independent protocol interoperability, published image signatures, arbitrary infrastructure, or production readiness. Use live product E2E, the fleet scale SLO, and deployment-specific campaigns for those claims.

Related documentation

- Testing and evidence index
- Conformance and qualification
- Connector fleet architecture
- Registry and routing
- Live product E2E