Connector fleet HTTP API

This reference describes the HTTP contracts used inside an AIP connector fleet. Use it to identify the correct process and route, construct a signed request, and interpret route, lease, replay, and capacity failures.

SECTION	API Reference
TYPE	Reference
PROTOCOL	AIP 1.0
SOURCE	docs/reference/connector-fleet-api.md

The contract applies to source revision `97be86e9efedf07ecf1783b03800f683f107fb04`. It is an implementation boundary between trusted fleet components. It is not a public connector-catalog administration API and does not add fields to AIP 1.0.

Three HTTP planes

Plane	Caller	Server	Purpose
Connector data	Central `aipd`	Connector host	Deliver a route-pinned action or cancellation and receive a signed result
Host lifecycle	Connector host	Connector control plane	Register, renew, drain, or take one pre-provisioned replica offline
Central ingress	Connector host	Central `aipd`	Publish provider events or return stream chunks for an assigned action

Each plane has its own signer and authorization checks. Bearer authentication for the public native API does not replace these signatures. Operational health, readiness, manifest, and metrics routes have no application authentication in their process routers; protect them at the deployment edge.

Connector-host data plane

One connector-host process exposes five routes:

Method	Path	Response
`GET`	`/health`	Process-liveness JSON
`GET`	`/ready`	Lease, drain, storage, and connector readiness JSON
`GET`	`/metrics`	Connector-host Prometheus text
`GET`	`/aip/v1/manifest`	The immutable host-bound `Manifest`
`POST`	`/aip/v1/messages`	A signed response `Envelope`

The admitted public endpoint must be an absolute URL ending exactly in `/aip/v1/messages`, without credentials, query, or fragment. HTTPS is required except for explicitly enabled loopback development. The default request-body limit is 4 MiB.

Health and readiness

`GET /health` reports process liveness only:

```
JSON
{
  "status": "ok",
  "protocol": "AIP",
  "version": "1.0",
  "service": "aip-connector-host"
}
```

`GET /ready` returns HTTP `200` only when all four conditions are true:

- the host is not draining;
- its registry lease has not expired;
- durable runtime storage passes its probe; and
- the connector-specific health probe is ready.

Otherwise it returns HTTP `503`. The body exposes `status`, `lease_valid`, `draining`, a `runtime` object with durability, readiness, detail, and recovery state, and a `connector` readiness object. Readiness does not prove that an external provider request succeeded.

Message authentication and route pinning

`POST /aip/v1/messages` accepts a complete AIP `Envelope`. Before dispatch, the host verifies all of these boundaries:

1. The JSON passes the `Envelope` schema and decodes successfully.
2. The Ed25519 envelope signature resolves to the configured central gateway

DID.

3. `from` matches the configured gateway principal, kind, and trust domain.
4. `to` matches this host principal.
5. `security.connector_route` matches the admitted tenant, instance, replica,

version, and manifest digest.

6. An action or cancellation matches the route's action and capability.
7. Any credential revision and approval authorization remains valid locally.

The central dispatcher places these fields in `security.connector_route`:

Field	Meaning
`action_id`, `capability_id`	Durable work coordinates
`tenant_id`	Verified tenant selected by the central runtime
`instance_id`, `replica_id`, `version_id`	Pinned fleet destination
`manifest_digest`	Admitted immutable capability contract
`catalog_revision`, `binding_policy_revision`	Catalog and binding snapshot
`credential_revision_ref`, `quota_policy_ref`	Pinned policy references
`replica_health_revision`, `fence_token`	Assignment and lease fences
`original_mode`	Client mode before the remote-hop projection
`approval_authorization`	Optional terminal approval record verified by the gateway

The deployed remote dispatcher sends actions and action-scoped cancellations. It projects verified tenant identity into the action. An async client action is executed synchronously at the host hop because the central runtime owns durable queueing. A streaming action retains streaming mode and receives only the fixed central callback destination.

The host signs every post-decode success or protocol-error envelope with its configured host key. It also correlates the response to the request. The central dispatcher authenticates the endpoint and verifies the expected host principal, DID, and trust domain before accepting the response.

Host message errors

HTTP	Code	Condition
`400`	`schema.envelope.invalid`	JSON does not pass the envelope schema
`400`	`envelope.decode.invalid`	Schema-valid JSON cannot decode as an envelope
`401`	`connector_host.route_mismatch`	Gateway signature or sender is not trusted
`403`	`connector_host.route_mismatch`	Recipient or pinned route does not match this host
`503`	`connector_host.not_ready`	Host is draining or its lease or health is invalid
`429`	`connector_host.capacity`	Local action concurrency is exhausted

Schema and decode errors occur before a trustworthy request envelope exists, so those responses are unsigned input errors. Later errors are correlated and signed. Credential or approval-journal checks can add a more specific `403` or `503` error.

