Qualify the Twenty connector

Use this procedure to evaluate one exact Twenty connector deployment. The result applies only to the recorded artifact, manifest, provider revision, workspace, credentials, configuration, topology, and executed cases.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/twenty/qualification/README.md

Source inspection, unit tests, image checks, and admission are separate gates. None establishes provider behavior without retained execution evidence.

Define the qualification claim

State one bounded claim before execution:

> The identified Twenty connector artifact preserves AIP identity, workspace isolation, governed CRM operations, mutation uncertainty, authenticated webhook admission, replay fencing, and recovery for the executed cases.

Exclude unexecuted operations, arbitrary provider revisions, downstream automation, provider exactly-once behavior, and release attestation. Record every case as `PASS`, `FAIL`, or `NOT RUN`.

Freeze every identity

Record these values in the run manifest:

Identity	Required value
AIP source	Full commit, clean-tree digest, Cargo lock digest, and target architecture
Host artifact	Immutable image digest, OCI labels, SBOM, provenance, signature, and binary version
Connector manifest	Manifest digest, profile, 22-operation catalogue digest, and admitted subset
Twenty mapping	Upstream revision `96a24563674313a3071d359bfccaf33d5e130ab8`
Provider	Immutable image digest or hosted release identity and API compatibility record
Workspace	Workspace UUID, external-account ID, tenant binding, and sanitized fixture identity
Credentials	Non-secret token ID, revision, scope, issuer, expiry, and rotation policy
Webhooks	HMAC-secret revision, endpoint identity, replay scope, and event destination
Fleet	Type, version, instance, replica, tenant, assignment, region, zone, and lease identity
State	PostgreSQL identity, schema state, backup point, and retention policy
Runner	Operating system, architecture, container engine, Rust, CLI, and clock source
Time	Run ID, UTC start and completion, operator, and reviewer

Use immutable artifacts during execution. Any identity change creates a new qualification claim.

Prepare isolated fixtures and evidence

Use a dedicated workspace containing synthetic records and metadata. Do not run destructive cases against an ordinary CRM workspace.

Create an owner-controlled evidence directory:

```
TEXT
twenty-qualification/<run-id>/
■■■ identity/
■■■ source-tests/
■■■ images/
■■■ admission/
■■■ operations/
■■■ webhooks/
■■■ failures/
■■■ recovery/
■■■ cleanup/
■■■ results.jsonl
■■■ summary.json
■■■ limitations.md
```

Keep secrets and unredacted CRM data outside the publishable bundle. Retain cryptographic digests and redacted evidence indexes instead.

Gate 1: verify source contracts

At the reviewed AIP revision, verify:

- connector ID `twenty` and profile `aip.connector.twenty.v1` remain stable;
- the upstream mapping revision is the frozen Forty-character commit above;
- exactly 22 unique operations map to unique capability IDs;
- eight reads are low risk and fourteen mutations require approval and idempotency;
- the catalogue contains twenty tools and two resources;
- provider routes remain fixed REST and metadata routes without a generic proxy;
- request, response, workspace, object-name, UUID, query, and body guards remain active;
- provider redirects remain disabled and Bearer authentication remains fixed;
- webhook HMAC, clock, nonce, workspace, event, replay, and redaction checks remain active;
- manifest and public capability assets match the implementation.

Fail this gate on source or generated-asset drift. Do not repair a mismatch by editing only a manifest or documentation asset.

Gate 2: run the connector tests

The package test gate is:

```
SH
cargo test --locked -p aip-connector-twenty
```

Retain the exact command, environment, exit status, and complete output. Confirm the eight source tests cover:

1. complete, unique, governed capability projection;
2. unsafe configuration and provider-identifier rejection;
3. deterministic create identity and forced provider upsert;

4. non-empty bulk filters and explicit permanent-delete mode;
5. host-owned webhook secrets and recursive response redaction;
6. signed webhook mapping and replay rejection;
7. replay expiry pruning and capacity enforcement;
8. mutation uncertainty during temporary provider failures.

These tests establish local mapping invariants. They do not contact a real Twenty provider or prove a built artifact.

Gate 3: verify the host image

Apply the source-owned product image policy to `aip-host-twenty`:

Check	Required result
Image identity	Immutable SHA-256 image ID and admitted digest agree
Runtime user	`10001:10001`
Entrypoint	`/usr/local/bin/aip-connector-host`
Component label	`aip-host-twenty`
Revision label	Exact AIP source identity
Version label	Exact connector release version
Runtime probe	Hardened read-only container returns `--help` successfully
Supply chain	Inspect, full history, SPDX SBOM, signature, provenance, and policy decision retained

The generic verifier builds and inspects the image. It does not start a Twenty workspace or execute an AIP operation.

