CrewAI Connector

The CrewAI integration has two deployment components:

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/crewai.md

1. the Rust `aip-connector-crewai` crate, which publishes AIP capabilities and

maps actions, streams, cancellation, and errors;

2. the Python sidecar, which loads deployment-owned Crew factories and executes

one isolated CrewAI crew per AIP action.

This boundary keeps Python framework objects and provider dependencies out of the Rust semantic core.

Capability Mapping

Each configured crew becomes `cap:crewai:{crew_id}`. The capability supports synchronous invocation, asynchronous execution, ordered streaming, and cancellation. It requires AIP approval and marks inputs and results as confidential, PII-bearing, and redaction-required.

The connector does not advertise automatic retry, transactions, reconciliation, compensation, or frozen credential-handle routing.

Run the Sidecar

Create a deployment-owned registry module:

```
PYTHON
from my_support_project.crew import SupportCrew


def crews():
    return {"support": lambda: SupportCrew().crew()}
```

Then configure and start the sidecar:

```
SH
export AIP_CREWAI_REGISTRY=my_deployment.registry:crews
export AIP_CREWAI_BEARER_TOKEN="$(openssl rand -hex 32)"
export AIP_CREWAI_STATE_FILE=/var/lib/aip-crewai/runs.json
aip-crewai-sidecar
```

The service exposes authenticated `/health`, `/runs`, `/runs/stream`, `/runs/{action_id}`, and `/runs/{action_id}/cancel` endpoints. The default listener is `0.0.0.0:8090`.

The sidecar permits an omitted bearer token for local tests. Production startup must require `AIP_CREWAI_BEARER_TOKEN`, keep the listener on a private network, and use the same secret only at the Rust connector boundary.

Safety and Recovery

The sidecar uses the AIP action id as durable run identity. Reusing that id with different input returns HTTP 409. It bounds input, timeout, concurrency, SSE frames, and retained events, and closes the active CrewAI stream during cancellation.

When a persisted run was still active at restart, the sidecar marks it as an uncertain failure rather than silently starting the crew again. Mount the state journal on durable storage. Active-active sidecars require an external atomic coordinator for action-id fencing.

The supplied container configuration disables CrewAI and OpenTelemetry telemetry by default. A deployment that enables tracing must separately govern data processing, retention, and redaction.

Connect the Rust Boundary

Construct `CrewAiSidecarConnector` with the sidecar URL and a non-empty list of stable crew descriptors, then configure the same bearer token used by the sidecar. Register the generated capabilities and frozen handlers in the AIP gateway host.

The stock `aipd` 1.0 binary does not provide CrewAI sidecar CLI options. Host composition, registry loading, tenant isolation, and secret distribution are deployment responsibilities.

Qualification Boundary

The repository includes deterministic sidecar tests and narrow exact-upstream evidence using a real CrewAI crew with a deterministic network-free model. That is not a complete retained isolated-live production campaign.

Production promotion must additionally prove the exact CrewAI revision, registry code, provider credentials, stream and cancellation races, restart recovery, side effects, capacity, tenant isolation, and observability policy.

The implementation repository keeps the sidecar operations reference at `crates/aip-connector-crewai/sidecar/README.md`. See also Pinned Upstream Baselines and Conformance and Qualification.