Gateway

Use this page to understand how the reviewed gateway converts a native AIP envelope plus transport-established trust into the context used by the runtime. It explains where signatures, replay protection, identity and tenant resolution, quer

SECTION	Architecture and Security
TYPE	Architecture
PROTOCOL	AIP 1.0
SOURCE	docs/architecture/gateway.md

The page describes version `2.0.0` of the Rust workspace at source revision `d7cce13d1d555644d04a4d73c66c95b113737635`. `aip-gateway` is a library composition boundary. It is not itself an HTTP, NATS, MCP, or A2A server, and its process placement is not prescribed by AIP 1.0.

Place the gateway between ingress and lifecycle

Transports authenticate connections or credentials and decode bytes. Profiles map a bounded foreign protocol into native AIP. The gateway receives the native envelope, establishes trusted message context, applies cross-message policy, and selects the runtime operation. The runtime owns lifecycle state and execution.

Boundary	Gateway responsibility	Responsibility kept elsewhere
Ingress	Accept a signed native envelope or an explicitly authenticated edge result	Listener, token, certificate, NATS peer, MCP, or A2A authentication
Trust	Bind signer or authenticated actor, reject replay, and resolve tenant, identity, and credential context	Deployment identity directory and credential material
Authorization	Apply operational object, action context, tenant, principal, and session checks before dispatch	Capability policy and lifecycle decisions inside the runtime
Dispatch	Map every native message family to its runtime, discovery, session, query, or observation operation	Durable stores, workers, handlers, and provider execution
Egress	Correlate native responses and enforce remote peer or callback destination policy	Public route serialization and deployment network controls

Compatibility profiles do not bypass this boundary. Their authenticated hosts construct native envelopes and call a trusted gateway entrypoint; they do not create a second action lifecycle.

Choose one of four ingress trust entrypoints

The library exposes four entrypoints so the caller cannot silently mix payload claims with transport evidence.

Entrypoint	Trust supplied by caller	Gateway behavior
`handle_envelope`	None	Verify the envelope signature and trusted DID binding under policy, then claim replay state
`handle_authenticated_envelope`	One already authenticated in-process principal	Replace `Envelope.from`, synthesize authenticated actor context, and claim replay state
`handle_verified_envelope`	Authenticated actor plus optional verified tenant and credential handle	Replace `Envelope.from`, project trusted tenant and credential context, and claim replay state
`handle_resolved_envelope`	Complete validated `ResolvedIdentity` from a trusted edge	Replace `Envelope.from`, install its actor, tenant, credential, and identity context, and claim replay state

The default policy is fail closed: signed envelopes are required, payload identity is not trusted, action senders are required, and message replay is rejected. The gateway's signer registry adds the independent DID-to-principal trust decision after signature verification proves key possession.

An explicit local-development constructor disables signature enforcement and allows payload identity. Its source contract says it must not serve a network-facing deployment.

Already authenticated entrypoints skip native envelope-signature verification because their caller owns that edge decision. They still apply replay claims, query authorization, and runtime policy. The resolved entrypoint trusts its caller to have completed identity consistency; other action paths invoke the configured resolver when policy requires it.

Follow the ordered request pipeline

```
MERMAID
flowchart LR
    E["Native envelope"] --> A["Authenticate edge or signed DID"]
    A --> R["Claim message replay window"]
    R --> I["Resolve trusted action identity"]
    I --> C["Build MessageContext"]
    C --> Q["Authorize operational query"]
    Q --> D["Dispatch native message family"]
    D --> L["Runtime or discovery owner"]
    L --> O["Correlated response envelope"]
```

The text equivalent is:

1. The caller supplies either a raw native envelope or transport-authenticated

actor state through the matching entrypoint.

2. The gateway verifies the native signature and trusted principal binding when

that work was not completed by the edge.

3. It rejects messages outside the configured past or future time window and

claims the message ID in the selected replay store when replay rejection is enabled.

4. For an action or delegated child action, it resolves deployment-owned

identity when the capability or request requires it. A cancellation can request operational tenant context through the same resolver.