Host lifecycle control plane

The standalone connector control plane exposes:

Method	Path	Purpose
`GET`	`/health`	Process liveness
`GET`	`/ready`	Registry schema, revision, and pool readiness
`GET`	`/metrics`	Fixed-cardinality registry pool snapshot as JSON
`POST`	`/aip/v1/connector-control`	Signed replica lifecycle transition

`GET /ready` returns HTTP `200` with `schema_version`, `catalog_revision`, and `pools` only when the restricted registry connection is healthy and the installed schema is current. It returns `503` with `schema_ok` and `catalog_ok` otherwise. Unlike connector-host metrics, control-plane `/metrics` is a JSON pool snapshot, not Prometheus text.

Signed lifecycle request

The request body has a signed outer wrapper:

```
JSON
{
  "request": {
    "request_id": "msg_0190c42f2d5a70008000000000000001",
    "issued_at": "2026-07-26T12:00:00Z",
    "expires_at": "2026-07-26T12:00:30Z",
    "signer_did": "did:key:z6Mk...",
    "command": {
      "operation": "heartbeat",
      "payload": {
        "replica_id": "crepl_booking_region_a_one",
        "previous_sequence": 17,
        "in_flight": 3,
        "connector_ready": true
      }
    }
```

```
  },
  "signature": "base64-ed25519-signature"
}
```

`signature` is standard-base64 Ed25519 over canonical JSON for the `request` object. `signer_did` must equal the DID pre-provisioned for that replica. The control plane rejects an invalid signature, an unknown replica, or a signer that does not match the registry record.

The accepted clock window is deliberately small. The request lifetime cannot exceed 30 seconds, `expires_at` must be after `issued_at` and the current time, and `issued_at` can be at most five seconds in the future.

Lifecycle commands

Operation	Payload	Successful outcome
`register`	`ConnectorHostRegistration`	New `lease`
`heartbeat`	`replica_id`, `previous_sequence`, `in_flight`, `connector_ready`	Renewed `lease`
`drain`	`replica_id`, `previous_sequence`	Renewed lease in draining state
`offline`	`replica_id`, `previous_sequence`	`offline`

Registration contains the complete proposed replica record plus `manifest_digest`, `artifact_digest`, `tenant_id`, `config_revision`, `secret_provider_ref`, and `capability_count`. The replica record carries its endpoint, peer principal and DID, trust domain, transport profile, topology, capacity, and initial health state. The control plane compares this proposal with the pre-admitted version, instance, and offline replica identity.

`previous_sequence` is a monotonic lease fence. A heartbeat cannot report more in-flight work than admitted capacity. `connector_ready=false` moves the replica offline instead of renewing it as ready. Drain stops new assignments while allowing already pinned work to finish.

Signed lifecycle response

Every normal or rejected operation returns a signed wrapper containing `response` and `signature`. The response repeats `request_id`, has its own 30-second `issued_at` and `expires_at` window, identifies the control-plane `signer_did`, and carries one outcome:

HTTP	Outcome	Meaning
`200`	`{"status":"lease","lease":{"sequence":n,"expires_at":[...]}}`	Register, heartbeat, or drain committed
`200`	`{"status":"offline"}`	Offline transition committed
`403`	`{"status":"rejected","code":"connector_control.rejected",...}`	Admission or configuration rejection; not retryable
`503`	`{"status":"rejected","code":"connector_control.unavailable",...}`	Storage or control service unavailable; retryable

The host verifies the response request ID, trusted control-plane DID, time window, and signature before applying a lease. A response-signing failure is an unsigned HTTP `500` input-style error because no signed response can be produced.

The wrapper's `issued_at` and `expires_at` fields are RFC 3339 strings. At this revision, `ConnectorHostLease.expires_at` uses the default `time` serialization instead: a nine-item array of year, ordinal day, hour, minute, second, nanosecond, and three UTC-offset components. Clients must not assume the lease field is also an RFC 3339 string.

Request IDs are replay protected. Retrying the exact signed bytes after a lost response is safe while that request remains the last committed lifecycle transition. Reusing the ID with different content, or replaying an older transition after a later state, is rejected.

