Train, test, and recover a CrewAI crew

Use this procedure to create one controlled CrewAI training artifact, evaluate the crew in a separate test Action, and retain enough evidence to accept or recover each outcome.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/crewai/guides/train-test-and-recover.md

Training and testing can invoke models, tools, and crew tasks many times. They are retry-unsafe provider operations even though their durable job events can be streamed.

The connector advertises cancellation for both. Their synchronous CrewAI methods remain non-cancellable once a worker thread starts.

Before you begin

Require:

- a discovered crew that explicitly admits `train` and `test`;
- exact Rust-host, sidecar, registry, CrewAI, crew-code, model, and tool

identities;

- a protected durable training directory owned by one sidecar;
- a reviewed training input set and independent evaluation input set;
- approved model, evaluator, tool, network, storage, and telemetry credentials;
- token, request, tool, rate, duration, and iteration budgets;
- an evidence store separate from ordinary application logs;
- authority to quarantine, preserve, promote, or remove generated artifacts.

Do not enable `train` merely to repair a readiness failure. Review the high-risk capability and its storage effects first.

1. Define the acceptance boundary

Write the training and evaluation claim before dispatch. Include:

Identity	Required record
AIP	Source revision, connector profile, host artifact, tenant, instance, and replica
Sidecar	Package, image digest, lockfile identity, state volume, and training directory
CrewAI	Default package version, method-verification source commit, and any source-overlay identity
Crew	Descriptor ID, registry revision, agents, tasks, process, models, and tools
Data	Training and evaluation dataset digests, access policy, and redaction class
Budget	Iterations, time, tokens, provider calls, tool calls, and monetary ceiling

Identity	Required record
Result	Artifact digest, evaluation evidence, pass rule, reviewer, and timestamp

Keep training and evaluation datasets distinguishable. Reusing the same examples can make a test look successful without measuring the intended generalization.

2. Stage the training directory

Configure `AIP_CREWAI_TRAINING_DIR` before sidecar startup. The directory must be a non-symlink and must not be group- or world-writable.

Use a connector-owned durable volume. Give exactly one active sidecar process write authority to that volume and its journal.

Inventory existing artifact basenames before selecting a new one. The connector validates a safe basename but does not prevent an intentional name from referring to an existing file.

Keep CrewAI's durable home on the same controlled state boundary. The default product image places `HOME` under the sidecar volume for its local credential- encryption key.

3. Freeze the training Action

Allocate one Action ID, key, and unique artifact name:

```
TEXT
Action ID:       act_crewai_train_support_017
Idempotency key: crewai-train-support-017-v1
Artifact name:  support-training-017.json
```

Build the input:

```
JSON
{
  "n_iterations": 4,
  "artifact_name": "support-training-017.json",
  "inputs": {
    "dataset_revision": "support-cases-2026-07-27",
    "policy_revision": "support-policy-12"
  }
}
```

Iterations must be between 1 and 100. The artifact name permits 1 through 128 letters, digits, dots, underscores, or hyphens and must begin with a letter or digit.

The input object is passed to every pinned CrewAI training kickoff. Do not put secret bytes or an untrusted filesystem path inside it.

4. Approve training as a bounded mutation

Review the immutable Action for:

1. exact crew and training dataset;
2. artifact name and overwrite disposition;
3. iteration and budget ceilings;
4. every model and tool authority reachable by crew tasks;
5. human-input handling required by pinned CrewAI training;
6. confidential-data and telemetry destinations;
7. rollback limitations and uncertain-outcome procedure.

Approve only that Action ID, key, and input hash. A prior test or ordinary run does not authorize training.

The connector persists its idempotency fence before scheduling CrewAI. A journal write failure therefore prevents provider admission; later failure can leave effects uncertain.

5. Observe the training job

Follow durable AIP chunks for the training Action. The sidecar normally emits `started` and a terminal event because the pinned training method runs in a worker thread with CrewAI streaming disabled.

Use separate status and event read Actions targeting the training Action ID:

```
JSON
{
  "run_action_id": "act_crewai_train_support_017"
}
```

Do not interpret the absence of token-level chunks as a stalled job. Use status, provider evidence, timeout policy, and resource observations together.

Do not request runtime cancellation as a normal control. It cannot stop the synchronous CrewAI training thread.

The sidecar can still persist `cancelled`, and Rust can return `remote_stop_confirmed: true`. That confirms only the sidecar route and must be treated as an uncertain provider outcome.

6. Inspect and seal the artifact

A completed train result contains the artifact basename and iteration count. It does not contain the file, a digest, or evaluation evidence.

After terminal completion:

1. verify that the resolved file remains under the configured training root;
2. record its owner, mode, size, modification time, and cryptographic digest;
3. scan content for credentials, prompts, personal data, and unintended

provider output;

4. copy the accepted bytes into versioned integrity-protected storage;
5. associate the copy with every identity and approval recorded in step 1;
6. leave the sidecar-owned source file untouched until retention policy allows

removal.

