Native HTTP API

This reference lists the native HTTP surface implemented by the product-neutral getaip-server gateway. Use it to select a route, identify its application authentication boundary, construct query parameters, and interpret the current respons

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

The route inventory applies to source revision `d7cce13d1d555644d04a4d73c66c95b113737635`. It describes an implementation binding, not additional AIP 1.0 wire requirements. Normative message fields remain owned by the protocol schemas.

Route and authentication classes

The router has four native security classes:

Class	Routes	Authentication in `getaip-server`
Service and public discovery	`/`, `/health`, `/ready`, `/metrics`, `/aip/v1/health`, `/aip/v1/manifest`	No application middleware; protect operational data at the deployment edge
Generic native envelope	`POST /aip/v1/messages`	Configured bearer identity when presented; otherwise native envelope signature policy; loopback insecure development only when explicitly enabled
Ergonomic native API	Actions, events, sessions, approvals, transactions, deliveries, receipts, audit, resources, capabilities, and WebSocket	Native bearer middleware, or the explicit loopback development identity
Connector ingress	`POST /aip/v1/connector-events` and `POST /aip/v1/connector-callbacks`	Signed connector route, sender, tenant, lease, and replay validation inside the ingress handler

The first class is unprotected only inside the application router. A reverse proxy or network policy should restrict readiness and metrics according to the deployment boundary. Public manifest exposure is a deployment decision.

The connector-ingress routes are not client action routes. Their detailed contract belongs to the connector-fleet API.

Authentication and trusted context

The direct `getaip-server` native HTTP edge supports one configured static bearer credential. A successful comparison establishes the configured principal, scopes, and optional tenant as server-owned context. The gateway replaces payload sender claims with that authenticated principal.

All ergonomic routes pass through the native-auth middleware. With bearer authentication configured, a missing or invalid credential returns HTTP `401` with error code `auth.native_http_unauthorized`. An explicit loopback development mode can substitute the daemon identity; it is not a production authentication mode.

`POST /aip/v1/messages` is different. With a valid bearer credential, it uses the trusted HTTP-edge identity and enforces replay policy. Without a bearer identity, it sends the envelope through native signature verification, trusted signer binding, and replay policy. The default gateway policy requires a trusted signed envelope on that path.

A `tenant_id`, principal filter, or sender field supplied by the client does not grant authority. Runtime query authorization still checks the established principal, scopes, object owner, and tenant. See Identity and trust for that authorization model.

Representations and response forms

The native envelope media type is `application/aip+json`. The current Axum handlers use JSON extraction and JSON responses; ordinary JSON responses carry `application/json`. Follow-mode event responses carry `text/event-stream`.

Timestamps in native records use RFC 3339. Identifiers are opaque. Clients should validate their documented prefix and length rules but must not derive time, tenant, or topology from their contents.

Response form depends on the route:

Route class	Success	Error
`POST /aip/v1/messages` and `POST /aip/v1/actions`	Complete AIP response `Envelope`	Complete `Envelope` with an `error` body
Ergonomic native queries and operations	Native body object such as `ActionStatus` or `ApprovalList`	Complete `Envelope` with `ProtocolError`
Tenant capability catalog	`CapabilityPage`	Direct `{"error": {"code": ..., "message": ...}}` object
Health, readiness, metrics, and manifest	Route-specific JSON or Prometheus text	Route-specific response
SSE follow	Named SSE events with JSON data and cursor IDs	One terminal SSE error event

Service and manifest routes

Method	Path	Response
`GET`	`/`	Process liveness payload
`GET`	`/health`	Process liveness payload
`GET`	`/ready`	Core readiness; HTTP `200` or `503`
`GET`	`/metrics`	Prometheus text exposition
`GET`	`/aip/v1/health`	Versioned alias of process liveness
`GET`	`/aip/v1/manifest`	Complete or filtered participant `Manifest`
`POST`	`/aip/v1/messages`	Response envelope for any supported native message body

`GET /aip/v1/manifest` accepts:

Parameter	Form
`capability_id`	Comma-separated capability IDs
`resource_kind`	Comma-separated resource kinds
`profile`	Comma-separated profile IDs
`risk`	Comma-separated risk values
`side_effect`	Comma-separated side-effect values
`requires_approval`	Boolean
`supports_streaming`	Boolean
`supports_transactions`	Boolean

An empty filter returns the complete local manifest. This route does not include the tenant connector-fleet catalog.

Action routes

Method	Path	Purpose
`POST`	`/aip/v1/actions`	Submit an envelope through the action-oriented alias
`GET`	`/aip/v1/actions`	List visible durable action views
`GET`	`/aip/v1/actions/{action_id}`	Read one `ActionStatus`
`GET`	`/aip/v1/actions/{action_id}/result`	Read the terminal `ActionResult`
`GET`	`/aip/v1/actions/{action_id}/events`	Read or follow action events
`GET`	`/aip/v1/actions/{action_id}/chunks`	Read or follow stream chunks
`POST`	`/aip/v1/actions/{action_id}/cancel`	Request cancellation

Both submission paths call the same envelope handler at this revision. `POST /aip/v1/actions` is the action-oriented, bearer-authenticated alias, but the handler does not reject another supported body type solely because of that path. Use `/aip/v1/messages` for non-action message families.

