# Transport bindings

Transport bindings move AIP messages without changing their protocol meaning. Use this reference to choose a binding, encode its frames, authenticate its peer, and recover after an interrupted delivery.

| | |
|---|---|
| SECTION | API Reference |
| TYPE | Reference |
| PROTOCOL | AIP 1.0 |
| SOURCE | docs/reference/transport-bindings.md |

This page describes source revision `d7cce13d1d555644d04a4d73c66c95b113737635`. It separates reusable transport codecs from the routes actually exposed by `getaip-server`. That distinction matters for SSE and WebSocket, where the daemon adds operational query behavior around the lower-level frame codecs.

## Choose a binding

| Binding | Direction | Use it for | Recovery source |
|---|---|---|---|
| Native HTTP | Request and response | Envelope exchange and ergonomic operational queries | Durable action, event, approval, transaction, and receipt views |
| Signed peer HTTP | Request and response | Daemon-to-host connector execution and trusted remote delegation | Pinned route plus durable action or delegation state |
| Native SSE | Server to client | Following action events, stream chunks, or global events | Durable event cursor and action status |
| Native WebSocket | Bidirectional | Envelope exchange and multiplexed event subscriptions | Durable cursor and operational HTTP reads |
| Native NATS | Publish or request and reply | Broker-routed native envelope delivery | Runtime state; this implementation does not configure JetStream persistence |

MCP and A2A reuse HTTP or streaming primitives but have independent protocol, session, version, and error rules. They are compatibility profiles, not aliases for the native bindings on this page.

## Semantics that do not move into the transport

Across every binding, the AIP envelope remains the semantic unit. Transport headers, subjects, connections, and reply inboxes do not replace:

- authenticated principal and tenant binding;
- action, session, delegation, and transaction identifiers;
- approval and policy decisions;
- idempotency keys and replay checks;
- lifecycle and terminal state;
- typed error category, retryability, and receipt evidence.

Network reachability proves neither identity nor authorization. A gateway must still bind a peer to a trusted principal and enforce scopes, tenant, route, approval, delegation, and transaction policy. Do not trust an envelope's `from`

field or caller-controlled transport metadata on its own.

## Common transport frame

The framework-neutral transport crate defines:

```
JSON
{
  "envelope": {
    "aip_version": "1.0",
    "message_type": "aip.discovery.v1.manifest_request",
    "message_id": "msg_0190c42f2d5a70008000000000000001",
    "sent_at": "2026-07-26T12:00:00Z",
    "body": {
      "manifest_request": {}
    }
  },
  "metadata": {
    "transport-key": "transport-value"
  }
}
```

`TransportMessage` contains one complete `Envelope` and a string-to-string metadata map. Empty metadata is omitted. `RawFrame` contains serialized bytes plus the same normalized metadata. The common traits distinguish fire-and-forget publish, request/reply, and correlation-scoped streaming; they do not prescribe durability or authentication.

## Native HTTP

The native HTTP profile is `aip.native.http.v1`. Its codec serializes a complete envelope as JSON and records two frame metadata values:

| Metadata | Value |
| --- | --- |
| `content-type` | `application/aip+json` |
| `aip-message-type` | The envelope's exact message-type string |

The path helper recommends `/aip/v1/actions` for actions, `/aip/v1/manifest` for manifest request or response bodies, and `/aip/v1/messages` for other message families. This helper selects a path only. The deployed daemon exposes manifest discovery as `GET`, so clients must use the method and route contract in the native HTTP API.

The daemon's Axum JSON extractors and responses currently use `application/json`. The signed peer client also sends through a JSON request builder, which uses `application/json`. The binding codec's `application/aip+json` media type is therefore not consistently emitted by the deployed HTTP surfaces at this revision. This difference is recorded in implementation status.

Protocol error categories map to HTTP as follows:

| Category | HTTP status |
| --- | --- |
| `auth` | `401` |
| `policy` | `403` |
| `temporary`, `transport` | `503` |
| `connector` | `502` |
| `economic` | `402` |
| `permanent` | `400` |

An HTTP status is a transport projection. Clients still use the protocol error code, category, `retryable`, and `retry_after_ms` fields for decisions.

