getaip CLI

getaip is the developer and operator client for native AIP, NATS, Model Context Protocol (MCP), schemas, connector qualification, connector admission, orchestration plans, and receipt verification. This reference covers all 15 command famil

SECTION	API Reference
TYPE	Reference
PROTOCOL	AIP 1.0
SOURCE	docs/reference/cli.md

Use the generated `--help` output for the binary actually installed. Use this page to understand how its commands select transports, obtain credentials, validate files, print results, and divide authority.

Invocation

Invoke an installed binary directly:

```
SH
getaip COMMAND --help
```

From the source repository, place CLI arguments after Cargo's `--` separator:

```
SH
cargo run -p getaip-cli -- COMMAND --help
```

All native HTTP and NATS security options are global. The command parser accepts them before or after a subcommand. Examples on this page put them first so their scope is visible.

Command index

Family	Leaf commands	Count
`schema`	`export`	1
`nats`	`signer-did`	1
`manifest`	`fetch`, `fetch-nats`, `validate`	3
`capability`	`list`	1
`action`	`call`, `call-nats`, `list`, `status`, `result`, `events`, `cancel`	7
`approval`	`list`, `records`, `get`, `request`, `decide`, `list-nats`, `request-nats`, `decide-nats`	8
`events`	`list`	1
`session`	`list`, `get`, `close`, `resume`	4
`transaction`	`get`	1
`audit`	`events`	1
`resource`	`list`, `read`	2
`conformance`	`run`	1
`mcp`	`inspect`, `call-tool`, `conformance`, `serve-stdio`, `bridge`	5
`connector`	`test`, `registry-migrate`, six `registry` commands, three `orchestration` commands	11
`receipt`	`get`, `verify`	2
Total		**49**

Global native HTTP options

Option	Default	Behavior
`--native-bearer-token TOKEN`	None	Adds one bearer token to native HTTP requests; conflicts with the token-file source
`--native-bearer-token-file PATH`	None	Reads the bearer token from an owner-only file; conflicts with the inline source
`--native-signing-seed-file PATH`	None	Signs native envelope requests with an Ed25519 key
`--native-principal-id PRINCIPAL_ID`	`agent:getaip:cli`	Sets the principal carried by signed native requests
`--native-trust-domain TRUST_DOMAIN`	None	Sets signed native trust-domain metadata; requires a signing seed
`--native-tls-ca-file PATH`	Platform roots	Adds one PEM root certificate for private-PKI endpoints
`--native-peer-did DID`	None	Pins the Ed25519 DID expected on signed native envelope responses
`--native-peer-did-file PATH`	None	Reads the pinned peer DID from a public configuration file
`--native-max-response-bytes BYTES`	`4194304`	Rejects a native HTTP response whose body exceeds this positive limit

The native HTTP client does not follow redirects. When native signing is enabled, a peer DID is mandatory. Signed envelope responses must match that DID and carry a valid signature and sender binding. They must also address the request sender, preserve session and correlation context, reference the exact request, and have a timestamp within five minutes of the local clock.

These envelope checks apply to commands that send native envelopes. Ordinary HTTP `GET` and ergonomic `POST` commands use bearer authentication and bounded response reading; configuring a signing seed does not convert every route into a signed-envelope exchange.

Global NATS options

Option	Default	Behavior
`--nats-username USERNAME`	None	Selects NATS username/password authentication
`--nats-password-file PATH`	None	Reads the NATS password from an owner-only file
`--nats-signing-seed-file PATH`	None	Signs AIP envelopes sent through NATS

The username and password file are a pair: providing only one is an error. Broker authentication and AIP envelope signing are separate. A signing seed must encode exactly 32 bytes as 64 hexadecimal characters.

Derive the DID to register for the configured signing key without printing the seed:

```
SH
getaip --nats-signing-seed-file /run/secrets/getaip-nats-seed \
  nats signer-did
```

The command prints a JSON object containing `did` and `principal_id`.