List actions

`GET /aip/v1/actions` accepts:

Parameter	Meaning
`state`	Stable lifecycle state
`capability_id`	Exact capability ID
`session_id`	Exact session ID
`principal_id`	Requesting principal recorded for the queued action
`tenant_id`	Exact tenant boundary
`approval_id`	Linked approval
`transaction_id`	Linked transaction
`cursor`	Page cursor returned by this route
`limit`	Requested page size
`include_results`	Include available final results
`include_receipts`	Include available receipt chains

The response is `ActionList` with `actions`, `total_size`, and an optional next `cursor`.

Read status or result

The status route accepts `tenant_id`, `include_result`, `include_receipts`, `include_chunks`, and `wait_ms`. The result route accepts `tenant_id`, `wait_ms`, `include_receipt`, and `include_terminal_events`.

`wait_ms` is serialized into the current request object, but the reviewed runtime does not wait. Status returns the current view immediately. Result returns the current terminal result or an `action.result_not_ready` error. Clients that need waiting should follow events or poll with their own bounded deadline.

Read events or chunks

The action-events route accepts:

Parameter	Meaning
`tenant_id`	Expected tenant boundary
`cursor`	Opaque event cursor

Parameter	Meaning
`limit`	Requested page size
`kind`	One event kind
`kinds`	Comma-separated event kinds
`include_chunks`	Include durable stream chunks
`follow`	Return an SSE follow stream

When `cursor` is absent, `Last-Event-ID` supplies the cursor. The chunks route uses `tenant_id`, `cursor`, `limit`, and `follow`, and always selects `aip.stream.chunk` with chunks included.

Cancellation accepts an optional JSON object:

```
JSON
{
  "reason": "The caller no longer needs the result"
}
```

A successful cancellation response does not prove that an external provider effect was reversed.

Global event route

Method	Path	Purpose
`GET`	`/aip/v1/events`	Read or follow events visible to the authenticated identity

Used parameters are `cursor`, `limit`, `kind`, comma-separated `kinds`, and `follow`. A query cursor takes precedence over `Last-Event-ID`. Authorization limits the returned principals and tenants; the global route does not use a caller-supplied tenant filter.

Session routes

Method	Path	Purpose
`GET`	`/aip/v1/sessions`	List visible sessions
`GET`	`/aip/v1/sessions/{session_id}`	Read one session
`POST`	`/aip/v1/sessions/{session_id}/close`	Close a session
`POST`	`/aip/v1/sessions/{session_id}/resume`	Resume a session
`POST`	`/aip/v1/sessions/{session_id}:close`	Suffix-form close alias
`POST`	`/aip/v1/sessions/{session_id}:resume`	Suffix-form resume alias

The list route accepts `principal_id`, `status`, `cursor`, and `limit`. Close accepts an optional `reason`. Resume accepts `resume_token` and `last_event_cursor`. A successful resume rotates the token; replace the supplied token with the returned one. Actions and sessions owns the lifecycle rules.

Approval routes

Method	Path	Purpose
`GET`	`/aip/v1/approvals`	List visible approval records
`GET`	`/aip/v1/approvals/{approval_id}`	Read one approval record

List filters are `status`, `approver`, `requester`, `tenant_id`, `cursor`, and `limit`. `include_action_status` and `include_receipts` add their corresponding views.

The one-record route accepts `tenant_id`, `include_action_status`, `include_receipts`, and `include_evidence_payload`. Evidence payload expansion uses export authorization and requires the `approval:sensitive` scope in addition to approval access.

Submit `ApprovalRequest` and `ApprovalDecision` bodies through `POST /aip/v1/messages`. There is no dedicated approval mutation route.

Transaction and callback-delivery routes

Method	Path	Purpose
`GET`	`/aip/v1/transactions/{transaction_id}`	Read by transaction ID
`GET`	`/aip/v1/transactions/by-plan/{plan_id}`	Read by plan ID
`GET`	`/aip/v1/transactions/by-action/{action_id}`	Read by action ID
`GET`	`/aip/v1/callback-deliveries`	List durable callback deliveries
`GET`	`/aip/v1/callback-deliveries/{delivery_id}`	Read one delivery

All transaction routes accept `tenant_id`, `include_result`, and `include_receipts`.

The callback list accepts `action_id`, `status`, `profile`, `target`, `tenant_id`, `cursor`, `limit`, and `include_receipts`. The one-record route accepts `tenant_id` and `include_receipts`.

