Use native AIP

Use this guide when a client needs the complete AIP action lifecycle without projecting it through MCP, A2A, or another compatibility protocol. You will discover a capability, invoke it with a stable action identity, follow durable state, a

SECTION	Build with AIP
TYPE	Guide
PROTOCOL	AIP 1.0
SOURCE	docs/guides/use-native-aip.md

This guide applies to AIP 1.0 and the Rust implementation at source revision `d7cce13d1d555644d04a4d73c66c95b113737635`. The commands were checked against that source; this documentation pass did not start a daemon or execute a provider operation.

Prerequisites

You need:

- a `getaip-server` and `getaip` build from the reviewed revision;
- the base URL of a native AIP endpoint;
- a native HTTP bearer credential accepted by that endpoint;
- permission to invoke and read the selected action;
- `curl` and two terminals for the optional loopback setup.

Run the loopback commands from the reviewed repository root.

For a remote connector capability, the deployment must also have an enabled connector registry, an active admitted version, an enabled tenant binding, and an eligible connector-host replica. The authenticated principal needs a deployment-owned trusted identity binding that resolves the execution tenant.

The daemon's `--native-tenant-id` option has a narrower role: it binds the static bearer credential to tenant-scoped HTTP capability discovery. It does not, by itself, establish the trusted tenant used for remote execution.

1. Prepare an optional loopback endpoint

Skip this step if you already have a native endpoint. For a local development endpoint, create an owner-only bearer-token file:

```
SH
umask 077
printf '%s\n' 'local-development-token' > .aip-native-token
```

Start the product-neutral daemon in the first terminal:

```
SH
cargo run --locked -p getaip-server -- \
  --bind 127.0.0.1:18080 \
  --service-id agent:getaip:server:native-guide \
  --native-bearer-token-file .aip-native-token \
  --native-principal agent:getaip:cli \
```

```
  --native-principal-scope action:read \
  --storage-dir .getaip-server-native-guide
```

Leave the process running. This command is restricted to loopback and uses a dedicated local state directory. It does not configure a connector registry, tenant catalog, remote host, or production identity provider.

Check readiness from the second terminal:

```
SH
curl --fail --silent --show-error http://127.0.0.1:18080/ready
```

Expected result: HTTP `200` with `"status": "ready"`. Readiness proves the configured daemon boundary is ready. It does not prove that an unconfigured connector or provider is reachable.

Set reusable client variables:

```
SH
export AIP_URL=http://127.0.0.1:18080
export AIP_TOKEN_FILE=.aip-native-token
```

For a non-loopback deployment, use HTTPS, owner-only credential files, and the deployment's expected peer DID and trust material. `getaip` supports a native signing-seed file, a pinned peer-DID file, a private-PKI CA file, and a bounded response size. Raw `curl` does not provide the same signed-response checks.

2. Fetch the participant manifest

Fetch the endpoint's native manifest before invoking a capability:

```
SH
cargo run --locked -p getaip-cli -- \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  manifest fetch "$AIP_URL"
```

For the loopback daemon, the manifest identifies `agent:getaip:server:native-guide` and includes the built-in `cap:aip:server:health` capability. It also advertises the profiles enabled by that daemon.

The manifest describes one participant's complete declared surface. It is not the same as the connector fleet's tenant-filtered catalog. A manifest may include local capabilities that are unrelated to connector bindings, while a tenant catalog can contain capabilities admitted from many connector versions.

3. Invoke one local native action

Call the built-in health capability with a stable action ID:

```
SH
export AIP_ACTION_ID=act_native_guide_health_001

cargo run --locked -p getaip-cli -- \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action call "$AIP_URL" \
  cap:aip:server:health \
  --action-id "$AIP_ACTION_ID" \
  --mode sync \
  --input '{}'
```

Expected result: an `aip.core.v1.action_result` envelope whose action ID is `act_native_guide_health_001` and whose status is `completed`. Store the action ID even when the first response is terminal. Durable state, not the lifetime of the HTTP connection, owns the action lifecycle.

The bearer edge authenticates this request as the daemon-configured `agent:getaip:cli` principal. An identity or `from` value inside the request is a claim and cannot override that authenticated actor.

