CrewAI authentication and admitted crews

The CrewAI integration has three distinct trust boundaries. Keeping them separate prevents an AIP caller from selecting Python code, changing the sidecar identity, or supplying deployment credentials through an Action.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/crewai/getting-started/authentication-and-admitted-crews.md

This page owns that boundary model. It does not enumerate every configuration value or replace tenant admission, operation policy, and production rollout controls.

Separate the three trust boundaries

Boundary	Trusted input	Enforced by	Protected decision
Caller to AIP	Verified envelope, principal, membership, tenant, and route	AIP edge, runtime, and connector host	Which admitted capability may receive the Action
Rust host to Python sidecar	Fixed sidecar base URL and bearer token	`aip-host-crewai` and `aip-crewai-sidecar`	Which host may call the private execution API
Sidecar to CrewAI dependencies	Deployment-owned registry, crew code, models, tools, and provider secrets	Sidecar deployment and crew implementation	What trusted Python code and external authority execute

One credential cannot substitute for another. A caller token does not authenticate the Rust host to the sidecar. The sidecar bearer does not grant model, tool, memory, or knowledge access.

Keep AIP identity outside Action input

The signed AIP route selects an admitted standalone host. Verified execution context carries the tenant and host binding before the connector interprets the capability ID.

An Action may select only an operation already published for its capability. It cannot provide:

- a tenant, connector instance, replica, or registry route;
- a Python module, registry attribute, or crew ID outside the capability;
- a sidecar origin or bearer token;
- a host signing key, credential revision, or admission identity;
- a model, tool, or provider credential through connector metadata.

The connector uses the verified tenant when it builds execution and idempotency context. Untrusted body fields do not replace that value.

Protect the host-to-sidecar channel

The Rust host requires a fixed `AIP_CREWAI_SIDECAR_URL`. The reviewed source rejects credentials, a query, or a fragment.

It does not reject a path prefix. Internal joins use absolute paths such as `/health` and `/jobs`, which discard that prefix. Configure an origin-only URL so the deployed route matches the value an operator reviews.

HTTPS is required except for an explicit loopback origin. Redirects are disabled, so the host does not forward its bearer credential to a redirected destination.

Production startup also requires `AIP_CREWAI_SIDECAR_TOKEN_FILE`. The host loads the owner-held value and adds it as an HTTP Bearer credential to health, job, event, and cancellation requests.

The host accepts a regular owner-only file of 1 byte through 16 KiB, trims surrounding ASCII whitespace, and requires non-empty UTF-8. The connector then rejects an empty value or any control character.

The reviewed connector does not reject an interior plain space or apply a second explicit length check after file loading. Its debug representation reports only whether a token is configured.

Place the host and sidecar on a private network even when HTTPS and bearer authentication are present. The sidecar API can start deployment-authored Python code and must not be exposed as a tenant-facing endpoint.

Provision the sidecar token file

The Python sidecar accepts one of two mutually exclusive token sources:

```
TEXT
AIP_CREWAI_BEARER_TOKEN_FILE=/run/secrets/crewai-sidecar-token
AIP_CREWAI_BEARER_TOKEN=<inline-development-value>
```

Use the file form for a controlled deployment. The sidecar requires the path to be a regular non-symlink file with no group or world permission bits. It accepts between 1 byte and 16 KiB.

The sidecar opens the file without following a symlink when the platform supports that flag. It compares the opened device and inode with the inspected file, decodes UTF-8, and strips surrounding whitespace.

Every protected route compares the complete `Authorization` value in constant time. Missing or invalid material returns `401` with a Bearer challenge.

Do not place the token in:

- a crew descriptor, Action input, manifest, capability, or registry entry;
- a command-line argument, image layer, public configuration, or evidence file;
- model, tool, memory, or knowledge configuration;
- an AIP credential handle or external-account label.

The source permits unauthenticated operation only when the sidecar bind host is loopback and `AIP_CREWAI_ALLOW_UNAUTHENTICATED=true`. Both conditions are required. This exception is a local-development boundary, not a production deployment option.

Admit crews through host descriptors

`AIP_CREWAI_CREWS_FILE` is a bounded JSON array loaded before the host joins the fleet. The file must contain between one and 1,000 descriptors and may not exceed 4 MiB.

Each descriptor has this public shape:

```
JSON
{
  "id": "support-primary",
  "name": "Support primary",
  "description": "Routes reviewed support cases",
  "allowed_operations": ["run", "status", "events", "cancel"]
}
```

The host applies these checks:

Field	Constraint
`id`	Non-empty after trimming, no control characters, unique, and valid inside each derived AIP capability ID
`name`	Non-empty after trimming
`description`	Optional; no field-specific bound or character check at this revision
`allowed_operations`	Non-empty set of known CrewAI operations with valid derived capability IDs

The 4 MiB file ceiling bounds the combined descriptor payload. It is not a substitute for field-specific name and description policy. Apply tighter deployment validation before generating public manifest text.

A descriptor that omits `allowed_operations` receives the compatibility policy `run`, `status`, `events`, and `cancel`. New deployments should state the intended set explicitly so a source-default change cannot widen policy silently.