## Signed peer HTTP

The pooled `NativeAipHttpClient` is the authenticated HTTP binding used for connector hosts and remote native delegation. Before sending, it:

1. validates the destination against scheme, host, DNS, and private-network

policy;

2. resolves the host and pins one address into a host-specific client;
3. disables redirects and applies the configured timeout and TLS roots;
4. signs the envelope with the route's request identity;
5. adds `AIP-Trust-Domain` and `Idempotency-Key` headers.

The idempotency header uses the delegation ID for a delegation request, the action ID for an action, and the message ID for any other body.

The default endpoint policy allows no hosts, disallows plaintext HTTP and private networks, uses a five-second request timeout, and limits a response to 4 MiB. A caller must opt into every destination. The default client cache retains at most 256 host-specific pools.

The client verifies a response before returning it. The response must:

- carry the exact expected peer DID and a valid Ed25519 envelope signature;
- come from the expected principal and target the request signer;
- preserve the request correlation ID;
- reference the exact request message in `in_response_to`; and
- have a timestamp within five minutes of the verifier's clock.

The peer-security object defaults to two retries after the first attempt. Retries occur only when sending the HTTP request returns a transport error. They reuse the same signed envelope and idempotency header, with bounded exponential delays starting at 50 ms. A returned HTTP response, malformed body, oversized body, or failed peer verification is not transport-retried by this loop.

Connector dispatch adds a complete `security.connector_route` and verifies additional assignment fences at both ends. See the connector fleet HTTP API for those fields and central callback ingress.

## Native SSE in `getaip-server`

Set `follow=true` on one of these bearer-authenticated native routes:

| Route | Stream scope |
|---|---|
| `/aip/v1/events` | Events visible to the authenticated identity |
| `/aip/v1/actions/{action_id}/events` | One action's events and optional chunks |
| `/aip/v1/actions/{action_id}/chunks` | One action's durable chunks |

The response uses `text/event-stream`. A query `cursor` takes precedence over `Last-Event-ID`; otherwise the header becomes the resume cursor. Persist only a cursor returned by the same route and authorization context.

The daemon emits these frame forms:

| Event name | | JSON data |
|---|---|---|
| An event's `kind` | Event ID | `{"event": ...}` |
| `aip.stream.chunk` | `chunk:{sequence}` | `{"chunk": ...}` |

| Event name | | JSON data |
|---|---|---|
| `aip.stream.cursor` | Returned cursor | `{"next_cursor": "..."}` |
| `aip.stream.terminal` | Current cursor when present | `{"terminal": true, "action_id": "..."}` |
| `aip.error` | None | `{"error": ...}` followed by stream termination |

Action-scoped follow ends after a terminal response. Global follow continues until disconnect or error. The current loop polls the durable view every 500 ms and emits an SSE keep-alive every 15 seconds.

SSE is delivery, not storage. A disconnect does not cancel the action. Resume with the retained cursor, accept possible duplicate delivery, and confirm terminal business state through the durable action or transaction view.

### Framework-neutral SSE codec

The lower-level `aip-transport-sse` crate has profile `aip.sse.stream.v1`. It serializes an entire envelope in each `data` field and uses generic event names such as `ack`, `chunk`, `done`, `result`, `delegation_request`, `delegation_result`, `heartbeat`, `heartbeat_ack`, `error`, or `message`.

Its `SseCursor` is an in-memory zero-based index rendered as `sse_{index}`. The embedded `SseTransport` groups messages by correlation ID and closes a stream after a done chunk, action result, delegation result, or error. These helpers do not describe the durable cursor or data shape returned by the deployed `getaip-server` follow routes.

## Native WebSocket in `getaip-server`

`GET /aip/v1/ws` upgrades through the native-auth middleware. A client text frame can use either of two forms:

- one complete AIP `Envelope`, which receives one response envelope; or
- a JSON object with `"type": "subscribe"`, which requests an action-scoped

or global event page.

A subscription object accepts `subscription_id`, `action_id`, `session_id`, `tenant_id`, `cursor`, `limit`, `kind`, `kinds`, `include_chunks`, and `follow`. With `action_id`, the daemon issues an action-events query. Without it, the daemon issues a global event query. `tenant_id` is used only by the action-scoped request at this revision. Responses remain complete AIP envelopes and include WebSocket cursor and terminal details in `trace`.