These routes are read-only. This revision has no public callback replay route and no dedicated remote-delegation outbox route.

Receipt and audit routes

Method	Path	Purpose
`GET`	`/aip/v1/receipts/{chain_id}`	Read one receipt chain
`GET`	`/aip/v1/receipts/by-receipt/{receipt_id}`	Find the chain containing a receipt
`GET`	`/aip/v1/audit/events`	Query visible audit events

Audit filters are `action_id`, `session_id`, `principal_id`, `tenant_id`, `transaction_id`, RFC 3339 `from` and `to`, `cursor`, and `limit`. `include_receipts=true` and `export=true` both use an export-class authorization check. Export also forces receipt inclusion.

Ordinary audit reads may be narrowed to the authenticated principal when the identity lacks a cross-principal read or export scope.

Resource routes

Method	Path	Purpose
`GET`	`/aip/v1/resources`	List visible resources
`GET`	`/aip/v1/resources/{resource_id}`	Read one resource

List filters are `capability_id`, `kind`, `tenant_id`, `cursor`, and `limit`. A one-resource read accepts `tenant_id`, `version`, and a comma-separated `accept` preference.

Tenant capability catalog

Method	Path	Purpose
`GET`	`/aip/v1/capabilities`	Search connector capabilities visible to the configured tenant

Filters are `capability_id`, case-insensitive `text`, `profile`, `cursor`, and `limit`. The response contains `catalog_revision`, `capabilities`, `next_cursor`, and `total`.

The route requires both native bearer authentication and a tenant configured for that bearer identity. It returns:

Status	Code	Condition
`404`	`connector_catalog.disabled`	No fleet catalog provider is configured
`403`	`connector_catalog.tenant_required`	The bearer identity has no configured tenant
`400`	`connector_catalog.invalid_query`	A typed filter is invalid
`409`	`connector_catalog.stale_cursor`	The catalog revision changed during pagination
`503`	`connector_catalog.unavailable`	The registry query failed

The default requested size is 100. The reviewed registry clamps the requested row limit between 1 and 200 and caps one page at 4 MiB of canonical capability definitions. A valid query can still return zero matches. A stale cursor must restart from the first page.

Pagination, expansions, and cursors

Durable action, session, approval, callback, resource, audit, and event lists default to 100 rows. The runtime clamps their requested row count between 1 and 1,000 rather than rejecting an out-of-range value. Event pages are also bounded to 4 MiB.

Cursors are route-specific continuation tokens. Echo only a cursor returned by the same operation and identity context. Do not parse or manufacture it. Capability catalog cursors additionally bind a catalog revision.

Expansion flags default to false. Request results, receipts, chunks, approval evidence, or audit export only when the caller has a real need and the required disclosure authority.

SSE and WebSocket

`follow=true` on action events, action chunks, or global events returns SSE. The current follow loop polls the durable view every 500 ms and sends a keep-alive every 15 seconds. Action-scoped follow ends when the response marks the action terminal; global follow continues until disconnect or error.

Method	Path	Purpose
`GET`	`/aip/v1/ws`	Upgrade to the authenticated native WebSocket transport

The route performs a native-authenticated WebSocket upgrade. A text frame can contain:

- one encoded AIP `Envelope`, which receives one response envelope; or
- a JSON control object with `"type": "subscribe"` for action-scoped or global

event reads.

A subscription accepts `subscription_id`, `action_id`, `session_id`, `tenant_id`, `cursor`, `limit`, `kind`, `kinds`, `include_chunks`, and `follow`. The reviewed server permits at most 16 concurrent subscriptions per connection and buffers eight outbound frames. Those are implementation limits, not AIP wire requirements.

The transport bindings reference owns reconnect, backpressure, and delivery semantics.

Separate fleet, host, and profile surfaces

These related routes do not belong to the ergonomic client API:

Process	Routes	Canonical owner
`getaip-server` connector ingress	`POST /aip/v1/connector-events`, `POST /aip/v1/connector-callbacks`	Connector fleet API
Lifecycle control plane	`GET /health`, `GET /ready`, `GET /metrics`, `POST /aip/v1/connector-control`	Connector fleet API
Standalone connector host	`GET /health`, `GET /ready`, `GET /metrics`, `GET /aip/v1/manifest`, `POST /aip/v1/messages`	Connector fleet API and connector-local operations

MCP and A2A have their own HTTP routes, media types, sessions, and error mappings. Use AIP through MCP or AIP through A2A. Product-specific or migration-only module mounts are not part of the product-neutral native route inventory.

Errors

Invalid path identifiers and typed filters return HTTP `400`. Missing bearer authentication returns `401`. Runtime errors map their `ProtocolError` category and code to an HTTP status, and ergonomic routes return that error in a complete envelope. Clients must use the typed error and retry fields rather than the status code alone.

Related documentation

- Use native AIP
- JSON schemas
- Errors and retry decisions