The control endpoint must be an absolute HTTPS URL ending exactly in `/aip/v1/connector-control`, with no credentials, query, or fragment. Explicit loopback development may use HTTP. Request and response bodies are limited to 64 KiB.

Central connector ingress

Central `aipd` exposes two canonical connector-origin routes:

Method	Path	Canonical body
`POST`	`/aip/v1/connector-events`	`EventStream` envelope for provider events
`POST`	`/aip/v1/connector-callbacks`	`StreamChunk` envelope for a streaming action

The connector host requires those exact paths when configuring its event and callback publishers. At this revision both routes mount the same shared handler, which selects behavior from the envelope body rather than the URL. Publishers should still use the canonical path for the body family.

Neither route uses native bearer middleware. The ingress validates the AIP envelope schema, exact central recipient, envelope signature, sender principal, registry state, tenant, version, manifest, attestation, and an active replica lease before storing content. Accepted responses are signed central envelopes correlated to the request.

Provider-event envelope

An event publisher sends a non-empty `EventStream` with no `next_cursor` and places these fields in `security.connector_event`:

Field	Meaning
`tenant_id`	Tenant fixed by the enabled connector instance
`instance_id`, `replica_id`, `version_id`	Originating fleet coordinates
`manifest_digest`	Active admitted version digest
`lease_sequence`	Exact current replica health revision
`channel_id`	Channel declared by the version manifest

The replica may be ready or draining, but its lease must remain current. The ingress binds each event actor to the signed host, writes verified tenant and connector provenance into event data, and then appends it to the durable event log. Reusing an event ID is accepted only when retained semantic content is equivalent; different content returns `connector_event.idempotency_conflict`.

Stream-callback envelope

A callback publisher sends one `StreamChunk`. Its `security.connector_callback` object contains the complete assigned route:

Field group	Fields
Work	`action_id`, `capability_id`, `tenant_id`
Destination	`instance_id`, `replica_id`, `version_id`, `manifest_digest`
Policy	`catalog_revision`, `binding_policy_revision`, `credential_revision_ref`, `quota_policy_ref`
Fences	`replica_health_revision`, `fence_token`
Mode	`original_mode` equal to `streaming`

The chunk action ID must equal the route action ID. All durable assignment fields must match the central route resolver. The current replica must still match the assigned endpoint, identity, DID, trust domain, transport profile, instance, and version. A lease renewal may advance the live health revision, but it cannot invalidate the

assignment's other fences.

Ingress defaults and responses

Bound	Default
Signed envelope	4 MiB
Events per event envelope	100
One enriched event	256 KiB
One stream chunk	256 KiB
Concurrent ingress operations	128
Maximum event age	24 hours
Maximum future clock skew	5 minutes
Maximum stream-callback age	5 minutes

Successful event ingress returns a signed `EventStream` containing the stored events. Successful stream ingress returns the signed `EventStream` produced by the central stream sink. The principal error cases are:

HTTP	Code	Retry
`404`	`connector_event.not_configured`	No
`400`	`schema.envelope.invalid` or `envelope.decode.invalid`	No
`400`	`connector_event.invalid` or `connector_event.idempotency_conflict`	No
`401`	`connector_event.not_authorized`	No
`503`	`connector_event.capacity`	Yes, after 100 ms
`503`	`connector_event.unavailable`	Yes, after 1,000 ms

An error produced after envelope decoding is signed when the central ingress signer is available. Pre-decode and not-configured errors use the ordinary native error response.

Request-flow summary

Flow	Sequence
Start or renew a host	Host signs lifecycle request → control plane verifies pre-provisioned identity and replay → registry transition commits → control plane signs lease → host verifies and applies lease
Invoke an action	Runtime resolves and fences route → `aipd` signs pinned envelope → host verifies gateway, route, lease, credential, and approval → connector executes → host signs result → central dispatcher verifies it
Publish a provider event	Host durably stages event → host signs event stream and route → central ingress verifies current registry state → event log accepts or identifies an idempotent replay
Stream a result	Central action carries fixed callback route → host signs each chunk → ingress compares the complete durable assignment → runtime stores and exposes the stream

Surface boundaries

This API does not expose catalog writes, connector admission, credential material, or registry database access to a connector host. Client-facing actions, queries, sessions, approvals, and audit routes remain in the native HTTP API. Complete deployment settings belong to connector-host configuration, and operational transitions belong to connector-host lifecycle.

Transport retry, destination, and response-authentication rules are detailed in transport bindings. Normative envelope and body definitions remain in JSON schemas.

Related documentation

- Deploy a connector fleet
- Errors and retry decisions
- Identity and trust