File presence alone does not prove successful training. Require the completed Action, terminal sidecar event, reviewed artifact, and source identities.

Quarantine an artifact when the Action failed, timed out, restarted, or lost its terminal evidence. Do not use it as an evaluator or deployment input until reconciled.

7. Freeze a separate test Action

Testing has its own Action ID, key, approval, inputs, and evaluator:

```
TEXT
Action ID:       act_crewai_test_support_017
Idempotency key: crewai-test-support-017-v1
Evaluator:       openrouter/reviewed-evaluator
```

Build the test input:

```
JSON
{
  "n_iterations": 6,
  "eval_llm": "openrouter/reviewed-evaluator",
  "inputs": {
    "dataset_revision": "support-evaluation-2026-07-27",
    "candidate_artifact": "support-training-017.json"
  }
}
```

`eval_llm` is a bounded CrewAI model selector, not an AIP credential handle. The deployment controls its provider credential and actual resolution.

Confirm that the crew factory intentionally loads the candidate artifact. The connector passes the test input to CrewAI but does not interpret a `candidate_artifact` convention.

8. Approve and observe testing

Review:

- evaluator model and provider authority;
- evaluation dataset separation and expected coverage;
- six complete crew executions in this example;
- all models and tools reachable by the evaluated crew;
- pass criteria and where CrewAI evaluator output will be retained;
- timeout and uncertain-outcome handling.

Follow the test Action and read status or events by its own ID. Do not target the training Action when checking test completion.

The completed connector result contains status, iteration count, and evaluator string. It does not return a structured scorecard.

9. Retain evaluator evidence

The pinned CrewAI method prints its evaluation result through CrewAI's output path. Configure an approved capture mechanism before starting the test.

Retain:

1. raw evaluator output in access-controlled evidence storage;
2. a redacted reviewer-facing summary;
3. evaluator model and credential revision;
4. training and evaluation data digests;
5. crew, registry, source, lockfile, and artifact identities;
6. iteration-level pass, failure, timeout, and tool-effect observations;
7. the declared acceptance rule and independent reviewer decision.

Do not infer a passing evaluation from the connector's `status: completed`. That value means the pinned CrewAI method returned without raising.

10. Classify each terminal state

State	Evidence decision
`completed` with matching terminal event	Inspect artifact or evaluator evidence before acceptance
`failed` before any `started` event	Verify whether provider admission occurred before classifying no effect
`failed` after `started`	Treat model, tool, file, and evaluator effects as uncertain
`cancelled` or local waiter ended	Treat underlying synchronous CrewAI work as potentially continuing
Restored `running` record becomes `failed`	Preserve `uncertain_outcome: true` and reconcile external state
No durable job under the expected ID	Check instance and journal ownership before another submission
Same Action repeats exactly	Observe the existing record; do not interpret it as a new run
Same Action conflicts	Restore the original input and key; never bypass the fence

Ordinary provider and timeout failures do not carry an `uncertain_outcome` field at this revision. Apply the table from execution evidence; only a restart-restored running record sets that field explicitly.

Provider or tool logs can contain prompts, inputs, URLs, outputs, and credentials. Use Action ID correlation and retain only approved redacted data.

11. Recover uncertain training

1. stop new training assignment to the affected crew instance;
2. preserve the Action, approval, journal, sidecar, registry, and volume

identities;

3. read durable status and all events for the original Action ID;
4. inspect the intended artifact without modifying or replacing it;
5. reconcile model, tool, storage, and human-input evidence;
6. classify the artifact as absent, complete, partial, corrupt, or unknown;
7. quarantine any bytes whose production path cannot be proved;
8. create a new artifact name and Action only after the earlier effect is

settled and policy authorizes a distinct attempt.

Never overwrite an uncertain artifact to make the directory look clean.

12. Recover uncertain testing

1. preserve the original test Action ID, key, evaluator, and input digest;
2. read durable status and events;
3. reconcile evaluator-provider requests and every crew model or tool effect;
4. determine whether each planned iteration started and produced evidence;
5. mark the campaign inconclusive when the required output is incomplete;
6. retain partial evidence without counting it as a passing result;
7. authorize a new test Action only under a new evidence identity.

A new test can be appropriate after reconciliation because evaluation is a new measurement. It is never an automatic retry of an unknown Action.

Completion checklist

- Training and evaluation claims were defined before dispatch.
- Source, artifact, crew, model, tool, data, and credential identities were retained.
- One active sidecar owned the journal and training directory.
- Training used a unique reviewed artifact basename.
- Train and test used separate Action IDs, keys, approvals, and inputs.
- A completed training artifact was inspected, hashed, and sealed.
- Evaluator output was retained separately from connector completion metadata.
- Timeout, cancellation, and restart states were treated as uncertain.
- No new mutation began before earlier effects were reconciled.

Related documentation

- CrewAI training and testing reference
- Run a CrewAI crew and stream events
- CrewAI connector configuration
- Approvals and policy
- Errors and retry decisions