`follow=true` repeats the query every 500 ms until an action becomes terminal, an error occurs, or the connection closes. The daemon allows at most 16 active subscriptions per connection and buffers eight outbound frames. If a slow consumer exhausts that buffer, delivery stops rather than growing memory without a bound.

Ping receives Pong with the same payload. Close is echoed when capacity allows. Binary application frames and inbound Pong frames are ignored by the daemon. After a disconnect, reconnect with a durable cursor or use native HTTP reads; connection state is not a recovery record.

### Framework-neutral WebSocket codec

The lower-level codec uses profile `aip.websocket.stream.v1`. It encodes one envelope per text frame, rejects non-text application frames, maps Ping to Pong, and groups its in-memory queues by session ID. Its subscription trait tracks correlation IDs only inside that transport object. The daemon's `"type": "subscribe"` control object is an application adapter above this codec.

## Native NATS

The native NATS profile is `aip.native.nats.v1`. A request or publish subject has this form:

```
TEXT
aip.v1.{trust_domain}.{service}.{version}.{message_type}
```

Dots, spaces, forward slashes, and backslashes in every component are replaced with underscores. For example, message type `aip.core.v1.action` becomes `aip_core_v1_action` in the subject. A daemon listens on the service wildcard:

```
TEXT
aip.v1.{trust_domain}.{service}.{version}.>
```

The library also defines a correlation-scoped stream subject:

```
TEXT
aip.v1.{trust_domain}.{service}.{version}.stream.{correlation_id}
```

The payload is JSON `TransportMessage`. For compatibility at the receive boundary, the decoder also accepts a raw envelope and wraps NATS headers as transport metadata. Outbound request, response, and publish operations use the wrapped form.

Every outbound frame sets these NATS headers:

| Header | Source |
| --- | --- |
| `AIP-Version` | `envelope.aip_version` |
| `AIP-Message-Type` | Exact unsanitized message type |
| `AIP-Message-Id` | Message ID |
| `AIP-Session-Id` | Session ID when present |
| `AIP-Correlation-Id` | Correlation ID when present |

The default request timeout is 30 seconds. The `getaip-server` listener processes both ordinary and queue subscriptions. It returns one `TransportMessage` only when the NATS message has a reply subject; a publish without a reply is still sent through the gateway. Decode failure produces a typed NATS transport-error envelope when a reply is possible.

A native NATS delegation route validates the broker destination against its endpoint policy, signs the request envelope, and performs one request/reply exchange. It applies the same peer-DID, signature, principal, recipient, correlation, request-reference, and clock checks used by signed HTTP. The HTTP client's retry-budget loop is not applied to this NATS request path.

Optional username/password authentication is attached only at connection time; the password is omitted from serialized configuration and redacted from debug output. Broker accounts, TLS, subject ACLs, availability, and retention remain deployment responsibilities. The current transport uses Core NATS APIs and does not configure JetStream persistence or durable consumers.

A queue group distributes requests among daemon listeners. It does not replace AIP action leases, idempotency reservations, replay protection, or durable terminal reads. When a configured listener is not running, daemon readiness is false.

## Compatibility-profile boundary

Do not apply native frame rules to MCP Streamable HTTP, legacy MCP HTTP/SSE, or A2A JSON-RPC and streaming. Use AIP through MCP and AIP through A2A for their media types, session headers, protocol versions, and errors.

## Delivery and recovery decisions

| Observation | Safe next step |
| --- | --- |
| HTTP request was rejected with a typed error | Follow its retry fields and the capability's side-effect contract |
| Signed peer send failed before a response | Let the bounded peer client reuse the same signed request and idempotency header; then query durable state |
| HTTP, WebSocket, or NATS outcome is uncertain | Do not create a new mutation identity; read the existing action, transaction, or receipt |
| SSE or WebSocket disconnected | Resume from the last retained route cursor and tolerate replay |
| NATS subscriber changed because of queue-group rebalance | Continue from runtime state; do not infer completion from broker delivery |
| Peer signature, principal, recipient, correlation, or request reference failed | Reject the response and investigate the trust boundary; do not retry it as a valid result |

See errors and retry decisions for the complete taxonomy, actions and sessions for durable recovery, and identity and trust for principal binding.

## Related documentation

- Configure `getaip-server`
- Use native AIP