Environment fallbacks and precedence

Variable	Corresponding option	Precedence at this revision
`GETAIP_NATIVE_BEARER_TOKEN`	`--native-bearer-token`	Explicit inline option, then environment
`GETAIP_NATIVE_BEARER_TOKEN_FILE`	`--native-bearer-token-file`	Explicit file option, then environment
`GETAIP_NATIVE_SIGNING_SEED_FILE`	`--native-signing-seed-file`	Explicit option, then environment
`GETAIP_NATIVE_PRINCIPAL_ID`	`--native-principal-id`	Environment, then option/default
`GETAIP_NATIVE_TRUST_DOMAIN`	`--native-trust-domain`	Explicit option, then environment
`GETAIP_NATIVE_TLS_CA_FILE`	`--native-tls-ca-file`	Explicit option, then environment
`GETAIP_NATIVE_PEER_DID`	`--native-peer-did`	Explicit inline option, then environment
`GETAIP_NATIVE_PEER_DID_FILE`	`--native-peer-did-file`	Explicit file option, then environment
`GETAIP_NATIVE_MAX_RESPONSE_BYTES`	`--native-max-response-bytes`	Environment, then option/default
`GETAIP_NATS_USERNAME`	`--nats-username`	Explicit option, then environment
`GETAIP_NATS_PASSWORD_FILE`	`--nats-password-file`	Explicit option, then environment
`GETAIP_NATS_SIGNING_SEED_FILE`	`--nats-signing-seed-file`	Explicit option, then environment
`GETAIP_MCP_BEARER_TOKEN`	MCP `--bearer-token`	Explicit subcommand option, then environment

The environment-first behavior for native principal ID and maximum response bytes is intentional documentation of the current code, not a general CLI precedence rule. Clear those variables before trying to override them on the command line.

An inline native bearer token conflicts with any token-file source, including one supplied through the environment. Inline and file peer-DID sources have the same mutual-exclusion rule.

File inputs

JSON values

Action input, identity context, approval objects, approval lifecycle messages, and MCP tool arguments accept either inline JSON or `@path`:

```sh
getaip action call "$AIP_URL" cap:example:create --input @request.json
```

`@path` is a CLI convention, not part of the AIP wire format. The file must contain one complete JSON value. Registry and orchestration commands instead take explicit path options and decode the expected typed object.

Credential and trust files

Secret-file readers reject symlinks, non-regular files, empty files, oversized files, and, on Unix, any group or other permissions. They strip trailing line endings. Public peer-DID and CA files must also be regular non-symlink files and must not be group- or world-writable.

File purpose	Maximum size
Native bearer token or NATS password	16 KiB
Native or NATS Ed25519 signing seed	1 KiB
Registry administrator database URL	16 KiB
Pinned native peer DID	1 KiB
Native TLS CA PEM	1 MiB

Registry and orchestration documents have separate bounded JSON readers. Their limits range from 1 MiB for trust policies to 32 MiB for admission packages, observations, and signed plans.

Native discovery and operations

Commands in this section print pretty JSON on success unless the output column says otherwise. `URL` is the daemon base URL.

Command syntax	Options and selectors	Result
`manifest fetch URL`	Global HTTP options	Fetches `GET /aip/v1/manifest`; prints a manifest
`capability list URL`	`--capability-id`, `--text`, `--profile`, `--cursor`, `--limit`, `--signed-native`	Queries the tenant-scoped capability catalog; signed mode invokes `cap:aip:server:connector-capabilities-query`
`action call URL CAPABILITY`	`--input` plus shared action options	Sends one native action envelope and prints the response envelope
`action list URL`	`--state`, `--capability-id`, `--session-id`, `--principal-id`, `--approval-id`, `--transaction-id`, `--cursor`, `--limit`, `--include-results`, `--include-receipts`	Prints a durable action page
`action status URL ACTION_ID`	`--include-result`, `--include-receipts`, `--include-chunks`, `--wait-ms`	Prints one action view
`action result URL ACTION_ID`	`--wait-ms`, `--include-receipt`, `--include-terminal-events`	Prints the final-result view or the server's current response
`action events URL ACTION_ID`	`--cursor`, `--limit`, repeatable `--kind`, `--include-chunks`, `--follow`	Prints an event page; follow mode is requested but buffered rather than emitted incrementally
`action cancel URL ACTION_ID`	`--reason`	Posts a cancellation request and prints the response
`events list URL`	`--cursor`, `--limit`, repeatable `--kind`, `--follow`	Reads the global event stream
`session list URL`	`--principal-id`, `--status`, `--cursor`, `--limit`	Prints a session page
`session get URL SESSION_ID`	None	Prints one session
`session close URL SESSION_ID`	`--reason`	Closes one session and prints the response
`session resume URL SESSION_ID`	`--resume-token`, `--last-event-cursor`	Submits reconnect state and prints the response
`approval list URL`	`--cursor`, `--limit` default `100`, repeatable `--kind`	Reads approval lifecycle events; omitted kinds expand to the built-in approval set