5. It builds one `MessageContext` from envelope correlation data and verified

actor, tenant, credential, and identity values.

6. It applies query and session policy before selecting the message-family

branch.

7. It invokes the runtime or discovery owner and creates a native response with

the original session and correlation coordinates.

Metrics count the received envelope and action before policy evaluation. They do not establish that authentication, authorization, or execution succeeded.

Authenticate a native signer independently of payload claims

For `handle_envelope`, the gateway verifies the Ed25519 signature carried in the envelope security object and obtains its DID. The DID must already map to a trusted AIP principal. `Envelope.from` must match that principal's ID and kind.

A self-generated DID, a valid signature, or a matching payload sender is not sufficient alone. The trust registry supplies the deployment decision that the key may act as that principal.

Authenticated-edge entrypoints overwrite `Envelope.from` with the actor they received. A payload cannot retain a different sender after the edge has established identity.

Claim replay before resolving authority

When replay rejection is enabled, the gateway checks `sent_at` against the configured retention window and maximum future skew. It then claims the `message_id` in the runtime replay store until the calculated expiration.

The claim occurs before identity resolution and dispatch. A duplicate ID or an out-of-window timestamp fails without repeating the action, query, or observation branch. Disabling replay rejection is an explicit policy change; an authenticated actor does not disable it automatically.

Replay protection is message-level evidence. Provider idempotency and uncertain outcome reconciliation remain separate runtime and connector responsibilities.

Resolve identity only when required

For a root action and the child action inside a delegation request, identity resolution runs when any of these conditions holds:

- the action carries an identity claim that requires verification;
- a fleet policy enables resolution for a capability absent from the bounded

local manifest;

- the capability ID is explicitly marked as identity-required; or
- the local capability contract requires a credential, accepted issuer, or

credential scope.

The deployment-owned resolver receives the authenticated actor, the claimed identity, and the accepted credential rules. It may return a verified tenant, an opaque credential handle, and a normalized identity context.

The gateway then enforces these invariants:

- the resolver cannot replace the transport actor's principal ID or kind;
- resolved scopes must be a subset of the transport-authenticated scopes;
- the verified tenant and credential handle must validate;
- a required credential must exist and use an accepted issuer and scopes;
- a tenant carried by the credential must match verified tenant membership; and
- tenant or credential values already present in resolved identity cannot

conflict with their verified forms.

The gateway restores the original authenticated actor after validation and projects only canonical verified tenant and credential references. The resolver never receives permission to elevate the edge actor.

The default resolver denies resolution. A deployment that enables fleet lookup, identity claims, or credential-bearing capabilities must install an appropriate trusted resolver.

Authorize operational reads and selectors

Before message-family dispatch, the gateway requires a sender for operational messages and applies `QueryAuthorizationService` to query operations. The authorization request identifies:

- the authenticated actor and optional verified tenant;
- the object family and requested operation;
- an optional selected principal;
- an optional tenant selector; and
- the actor's effective scopes.

Missing membership, tenant mismatch, insufficient authority, and protected audit access become typed authorization errors. If both the envelope and the request carry a session ID, the values must match.

This query gate does not replace action governance. The runtime still evaluates capability contracts, idempotency, approvals, transactions, and handler admission for executable work.

Dispatch by owner, not by transport

The gateway's native message match is exhaustive, but its branches fall into a small set of ownership patterns.

Message purpose	Gateway action	State owner
Handshake	Verify that the declared client matches the authenticated actor, negotiate profile and capability intersections, and create a secure session	Runtime session service
Manifest request	Filter and return the bounded local manifest	Gateway local composition
Received manifest	Validate it as a remote observation without publishing it as a callable local implementation	Remote observation registry and event log
Action or delegated child action	Require an actor and submit the trusted context	Runtime scheduler and selected handler
Lifecycle or operational query	Apply query policy and call the context-aware runtime view	Corresponding durable runtime service
Result, status, event, audit, receipt, or remote view	Ingest or record the authenticated observation	Runtime lifecycle and event stores
Cancel, approval, transaction, escalation, or channel work	Select the corresponding runtime transition	Dedicated runtime service

An action can return a terminal `ActionResult` or a queued `Ack`; the gateway does not infer completion from the transport request. Errors from the runtime are mapped into typed AIP error envelopes by the caller-facing boundary.