4. Discover a tenant-visible fleet capability

Use this step against an endpoint whose connector catalog is enabled. Point `AIP_URL` and `AIP_TOKEN_FILE` at that deployment, then request the first bounded page:

```
SH
cargo run --locked -p getaip-cli -- \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  capability list "$AIP_URL" \
  --limit 50
```

The response contains:

- `catalog_revision`, which identifies the registry view used for the page;
- `capabilities`, a list of capability definitions and their contract and

schema digests;

- `next_cursor`, an opaque cursor when another page exists;
- `total`, the total matches at that revision.

The bearer credential must be mapped to a tenant for this HTTP route. The catalog returns only capabilities with an enabled binding for that tenant, an enabled connector instance, an active version, and an enabled connector type. An optional `--profile` filter selects versions whose manifest advertises that profile.

Catalog visibility is admission evidence, not liveness evidence. The query does not require a ready replica. Route selection can still fail later if no eligible host has a valid lease and available capacity.

To continue pagination, pass the exact `next_cursor` returned by the preceding page:

```
SH
export NEXT_CURSOR='paste-the-exact-next_cursor-value'

cargo run --locked -p getaip-cli -- \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  capability list "$AIP_URL" \
  --cursor "$NEXT_CURSOR" \
  --limit 50
```

Do not combine a cursor with pages from another catalog revision. If the server reports `connector_catalog.stale_cursor`, discard the old cursor and restart from the first page.

5. Check the selected contract

Before invoking a catalog capability, inspect its complete definition. At a minimum, record and evaluate:

- the exact capability ID, contract digest, and schema digest;
- the input and output schemas;
- declared execution modes and streaming behavior;
- side effects, risk, and human-approval requirements;
- idempotency key requirements and scope;
- transaction, reconciliation, cancellation, and compensation support;
- credential and data boundaries;
- profile bindings required by the client.

Do not infer these properties from a display name. If the contract changed since the caller last evaluated it, repeat the caller's policy and input decision against the new revision before submitting work.

6. Invoke without choosing a connector host

Prepare the selected input as a JSON object and keep the identifiers outside the file:

```
SH
export AIP_CAPABILITY_ID='cap:replace:with:catalog-id'
export AIP_ACTION_ID='act_replace_with_stable_id_001'

cargo run --locked -p getaip-cli -- \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action call "$AIP_URL" \
  "$AIP_CAPABILITY_ID" \
  --action-id "$AIP_ACTION_ID" \
  --input @request.json
```

Add `--mode`, `--idempotency-key`, approval, or transaction options only as required by the selected contract. For a replay-sensitive mutation, retain the original action ID, idempotency key, transaction ID, and plan ID together.

Do not send an instance ID, replica ID, host URL, connector version, or credential reference. The central runtime uses the verified execution tenant and capability ID to resolve an enabled binding. It persists an action-scoped route before dispatch and sends a signed native AIP envelope to the assigned host. The host then rechecks the pinned route and trusted gateway identity.

The client sees the ordinary AIP action lifecycle in both placements. A local capability runs through its admitted in-process handler. A fleet capability runs through the remote handler, registry assignment, and connector host. A client retry with the same action identity does not authorize choosing another provider account or replica.

7. Read or follow the durable action

Read the current lifecycle view independently of the submission response:

```
SH
cargo run --locked -p getaip-cli -- \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action status "$AIP_URL" \
  "$AIP_ACTION_ID" \
  --include-result \
  --include-receipts
```

For long-running or streaming work, follow action-scoped events and chunks:

```
SH
cargo run --locked -p getaip-cli -- \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action events "$AIP_URL" \
  "$AIP_ACTION_ID" \
  --include-chunks \
  --follow
```

Persist the last acknowledged cursor before reconnecting. A resumed consumer must tolerate replay. A cursor orders the retained event view; it is not an exactly-once delivery token.

If the submission connection failed before a response, query the stable action ID first. Retry only under the capability's idempotency and uncertainty rules, using every original identity. Generating a new action ID creates a new logical operation and can permit another provider effect.

8. Request cancellation

Request cancellation of the same durable action:

```
SH
cargo run --locked -p getaip-cli -- \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action cancel "$AIP_URL" \
  "$AIP_ACTION_ID" \
  --reason 'The requester withdrew the operation'
```

For a fleet action, cancellation uses the persisted route assignment rather than resolving a new host. The runtime records cancellation intent, but a provider mutation may already have completed or may not support remote cancellation. Query the action and any transaction or reconciliation state before deciding that another mutation is safe.