Command syntax	Options and selectors	Result
`approval records URL`	`--status`, `--approver`, `--requester`, `--tenant-id`, `--cursor`, `--limit`, `--include-action-status`, `--include-receipts`	Prints an approval-record page
`approval get URL APPROVAL_ID`	`--include-action-status`, `--include-receipts`	Prints one approval record
`approval request URL REQUEST`	Inline JSON or `@path`	Sends an `ApprovalRequest` envelope and prints the response envelope
`approval decide URL DECISION`	Inline JSON or `@path`	Sends an `ApprovalDecision` envelope and prints the response envelope
`transaction get URL`	Exactly one of `--transaction-id`, `--plan-id`, `--action-id`; optional `--include-result`, `--include-receipts`	Prints one transaction view
`audit events URL`	`--action-id`, `--session-id`, `--principal-id`, `--transaction-id`, `--from`, `--to`, `--cursor`, `--limit`, `--include-receipts`, `--export`	Prints an audit page or authorized evidence export
`resource list URL`	`--capability-id`, `--kind`, `--cursor`, `--limit`	Prints a resource page
`resource read URL RESOURCE_ID`	`--version`, repeatable or comma-separated `--accept`	Prints one resource representation
`receipt get URL`	Exactly one of `--chain-id` or `--receipt-id`	Prints one receipt chain

`--follow` asks the native HTTP route for SSE. At this revision the shared GET helper buffers the bounded response rather than operating as a long-lived SSE client. Use the transport bindings and a streaming client when continuous consumption is required.

Shared action options

Both HTTP and NATS action calls accept:

Option	Wire effect		
`--action-id ACTION_ID`	Replaces the generated action ID		
`--idempotency-key KEY`	Sets the action idempotency key		
`--mode sync\`	async\	streaming`	Sets the requested action mode
`--identity JSON_OR_@FILE`	Decodes an `IdentityContext` into the action		
`--approval JSON_OR_@FILE`	Decodes an `ApprovalDecision` into the action		
`--transaction-mode MODE`	Creates the action transaction object		
`--transaction-id ID`	Adds a transaction ID when transaction mode is present		
`--plan-id ID`	Adds a plan ID when transaction mode is present		
`--compensation-for ACTION_ID`	Adds the compensated action when transaction mode is present		

Transaction mode values are `execute`, `dry-run`, `plan`, `commit`, `compensate`, and `rollback-not-supported`. The core protocol also defines `reconcile`, but this CLI enum does not expose it. Transaction detail options are serialized only when `--transaction-mode` is present.

These options construct claims; they do not grant identity, approval, transaction, or provider authority. The receiving deployment remains authoritative.

Native NATS commands

NATS commands use `SERVER_URL` plus `--trust-domain local`, `--service getaip-server`, and `--version v1` defaults. They use Core NATS request/reply and print the response envelope. A peer `Error` envelope makes the command fail.

Command syntax	Additional input
`manifest fetch-nats SERVER_URL`	Subject-routing options only
`action call-nats SERVER_URL CAPABILITY`	`--input`, shared action options, subject-routing options
`approval list-nats SERVER_URL`	`--cursor`, `--limit 100`, repeatable `--kind`, subject-routing options
`approval request-nats SERVER_URL REQUEST`	Inline JSON or `@path`, subject-routing options
`approval decide-nats SERVER_URL DECISION`	Inline JSON or `@path`, subject-routing options

Schema, conformance, and local verification

Command syntax	Input	Success output
`schema export DIR`	Output directory	Writes every built-in schema as pretty JSON; otherwise silent
`manifest validate FILE`	Manifest JSON	`manifest valid`
`conformance run`	Optional `--envelope FILE`	JSON containing `name`, `passed`, and `detail`
`receipt verify FILE`	Receipt-chain JSON	JSON containing `status`, `chain_id`, `receipt_count`, and recomputed `root_hash`

Manifest validation applies the built-in manifest schema. Receipt verification checks every previous-hash link, every receipt hash, and the chain root. Core conformance checks either the supplied envelope or a built-in manifest-request message body. These local checks do not prove deployment conformance or live provider behavior.

MCP compatibility commands

For `inspect`, `call-tool`, and `conformance`, select exactly one client transport:

- `--url URL` for Streamable HTTP; the CLI preserves a URL ending in `/mcp` or `/mcp/v1` and otherwise joins `/mcp`;
- `--command PROGRAM` plus repeatable `--arg VALUE` for a stdio subprocess.

MCP bearer authentication is independent of native HTTP authentication.

