Cal.diy connector

Use the Cal.diy connector when an AIP application must read or change scheduling state through a typed, tenant-bound capability surface. The connector runs in its own aip-host-cal-diy process; product-neutral getaip-server routes admitted a

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/cal-diy/README.md

The reviewed connector exposes 81 operations from one pinned Cal.diy source revision. It adds AIP approval, idempotency, routing, evidence, and webhook boundaries around those provider operations. It does not expose every Cal.diy administrator, credential-provisioning, or compatibility route.

At a glance

Property	Reviewed value
Connector id	`cal-diy`
Standalone process	`aip-host-cal-diy`
AIP workspace version	`2.0.0`
Reviewed AIP source	`d7cce13d1d555644d04a4d73c66c95b113737635`
Pinned Cal.diy source	`f00434927386c9ecdcbd7e6c5f82d22044a245bc`
Connector profile	`aip.connector.cal_diy.v1`
Webhook profile	`aip.http.webhook.v1`
Capability prefix	`cap:cal_diy:`
Business operations	81
Provider authentication	Bearer token or OAuth client credentials
Execution	Synchronous; no connector streaming, async execution, or cancellation
Mutation policy	Approval and external-account-scoped idempotency required

The source revision identifies what was reviewed. It is not a claim that a different branch, rebuilt image, Cal.diy deployment, or provider revision has the same behavior.

Choose this connector when

The connector fits an integration that needs to:

- discover availability and reserve a slot;
- create and manage bookings with explicit approval;
- read or change event types, schedules, calendars, conferencing, and

webhooks;

- keep provider credentials in the connector deployment rather than in AIP

messages;

- route each authenticated tenant to an owner-admitted, account-specific host

without placing provider credentials in AIP messages;

- turn authenticated Cal.diy webhooks into durable AIP events;
- prevent a retry from silently repeating an uncertain provider mutation.

Use a provider-owned administrative integration instead when the task must create API keys, complete browser OAuth, administer OAuth clients, verify email or phone ownership, or invoke an upstream route outside the frozen catalogue.

Do not adopt the reviewed connector unchanged for a different Cal.diy revision. Compare the provider routes and schemas first, then repeat connector conformance and deployment qualification for the exact artifact.

Deployment boundary

The reference topology separates protocol routing from product execution:

1. An authenticated caller submits a native AIP action to `getaip-server`.
2. `getaip-server` applies protocol validation, identity, authorization, approval, and

runtime policy.

3. The connector registry selects an admitted `cal-diy` instance and healthy

replica for the tenant.

4. The remote connector path sends the action to `aip-host-cal-diy`.
5. The host uses its deployment-owned Cal.diy account and secret-file

configuration.

6. The connector maps the capability to one fixed Cal.diy method, path, and

API-version header.

7. The result returns through the host and central runtime with provider

credential material excluded.

The registry does not execute images or load product crates dynamically. An operator builds and deploys the Cal.diy host separately, then admits its exact identity, contract, artifact, and evidence. Adding another admitted instance does not recompile `getaip-server`.

Supported capability families

The family counts below come from the 81-entry `ALL_OPERATIONS` catalogue at the reviewed source revision.

Family	Count	Supported outcome
Profile	2	Read or update the authenticated profile
Event types and private links	15	Manage event types, private links, and event-type webhooks
Slots and reservations	5	Query availability and manage a short-lived reservation
Bookings	22	Read, create, reschedule, cancel, confirm, reassign, and inspect artifacts
Calendars and free/busy	21	Manage connections, events, busy time, destination, and conflict selection

Family	Count	Supported outcome
Conferencing	5	Discover, select, connect, or disconnect conferencing applications
Schedules	6	List, read, create, update, or delete schedules
User webhooks	5	List, read, create, update, or delete user-level webhooks
Total	**81**	Complete admitted business-operation catalogue

Each capability advertises both `aip.native.http.v1` and the connector profile. Its binding contains the exact provider method, path template, API version, and operation suffix. Input and output remain typed JSON; a provider response is bounded before it is parsed or returned.

Credential and tenant boundary

The standalone host accepts exactly one provider authentication mode:

- an owner-controlled Bearer-token file; or
- an OAuth client id with an owner-controlled client-secret file.

The connector does not perform an OAuth browser flow and does not refresh an OAuth credential through the agent surface.

The reviewed standalone host also requires one stable Cal.diy account id. It does not expose the connector library's optional tenant-to-account credential-handle composition. In the standalone fleet, bind each tenant to an admitted account-specific instance through owner-controlled registry topology; action input cannot choose another account or secret.