Gate 4: verify admission and workspace isolation

Require one verified admission package whose artifact and manifest digests match the running replica. Confirm the admitted operation subset equals discovery.

Verify all fleet identities in signed requests, registry state, host configuration, and responses. The provider workspace UUID must equal the intended external-account binding under deployment policy.

Use two isolated tenants and workspaces for negative cases:

Case	Required result
Authorized tenant and workspace	Discovery and bounded read succeed through the admitted replica
Authorized tenant, wrong external account	No eligible route or provider dispatch
Unauthorized tenant	No discovery result and no eligible route
Valid token, wrong configured workspace	Qualification fails before any claim of isolation
Cross-workspace record ID	No data disclosure or mutation outside the bound workspace

Preserve registry, gateway, host, and provider evidence for each negative case.

Gate 5: exercise all 22 operations

Execute every admitted operation against synthetic fixtures. Record request, approval, provider dispatch, result, verification read, cleanup, and failure state for each mutation.

Family	Operations requiring individual evidence
Query	`record.list`, `record.get`, `record.find_duplicates`, `record.group_by`
Create and update	`record.create`, `record.create_many`, `record.update`, `record.update_many`
Delete and merge	`record.soft_delete`, `record.soft_delete_many`, `record.destroy`, `record.destroy_many`, `record.restore`, `record.restore_many`, `record.merge`
Metadata	`metadata.list`, `metadata.get`, `metadata.create`, `metadata.update`, `metadata.delete`
OpenAPI	`openapi.core`, `openapi.metadata`

For reads, verify bounds, filters, pagination, object identity, workspace, and redaction. For mutations, verify before and after state without assuming that an AIP terminal result proves every downstream effect.

Gate 6: verify governance and idempotency

For each of the fourteen mutations, retain:

- immutable Action and capability identity;
- canonical input hash and idempotency-key digest;
- approval request, decision, policy hash, and evidence references;
- provider request identity when supplied;
- terminal AIP result or structured error;
- verification read and cleanup result.

Replay the same Action and key with identical input. Require one governed intent and one stable result. Change the input under the same key and require collision rejection before provider dispatch.

For `record.create`, `record.create_many`, and `record.update`, verify the documented safe-with-key path. Treat every other mutation as unsafe to retry.

Interrupt provider responses after possible commit. Require uncertain mutation classification and provider-state reconciliation without a replacement Action or key.

Gate 7: verify mutation safeguards

Run bounded negative cases for:

- missing approval or idempotency key;
- empty, absent, `null`, or `{}` bulk filters;
- permanent-delete requests without explicit permanent mode;
- malformed object names and record UUIDs;
- oversized request and response bodies;
- invalid metadata resource and query combinations;
- supplied webhook secret in metadata input;
- unsupported page-layout field combinations;
- direct provider GraphQL or arbitrary-route attempts.

Require rejection before provider dispatch when the connector owns the guard. Retain provider evidence when enforcement belongs to Twenty.

Gate 8: verify webhook security and delivery

Enable webhook ingress only with shared durable replay state. Exercise these cases with the exact raw request body:

Case	Required result
Valid delivery	HMAC accepted, deterministic event created, and central outcome recorded
Same nonce replay	Rejected without a second event
Forged signature	Rejected without replay or event persistence
Timestamp outside five minutes	Rejected
Wrong workspace UUID	Rejected
Missing required header	Rejected before connector admission
Body above four MiB	Rejected
Replay store unavailable	Fail closed without accepting the nonce as complete
Central endpoint unavailable	Local durable outbox owns exactly one accepted event

Confirm replay capacity at 100,000 live nonces and verify expiry pruning. Run a concurrent same-nonce race across every replica of one logical instance.

Gate 9: verify errors, limits, and redaction

Exercise provider `401`, `403`, `429`, ordinary `4xx`, `5xx`, connection failure, post-dispatch transport failure, invalid JSON, and oversized response.

Verify the exact connector error class, retry flag, retry delay, remote status, provider request ID, uncertainty flag, source component, and operation. Confirm that returned provider details use the bounded allowlist.