The response helper copies the request session and correlation IDs, identifies the local manifest agent as sender, and addresses the authenticated actor. It does not itself guarantee that every embedding signs the returned envelope; transport and peer-specific code own that requirement.

Admit local capabilities atomically

A network-facing gateway that declares callable local capabilities uses the constructor that admits the manifest and complete handler map before the gateway becomes observable. Invalid schema, binding, implementation claim, or projected identifier fails the composition without publishing partial discovery or handler state.

Outbound or frozen connectors can similarly discover a manifest, create one handler per non-resource capability, and atomically admit the manifest-handler set before becoming registered for readiness.

This library capability does not mean the primary `getaip-server` bundles product adapters. Its target fleet composition uses product-neutral catalog and remote handler services, while explicitly trusted local modules use the startup-frozen local path.

Separate handshake, discovery, and action authority

A handshake requires transport-established identity. The declared client must match the authenticated actor, and the gateway accepts only the intersection of requested and configured profiles. With no compatible profile, it returns a rejected handshake response rather than creating a session.

An accepted handshake creates a secure runtime session and returns its resume token plus the requested capability intersection present in the local manifest. This negotiation does not grant a later action permission or prove provider readiness.

A received remote manifest passes manifest admission and is stored as a remote observation. It does not join the local callable catalog or install an action handler. Discovery, identity, authorization, and implementation admission remain separate decisions.

Route delegation without weakening the child action

A delegated child action passes through the same identity-resolution boundary as a root action. The request target must be the local gateway or a configured remote route.

Static routes selected by delegate principal or child capability take precedence. Connector-owned delegation routers are consulted only when no static route matches. If neither external path accepts the request, the runtime can use its local delegation path only for the local target.

Native HTTP and NATS peer bindings sign outbound envelopes and authenticate the expected response principal, DID, and trust domain. HTTP destinations are checked against URL, host, DNS, redirect, address-range, timeout, response-size, and optional private-PKI policy. Transport retry does not change the delegation identity.

Protect callback egress independently

The callback dispatcher selects native HTTP, native NATS, in-process SSE, or an A2A push profile. Externally delivered native callbacks require a configured signer. HTTP and NATS destinations pass the deployment's destination policy; in-process SSE is keyed by correlation ID.

A2A push credentials can be sealed with a deployment-provided AES-256-GCM key before durable storage. Every gateway replica that may recover those callbacks needs the same key. Callback destination authorization, credential encryption, delivery retries, and action completion are distinct states.

Interpret readiness narrowly

`GatewayReadiness` probes runtime storage and every connector registered in the gateway's local connector map. It reports ready only when all those checks pass.

That result does not cover an external profile edge, registry, remote connector host, provider, callback target, or deployment load balancer. The daemon composes additional fleet and process-level readiness outside the library report.

Failure ownership and non-implications

Failure	Owning decision
Invalid or untrusted native signature	Reject at raw-envelope authentication
Duplicate or stale message ID	Reject at replay claim before dispatch
Resolver actor or scope substitution	Reject identity resolution
Required tenant or credential evidence is missing	Reject trusted-context construction
Unauthorized query or session selector mismatch	Reject before message-family routing
Invalid local manifest-handler set	Reject gateway construction or connector registration atomically
Runtime storage, governance, or handler failure	Return a typed runtime-derived AIP error
Remote peer or callback destination failure	Retain the runtime's delivery or delegation evidence for bounded retry or recovery

The gateway does not own product data-transfer objects, provider credentials, provider APIs, listener authentication configuration, durable store algorithms, or deployment topology. Source support for an entrypoint or egress profile does not prove that a deployment enabled, secured, or qualified it.

Related pages

- Runtime
- Security model
- Identity and trust
- HTTP API
- Transport bindings