CrewAI sidecar contract and lockfile

This reference identifies the Python artifact behind the CrewAI connector and the private HTTP contract used by its Rust host. Use it to compare image provenance, request shapes, durable records, and failure responses.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/crewai/reference/sidecar-contract-and-lockfile.md

The sidecar is not a tenant-facing AIP API. AIP identity, admission, policy, approval, routing, and signed results remain owned by the Rust host and shared runtime.

Fix the reviewed identity

Property	Reviewed value
AIP source	`97be86e9efedf07ecf1783b03800f683f107fb04`
Sidecar package	`aip-crewai-sidecar 1.0.0`
Python range	`>=3.10,<3.14`
Default CrewAI package set	`crewai`, `crewai-cli`, and `crewai-core 1.15.5`
Method-verification source	`bfa652a7be8637562cc9b0833f75d927a64552d1`
Method-source package version	`1.15.2`
Default listener	`0.0.0.0:8090`
Application identity	FastAPI title `AIP CrewAI Sidecar`, version `1.0.0`

The operation catalogue records the method-verification source revision. That revision is not the package identity of the default product sidecar.

Choose the artifact definition deliberately

The reviewed tree contains two image definitions with different identities.

Definition	Dependency resolution	CrewAI installation	Selected by product Compose
`Dockerfile`	Two frozen `uv sync` steps	Complete lockfile set at `1.15.5`	Yes
`Dockerfile.upstream`	Frozen lockfile first	Reinstalls only `lib/crewai` from a supplied source stage, without dependencies	No

The default image labels CrewAI version `1.15.5`. It also places `CREWAI_STORAGE_DIR` and `HOME` under `/var/lib/aip-crewai` so journal and CrewAI-local state share one durable volume.

The source-overlay definition accepts a `CREWAI_SOURCE_REVISION` build argument and copies an external stage named `crewai-source`. It verifies only that the argument is non-empty; it does not compare the argument with copied bytes.

At the reviewed CrewAI source commit, the reinstalled distribution reports `1.15.2`. The overlay does not reinstall `crewai-core` or `crewai-cli`, so those dependencies remain at lockfile version `1.15.5`.

The source-overlay definition also lacks the default image's durable `HOME` and `CREWAI_STORAGE_DIR` declarations. Configure equivalent paths explicitly before using that artifact with persistent credentials or storage.

Bind the source files and lockfile

These SHA-256 values are calculated from each blob at the reviewed AIP commit.

Source blob	SHA-256
`sidecar/pyproject.toml`	`d0b1278261efa26f9bc8905040dcf00066b19f7bd64beca44d206777572b32a0`
`sidecar/uv.lock`	`946f033e72f52d23172d9e1d9179f1057ad8fb2975a2d79d76d1e6bb469baa02`
`sidecar/Dockerfile`	`2c5e874b424d1eb6e0eea1204f6475d06088233c940b36361feb3a7591b6c388`
`sidecar/Dockerfile.upstream`	`1c8ab9e9835dce1aa5e6cbe50c7522732bfb48b321f8bf5031d1378c9d28e61e`
`sidecar/aip_crewai_sidecar/app.py`	`705046d4662fee1f1fa5b678ebfed6c7f9e88e1c6e8e69a502e78f23fa151998`
`sidecar/aip_crewai_sidecar/registry.py`	`a6d9444e9a89812beeb1c3f1af206d470ae056d67d2d34485df1e7413c8917fb`

An image digest and retained software bill of materials remain the deployment identity. Source hashes identify reviewed inputs; they do not prove which bytes a registry image contains.

Keep the HTTP boundary private

Every route requires the complete Bearer header unless the process binds to loopback and explicitly enables unauthenticated development mode. The sidecar compares the expected header in constant time.

Uvicorn disables proxy-header trust and omits its server header. These settings do not supply TLS, AIP identity, tenant routing, or admission. Deploy the sidecar behind the fixed private route used by `aip-host-crewai`.

The Rust host disables redirects and uses HTTPS except for explicit loopback HTTP. Do not expose the sidecar port through the public AIP edge.

Read the start and cancel models

Every startable operation uses this closed request envelope:

Field	Required	Constraint
`crew_id`	Yes	String from 1 through 256 characters
`action_id`	Yes	String from 1 through 256 characters
`operation`	No	Frozen operation enum; default `run`
`input`	No	JSON object; default empty object; encoded value at most 1 MiB
`timeout_ms`	No	Integer from 1 through `3600000`; default `600000`

Operation-specific validation then applies exact fields and bounds. The sidecar rejects unknown request-envelope fields.