Inspect logs, traces, AIP errors, events, and retained files for API tokens, webhook secrets, signing seeds, raw idempotency keys, raw HMAC material, and unapproved CRM data. Any secret disclosure fails qualification.

Gate 10: verify restart and fleet behavior

Execute each fault with before, fault, after, and recovery evidence:

Fault	Required result
Graceful host restart	Same instance state returns ready and prior results remain readable
Host termination during read	Route fails closed and bounded retry follows read policy
Host termination during mutation	Original Action remains fenced until provider state is reconciled
PostgreSQL outage	Readiness fails and no stateful admission bypass occurs
Registry lease expiry	Replica becomes ineligible and recovers with the same logical identity
Credential rotation	Old revision stops authorizing and new revision requires policy agreement
Provider outage	Readiness and operation errors preserve their distinct evidence
Two host replicas	Shared Action, idempotency, replay, event, outbox, and lease state remain coherent

Do not replace PostgreSQL with empty state during recovery. Preserve the original Action, replay scope, event, outbox, instance, and replica evidence.

Inspect retained local evidence separately

The retained bundle inspected for this page records one historical local run with these identities:

Field	Retained value
Scope	`local_e2e_qualification_not_release_attestation`
AIP source	`workspace-sha256:43f82b2d60d04f08750d3c7bc0eb2564e53b6d20a5ad6de06d3ce5f618605fdc`
Host artifact	`sha256:35e12b264ad28ee21150a77d8de6aaaf8b86d264822998b6465399564f492ff5`
Manifest	`sha256:a23d7fdc41dfe1f729ea84971f1778d357d5fa7d0eff75e13aa18fb5cac8e495`
Live evidence	`sha256:810714686150b4846936b2d1880ad5925585f15830fd5cec8930b9a9d02224ac`
Twenty mapping	`96a24563674313a3071d359bfccaf33d5e130ab8`
Admission plan	`verified` for the recorded artifact and version

Its native summary records 22 discovered capabilities, two reads, two governed mutations, two approval decisions, two durable result reads, two replays, one create, and one destroy as passed.

Its webhook summary records one accepted signed delivery, one rejected replay, one rejected forgery, and one observed central event as passed.

These summaries are retained historical evidence for their recorded workspace digest and artifacts. They do not map that workspace digest to reviewed commit `97be86e9efedf07ecf1783b03800f683f107fb04` and do not cover all gates above.

The reviewed source-owned five-product Compose topology also has no Twenty runtime service. Current exact-artifact qualification therefore remains unestablished until a complete run seals matching identities and cases.

Inspect and seal the evidence bundle

Require these groups before deciding the result:

- source revision, lockfile, dependency, runner, and clock identities;
- image inspect, history, SBOM, signature, provenance, and policy decisions;
- admission package, manifest, trust decision, and registry state;
- tenant, workspace, credential, provider, and fixture identities;
- test commands, outputs, exit codes, and per-case result ledger;
- signed native requests, approvals, responses, and verification reads;
- webhook requests, replay decisions, events, and outbox outcomes;
- fault injection, readiness, lease, storage, and recovery timelines;
- sanitized logs and explicit limitations;
- cleanup evidence and a cryptographic manifest of every retained file.

Validate that every indexed file exists. Keep restricted raw evidence separate from the publishable index.

Decide the result

Result	Rule
`PASS`	Every required gate passed for the frozen identities and every limitation is explicit
`FAIL`	A required case failed, identity drifted, evidence conflicts, or a secret escaped
`NOT RUN`	The case did not execute or lacks sufficient retained evidence

A bounded qualification may pass with explicit out-of-scope cases only when the claim excludes them. Missing evidence never becomes a passing result.

Publish artifact, manifest, provider, workspace, and topology identities with the executed case list, evidence-manifest digest, result, and limitations. Do not publish credentials or unredacted CRM content.

Clean up without losing proof

After sealing the evidence bundle:

1. remove synthetic records, metadata, views, layouts, and webhooks through governed cleanup;
2. verify cleanup with bounded reads;
3. revoke run-scoped credentials, assignments, and webhook secrets;
4. stop only the isolated qualification topology;
5. retain required state snapshots under evidence policy;
6. verify evidence hashes after transfer;
7. record cleanup status separately from the qualification result.

Cleanup failure is an operational failure. It does not rewrite the result of an already executed case.

Related documentation

- Twenty connector overview
- Twenty capability index
- Deploy the Twenty connector
- Twenty health, observability, and recovery
- Conformance and qualification