The connector library does implement a fail-closed credential-routing mode for specialized compositions and frozen conformance tests. That mode requires a trusted runtime tenant context, owner-defined bindings, and a credential provider. Do not assume it is active in `aip-host-cal-diy`, and do not adopt the migration-only bundled daemon as the target architecture merely to obtain its legacy command-line wiring.

Mutation safety

All non-GET operations require AIP approval and an idempotency key scoped to the external account. Reading booking recordings or transcripts also requires approval because the result may contain restricted meeting data.

Booking create, reschedule, and cancel require a validated plan before commit. Other mutations still support the connector transaction modes, but their dry run checks policy and input without claiming a complete downstream simulation.

Cal.diy does not provide universal provider idempotency for this catalogue. The connector therefore records a durable compare-and-set dispatch claim before a mutation can cross the provider boundary. Reusing the same key and input can return the stored result; reusing the key with different input is a collision.

If a crash or timeout leaves a dispatched mutation without trusted terminal evidence, the ledger marks the outcome uncertain. The connector does not blindly retry that mutation. Reconciliation must use the dispatch ledger, a trusted transaction checkpoint, or operator evidence before another provider effect is allowed.

Reads are retryable. Mutations are not advertised as retry-safe. Some create operations name a best-effort compensating delete or cancel capability, but compensation is a separately approved action, not an atomic rollback.

Signed event ingress

When webhook secrets are configured, Cal.diy sends a delivery to the standalone host route:

```
TEXT
POST /webhooks/cal-diy/{subscription_id}
```

The path selector chooses a deployment-owned secret reference. The host requires `X-Cal-Signature-256` and `X-Cal-Webhook-Version`, preserves the exact raw body, enforces a 1 MiB body limit, and passes a trusted receipt time to the connector.

The connector verifies the raw-body HMAC, the supported webhook version `2021-10-20`, the event-time window, and durable replay identity. Accepted events are appended locally and queued for central delivery. A replay resolves to the same deterministic event identity; a replay-store or event-store failure fails closed instead of acknowledging unretained state.

Creating or updating a provider webhook also requires its subscriber URL to match an operator-configured HTTPS prefix. No destination is accepted by default.

Deliberate exclusions

The source names 38 operator-only upstream routes that are not capabilities. They cover credential creation or refresh, browser OAuth and callbacks, OAuth client administration, calendar or conferencing connection setup, Stripe connection, and verified email or phone workflows.

Two deprecated singular calendar-event aliases are also excluded in favor of the canonical plural event routes.

These are security and compatibility boundaries, not missing catalogue rows. The connector also does not provide:

- arbitrary provider URL or method forwarding;
- raw credential input in an AIP action;
- dynamic product code loading in `getaip-server`;
- streaming results, connector-managed async execution, or cancellation;
- an automatic retry for an uncertain mutation;
- a production-readiness claim for an unqualified deployment.

Evidence status

The reviewed source contains connector contract tests and a frozen conformance suite for the complete catalogue. A retained historical isolated-live report also records an exact-upstream Cal.diy campaign, but that report does not prove the current source revision, a rebuilt host, or a new Cal.diy deployment.

Treat evidence labels narrowly:

Evidence	What it supports	What it does not support
Source and schemas	Implemented contract at the reviewed commit	Runtime behavior of an unbuilt artifact
Deterministic tests	Reproducible connector behavior under test fixtures	Execution against a real Cal.diy service
Historical isolated-live report	The named historical artifacts and upstream	Current deployment qualification
Deployment campaign	Only the exact identities and scenarios retained by that run	Other images, revisions, tenants, or topology

Choose the next document

The Cal.diy section is organized by task:

- start with `getting-started/quickstart.md` for the first admitted host and

read-only invocation;

- use `getting-started/authentication-and-tenant-routing.md` to choose a credential and account-isolation model;
- use `reference/configuration.md` for the complete host settings;
- open `capabilities/README.md` to choose a capability family;
- use the guides for booking and availability workflows;
- use the operations pages for deployment, health, recovery, and rollout;
- use `qualification/README.md` before making a readiness claim.

Use the global references below for fleet-wide routing, conformance, migration, and evidence boundaries.

Related documentation

- Connector documentation
- Pinned upstream baselines
- Conformance
- Historical Cal.diy isolated-live evidence
- Migrate bundled connectors