Cancellation uses a separate closed envelope:

Field	Required	Constraint
`crew_id`	Yes	String from 1 through 256 characters
`action_id`	Yes	String from 1 through 256 characters; must equal the path ID
`reason`	No	String of at most 2,048 characters

The request model does not reject control characters in `reason` at this revision.

Use only the fixed private routes

Method	Route	Purpose
`GET`	`/health`	Read registry, policy, journal, training, and capacity status
`POST`	`/jobs`	Start or observe one blocking durable job
`POST`	`/jobs/stream`	Start or observe one job as SSE
`GET`	`/jobs/{action_id}`	Read one durable job record
`GET`	`/jobs/{action_id}/events`	Read a JSON slice or follow SSE from a cursor
`POST`	`/jobs/{action_id}/cancel`	Cancel the sidecar task for one job
`DELETE`	`/jobs/{action_id}`	Archive one terminal durable record

Legacy aliases remain for `POST /runs`, `POST /runs/stream`, `GET /runs/{action_id}`, and `POST /runs/{action_id}/cancel`. The two start aliases accept only the `run` operation.

There is no legacy events or archive alias. AIP callers must not call either route family directly.

Supply the required headers

Header	Applies to	Behavior
`Authorization: Bearer ...`	Every protected route	Exact sidecar credential
`Idempotency-Key`	Every startable job	Required, non-empty, at most 512 UTF-8 bytes, no code point below U+0020
`Accept: text/event-stream`	Job events read	Selects streaming instead of the JSON slice
`X-AIP-Archive-Confirmation`	Archive	Must equal the archived Action ID

The Rust connector also sends `X-AIP-Action-ID` on job starts and explicit cancel Actions. The reviewed FastAPI handlers do not read that header; the body and path IDs remain authoritative inside the sidecar.

Interpret records and event streams

A durable record contains Action ID, crew ID, operation, request hash, status, events, and optional output. Raw operation input and idempotency-key bytes are not persisted, but they contribute to the request hash.

Events have this stored form:

```
JSON
{
  "event": "completed",
  "sequence": 3,
  "data": {
    "status": "completed"
  }
}
```

SSE serializes the same record under its event name:

```
TEXT
event: completed
data: {"event":"completed","sequence":3,"data":{"status":"completed"}}
```

One job status is `running`, `completed`, `failed`, or `cancelled`. A blocking route returns only after terminal state or its HTTP wait timeout. The timeout does not prove that the job or worker thread stopped.

Stored events and output can contain model, tool, knowledge, or provider data. Treat the journal as confidential even though credential and raw-input bytes are absent.

Interpret HTTP failures

Status	Source-owned meaning
`400`	Legacy start used a non-run operation, or cancel path and body IDs differ
`401`	Missing or invalid sidecar Bearer credential
`403`	Startable operation is disabled by sidecar policy
`404`	Crew or durable job is absent
`409`	Action ID conflicts with different material, or a running job is archived
`413`	Encoded operation input exceeds 1 MiB
`416`	Event cursor is negative or beyond the retained event count
`422`	Request or operation-specific validation failed
`428`	Idempotency key or archive confirmation is missing
`504`	Blocking HTTP wait exceeded the requested operation timeout
`507`	Journal capacity or durable persistence prevented admission

An ordinary provider failure becomes a terminal `failed` event with exception class and a generic message. It has no `uncertain_outcome` field at this revision.

The sidecar logs such failures with `LOGGER.exception`. Restrict and redact sidecar logs because provider exception text and traceback may contain governed content or credentials.

Verify an artifact before deployment

1. identify which Dockerfile definition produced the image;
2. verify the exact AIP commit and the relevant source-blob hashes;
3. retain the immutable image digest, labels, build provenance, and SBOM;
4. inspect installed `crewai`, `crewai-core`, and `crewai-cli` distributions;
5. verify durable `HOME`, CrewAI storage, journal, and training paths;
6. compare authenticated health with the admitted descriptor set;
7. qualify route, timeout, restart, cancellation, and archive behavior;
8. record every mismatch between published support and provider behavior.

Do not infer a source-overlay build from the revision label alone. Do not infer provider termination from a successful cancellation response.

Related documentation

- CrewAI connector overview
- CrewAI connector configuration

- CrewAI authentication and admitted crews
- CrewAI capability index
- Errors and retry decisions