9. Send a raw envelope when necessary

Use the generic native message endpoint when implementing a client or testing a message family without an ergonomic CLI command. Keep this example on the loopback endpoint from step 1:

```
SH
export AIP_URL=http://127.0.0.1:18080
export AIP_TOKEN_FILE=.aip-native-token
```

Save this example as `action-envelope.json` and replace `sent_at` with the current RFC 3339 time:

```
JSON
{
  "aip_version": "1.0",
  "message_type": "aip.core.v1.action",
  "message_id": "msg_native_guide_raw_0001",
  "sent_at": "2026-07-26T20:00:00Z",
  "from": {
    "id": "agent:getaip:cli",
    "kind": "agent"
  },
  "body": {
    "action": {
      "id": "act_native_guide_raw_0001",
      "capability_id": "cap:aip:server:health",
      "input": {},
      "mode": "sync"
    }
  }
}
```

Load the development token without placing it literally in the command:

```
SH
AIP_TOKEN=$(tr -d '\r\n' < "$AIP_TOKEN_FILE")

curl --fail-with-body \
  -H "Authorization: Bearer $AIP_TOKEN" \
  -H 'Content-Type: application/aip+json' \
  --data @action-envelope.json \
  "$AIP_URL/aip/v1/messages"

unset AIP_TOKEN
```

The endpoint validates the envelope schema and dispatches the typed body. With bearer authentication, the configured edge principal is authoritative; the payload's `from` field does not establish identity.

For signed peer traffic, use a client that signs the envelope, pins the expected peer DID, verifies the response signature and correlation, rejects redirects, and limits response size. `getaip` implements those checks when its native signing options are configured.

Verify the outcome

For the loopback path, require readiness and the stored local result:

```
SH
curl --fail --silent --show-error "$AIP_URL/ready" >/dev/null

cargo run --locked -p getaip-cli -- \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action status "$AIP_URL" \
  act_native_guide_health_001 \
  --include-result >/dev/null
```

Both commands must exit with status `0`. For a fleet capability, also confirm that the terminal result belongs to the original action ID and that any required receipt, transaction, or reconciliation evidence is present. A successful call demonstrates only the configured path and provider response; it is not connector qualification or a production-readiness claim.

Stop and clean up the loopback state

Stop the local daemon with `Control-C`. Keep `.getaip-server-native-guide` if you need to restart the daemon and inspect the same action. When the local history and development credential are no longer needed, inspect and remove only those two paths:

```
SH
du -sh .getaip-server-native-guide
ls -l .aip-native-token
rm -rf -- .getaip-server-native-guide
rm -- .aip-native-token
```

Do not apply this cleanup to a shared endpoint, connector registry, provider, or retained qualification evidence.

Resolve common failures

Symptom	Meaning	Safe next action
Native route returns `401`	Bearer credential is missing or invalid	Confirm the intended token file and endpoint; do not print the token
Catalog returns `connector_catalog.disabled`	This daemon has no connector catalog	Configure the registry data plane or use the endpoint's manifest for local capabilities
Catalog returns `connector_catalog.tenant_required`	The bearer identity has no discovery tenant	Bind the intended static credential with `--native-tenant-id`; do not copy a tenant from request data
Catalog returns `connector_catalog.stale_cursor`	The registry changed during pagination	Restart from the first page and evaluate the new revision
Capability is absent	No visible enabled binding and active version matched the tenant and filters	Check the tenant binding, instance, version, type, filters, and catalog revision
Remote execution reports that a verified tenant is required	Discovery identity was configured, but runtime identity enrichment did not establish execution tenant	Correct the trusted identity binding for the authenticated principal; do not rely on `--native-tenant-id`
Invocation has no eligible replica	Admission was visible, but no ready route has valid lease, capacity, and policy	Preserve the action identity and ask the operator to restore or drain the fleet deliberately
Client disconnected after submission	The durable action outcome is unknown to the client	Query the original action ID before any retry
Cancellation returns but the provider may have changed state	Cancellation is intent, not proof of rollback	Inspect the terminal action, provider operation, transaction, and reconciliation evidence

Related documentation

- Complete the connector fleet quickstart

to observe one deterministic remote path.

- Read Capabilities and contracts before choosing

modes, retries, or transaction options.

- Read Actions and sessions for lifecycle,

idempotency, events, and cancellation semantics.

- Use the Native HTTP API for exhaustive route and

query syntax.

- Use Errors and retry decisions before automating recovery.