Command syntax	Options	Result
`mcp inspect`	Client transport and optional `--bearer-token`	Initializes the peer and prints its projected AIP manifest
`mcp call-tool TOOL`	`--arguments JSON_OR_@FILE`, client transport, optional token	Initializes the peer, calls one tool, and prints the result
`mcp conformance`	Client transport and optional token	Runs outbound MCP checks and prints the report
`mcp serve-stdio --url URL`	Optional `--bearer-token`	Exposes the HTTP MCP endpoint as a stdio MCP server until stopped
`mcp bridge --server stdio`	`--url URL`, optional token	Exposes an HTTP upstream over stdio until stopped
`mcp bridge --server http`	`--command PROGRAM`, repeatable `--arg`, `--bind 127.0.0.1:18090`	Exposes a stdio subprocess over Streamable HTTP until stopped

Connector qualification commands

`connector test URL` fetches the manifest, checks every repeatable `--capability CAPABILITY`, and optionally invokes every repeatable `--invoke-capability CAPABILITY` with common `--input`. Each invocation must return the `--expect-status` value, which defaults to `completed`.

The success JSON contains `status`, the manifest agent, the full capability list and count, and invoked capability statuses. This is a bounded interface check, not complete connector qualification.

`connector registry-migrate` installs or verifies immutable registry migrations:

```
SH
getaip connector registry-migrate \
  --database-url-file /run/secrets/registry-admin-url
```

Its pool defaults are four control connections, one data connection, and a 5,000 ms acquisition timeout. All three bounds must be greater than zero. The success JSON reports the component and installed schema version.

Connector registry commands

These are short-lived operator commands. They do not belong in a long-running gateway or connector-host process.

Command syntax	Required options	Result
`connector registry sign-evidence`	`--kind`, `--artifact-digest`, `--manifest-digest`, `--document`, `--signing-seed-file`, `--signer-identity`; optional `--valid-for-seconds 86400`	Prints signed release evidence
`connector registry sign-package`	`--package`, `--signing-seed-file`, `--signer-identity`	Prints a signed admission package
`connector registry plan`	`--package`, `--trust-policy`	Verifies without writing and prints package identity, digests, counts, and `write_performed: false`
`connector registry apply`	`--package`, `--trust-policy`, database options	Applies or resumes the exact verified package and prints its durable operation
`connector registry status`	`--package-id`, optional `--revision`, database options	Prints one durable admission operation; omitted revision selects the latest
`connector registry revoke`	`--package-id`, `--revision`, `--reason`, database options	Stops new traffic for the applied package and prints the retained operation

Evidence kinds are `oci-signature`, `sbom`, `provenance`, `conformance`, `vulnerability`, `license`, and `revocation`. Evidence validity must be between 60 seconds and 30 days. Signer identities must contain 1 to 512 bytes.

Database options shared by `apply`, `status`, and `revoke` are `--database-url-file`, `--control-max-connections 2`, `--data-max-connections 2`, and `--acquire-timeout-ms 5000`. Each numeric bound must be greater than zero.

Connector orchestration commands

The CLI derives and verifies platform-neutral operation plans. It does not execute the operations on Kubernetes, Nomad, ECS, Docker, or another workload platform.

Command syntax	Required options	Result
`connector orchestration plan`	`--package`, `--admission-trust-policy`, `--intent`, `--observed`, `--orchestration-policy`	Prints the next deterministic unsigned plan; performs no write
`connector orchestration sign`	Plan inputs plus `--signing-seed-file` and `--signer-identity`	Re-derives, signs, verifies, and prints the exact plan

Command syntax	Required options	Result
`connector orchestration verify`	`--plan`, `--current-observed`, `--orchestration-policy`	Verifies signature, limits, semantics, and the fresh observation fence; prints operation IDs and `write_performed: false`

The signing DID must be trusted by the executor policy. Verification is an executor precondition, not execution itself. See the connector orchestration guide for the external executor's responsibilities.

Output and exit behavior

The process returns status 0 only when the selected operation completes successfully. Clap parsing, file decoding, local validation, transport, authentication, authorization, protocol, registry, orchestration, and conformance failures print a diagnostic to standard error and return non-zero.

Most finite commands print pretty JSON to standard output. The documented exceptions are schema export, the concise manifest-validation line, the compact registry-migration JSON line, and long-running MCP proxies. Native ergonomic HTTP helpers wrap a successful non-JSON body as `{ "body": "..." }` before printing it. Scripts should use exit status first and parse JSON only for a command whose output contract above is JSON.

Security boundaries

- Prefer owner-only credential files over inline secret arguments.
- Do not treat a bearer token, signed request, or NATS connection as tenant or

policy authority by itself.

- Pin a peer DID when using signed native HTTP and protect the local clock used

by freshness checks.

- Keep registry administrator database credentials in short-lived operator

commands; hosts and gateways use narrower roles.

- Review signed admission and orchestration objects before passing them to an

external executor.

- Treat command output as potentially sensitive operational evidence.

Related reference

- Environment variables
- Native HTTP API
- Connector registry and routing
- Connector admission and supply-chain trust
- Errors