The descriptor admits a public crew identity and operation surface. It does not load Python code or prove that a provider credential has the intended scope.

Bind descriptors to a private registry

The sidecar loads `AIP_CREWAI_REGISTRY` using `module:attribute` syntax. The referenced attribute must be a non-empty mapping or a zero-argument callable that returns one.

Mapping keys are non-empty crew IDs. Each value must be a zero-argument factory or a cloneable Crew instance.

The sidecar calls the factory or creates a deep clone for each job. This prevents mutable CrewAI objects from being reused across requests by the registry layer. It does not sandbox the resulting Python code.

The registry may contain crews that the host does not advertise. A caller still cannot reach them because the Rust descriptor owns capability publication and routing.

Every host-admitted crew must exist in the registry. Authenticated sidecar health publishes its available crew IDs, and host readiness fails when an admitted ID is absent.

Align both operation allowlists

The Rust descriptor may admit any subset of the ten frozen operations. The sidecar environment admits only operations that can start jobs:

```
TEXT
run,batch_run,replay,train,test,knowledge_query,memory_reset
```

`status`, `events`, and `cancel` are control operations. The sidecar serves and reports them automatically instead of accepting them in `AIP_CREWAI_ALLOWED_OPERATIONS`.

The sidecar default admits only `run`. Host readiness compares each descriptor allowlist with the authenticated health response. A host that advertises an operation missing from sidecar health remains unready.

Use the intersection intentionally:

1. review the crew code and its model, tool, memory, and knowledge effects;
2. enable only the sidecar job operations required by that deployment;
3. give each host descriptor the same or a narrower operation policy;
4. admit the resulting capabilities for the intended tenant;
5. verify authenticated health before enabling traffic.

Never repair a readiness failure by enabling every operation without reviewing the added side effects. Training and memory reset have materially different risk from status and event reads.

Keep CrewAI credentials deployment-owned

Crew factories may obtain model, evaluator, tool, storage, tracing, memory, or knowledge credentials through deployment-controlled code and secret mounts. The connector does not resolve those credentials from an Action.

Its implementation support declares dynamic AIP credential handling as unsupported. Provider credentials therefore remain outside the connector's capability schema and discovery output.

Apply least authority to each dependency. A crew that only classifies local records should not inherit a credential that can mutate an unrelated provider. Treat tools as executable code with their own network and data boundaries.

Telemetry is disabled in the controlled fleet profile. Enabling CrewAI or OpenTelemetry tracing requires a separate review of content, credential, and destination exposure.

Isolate tenants and active processes

Use one connector instance for one reviewed tenant, descriptor set, registry, sidecar token, and durable sidecar state boundary. Deployment metadata may also record an external account, but the reviewed CrewAI host does not compare that label with a descriptor ID automatically.

Only one active sidecar process may own a state journal. Horizontal scaling uses independent connector instances with independent state volumes, tokens, replica identities, and routes.

Do not mount one journal into active replicas. The local file lock is a single-writer process fence, not a distributed coordinator.

Rotate without splitting the boundary

The host and sidecar load their authentication and admission inputs at startup. Treat rotation as a coordinated replacement rather than an undocumented live reload.

1. freeze new unsafe assignment to the affected instance;
2. inventory running jobs and preserve their Action IDs and journal owner;
3. create a new token file and, when needed, a reviewed descriptor or registry

revision;

4. start a replacement sidecar against a new or deliberately transferred

state boundary;

5. start its matching Rust host with the same new token and fixed origin;
6. require authenticated health, descriptor agreement, and expected discovery;
7. route new Actions only after those checks pass;
8. drain the old instance before revoking its token or removing its state;
9. retain redacted identity and transition evidence.

Never let an old host authenticate to a new sidecar unintentionally. Never delete a journal while a job outcome may still be uncertain.

Diagnose boundary failures

Symptom	Decision
Host rejects the token	Check file type, permissions, UTF-8, length, and prohibited characters without printing it
Sidecar returns `401`	Compare the mounted token sources and restart ownership; do not put the value in a diagnostic command
Sidecar refuses startup without a token	Configure the protected file, or bind loopback and enable the explicit development exception
Descriptor is rejected	Check file count and size, non-empty IDs and names, ID controls, uniqueness, capability syntax, and operation sets
Registry fails to load	Verify `module:attribute`, importability, mapping shape, IDs, and factory or clone behavior
Host is healthy but not ready	Compare admitted crew IDs and operations with authenticated sidecar health
A crew is present only in the registry	Add a reviewed host descriptor and admission policy before exposing it
Provider returns an authentication error	Investigate the crew-owned model, tool, or provider credential, not the sidecar token
Data appears across tenant boundaries	Disable routing and reconcile tenant, instance, descriptor, registry, state volume, and provider authority

Preserve Action ID, tenant, instance, replica, descriptor revision, sidecar artifact, registry identity, credential revision, and redacted error category. Do not retain token bytes or raw model and tool content by default.

Related documentation

- CrewAI connector overview
- Run the CrewAI quickstart
- Identity and trust
- Connector credentials and rotation
- Errors and retry decisions