Connector catalog

Use this catalog to choose the product boundary that matches your integration. Each connector translates a bounded provider API and event surface into AIP capabilities, contracts, lifecycle state, and errors. The connector does not change t

SECTION	Connectors
TYPE	Documentation
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/README.md

The current public catalog contains six maintained product connectors. Each has a connector crate and a standalone host package at the reviewed source revision. Source presence establishes implementation scope only; qualification and live-product evidence remain separate claims.

This catalog targets source revision `d7cce13d1d555644d04a4d73c66c95b113737635`. The authoring pass inspected code and deployment artifacts without building a host, starting a service, or contacting a provider.

Choose a connector

Connector	Choose it when you need	Current provider boundary
Cal.diy	Scheduling, availability, bookings, calendars, conferencing, or booking webhooks	Versioned Cal.diy API operations, account-aware credentials, mutation fencing, reconciliation, and signed webhook ingress
Hermes Agent	Hermes API-server discovery, chat, responses, structured runs, sessions, or governed operator behavior	One or more configured Hermes endpoints with endpoint identity, optional tenant binding, and opt-in operator policy
Chatwoot	Customer-support accounts, conversations, messages, contacts, inboxes, teams, automation, or webhooks	A frozen account-scoped operation set, API-token authentication, and signed replay-fenced webhook ingress
Dify	Dify applications, workflows, conversations, media, human input, annotations, or knowledge resources	Separate application and knowledge-workspace credentials with bounded operation and response handling
CrewAI	Versioned crews, runs, status, events, cancellation, replay, training, testing, knowledge, or memory reset	An authenticated Rust-to-Python sidecar boundary with connector-owned AIP identity and lifecycle state
Twenty	Workspace CRM records, duplicate detection, aggregation, metadata, OpenAPI discovery, or webhooks	Fixed REST operations over validated object and resource identifiers, header-only authentication, governed mutations, and signed webhook ingress

These connectors are not arbitrary HTTP proxies. Their manifests, schemas, path construction, credential rules, risk, approval, idempotency, and response limits define the callable surface.

Choose by operational boundary

Product fit is necessary but not sufficient. Before adopting a connector, answer these questions:

1. Which exact provider account, workspace, application, endpoint, or sidecar

does one connector instance own?

2. Which tenant is authorized to discover and invoke that instance?

3. Which calls read data, mutate data, or permanently delete data?
4. Which mutations require approval, idempotency, reconciliation, or compensation?
5. Does the provider deliver webhooks or streaming output, and where is replay state stored?
6. How are provider credentials referenced, rotated, and revoked without entering the registry or manifest?
7. What exact implementation, qualification, and live-product evidence exists for the artifact you plan to deploy?

Open the connector overview before selecting capability IDs. The overview owns the provider boundary, deployment shape, upstream baseline, safety constraints, and evidence status for that connector.

Understand the common host model

All six product connectors can run behind the common standalone host boundary. The product-neutral `getaip-server` process routes admitted actions; the host owns the provider client, provider credentials, connector implementation, replica lifecycle, and product-specific ingress.

The common bootstrap provides registration, heartbeat, readiness, drain, offline transition, signed control-plane communication, gateway authentication, credential references, resource limits, and remote dispatch. Product hosts add only the configuration and HTTP mounts required by their provider.

Provider secrets are read by the host from deployment-owned files or secret references. They are not capability metadata, registry catalog values, or caller input. A connector instance is not ready merely because its process or health route is reachable; admission, registration, leases, credentials, and provider prerequisites also matter.

The legacy bundled daemon remains a migration boundary. New fleet deployments should not compile product-specific connector clients into the product-neutral daemon.

Read source and evidence cues correctly

Connector	Source-side evidence available at this revision	What remains a separate claim
Cal.diy	Connector and host source plus a dedicated isolated qualification harness	Whether a named Cal.diy artifact completed that harness and remains compatible now
Hermes Agent	Connector and host source plus a two-endpoint smoke topology	Whether an exact Hermes artifact passed the intended operation and operator matrix
Chatwoot	Connector and host source plus the deterministic five-product fleet profile	Live Chatwoot behavior, isolated qualification, and production operation
Dify	Connector and host source plus the deterministic five-product fleet profile	Live Dify behavior, isolated qualification, and production operation
CrewAI	Connector, host, sidecar, and deterministic fleet-sidecar source	Qualification of an exact crew registry, sidecar image, recovery path, and provider dependencies
Twenty	Connector and host source	A dedicated example, reference-fleet product profile, isolated qualification, or live-product run

The deterministic product-fleet profile uses controlled upstream fixtures for Cal.diy, Hermes Agent, Chatwoot, Dify, and CrewAI. It tests fleet and connector mechanics but does not constitute live verification against those products. Twenty is not part of that profile at the reviewed revision.

Do not copy an evidence statement from one connector to another. Record the connector version, upstream revision, image digest, configuration identity, tenant, procedure, timestamp, and retained result for every

qualification or live claim.

Navigate connector documentation

Opening a catalog entry changes from global protocol documentation to the selected connector's local documentation context. Each connector keeps the stable overview route shown above and organizes deeper pages by reader task:

Local section	Use it for
Overview	Product boundary, version, deployment choice, capability groups, and evidence
Getting started	Credentials, tenant routing, first discovery, and first bounded call
Capabilities	Operation groups, contracts, inputs, outputs, side effects, and errors
Guides	Product workflows that combine several operations safely
Reference	Configuration, provider routes, webhooks, limits, and exclusions
Operations	Deployment, readiness, metrics, recovery, and credential rotation
Troubleshooting	Symptom-led diagnosis and outcome-unknown decisions
Qualification and changelog	Exact evidence, compatibility, migrations, and deprecations

The global sidebar keeps one Connector Catalog entry instead of listing every product page. This keeps navigation usable as the catalog grows. Connector-local menus can expand independently without changing protocol and operator navigation.

Add or migrate a connector

Use Build a connector when no catalog entry owns the provider boundary. A new connector needs an immutable identity, bounded provider surface, source-owned schemas, contracts, secret references, standalone host composition, negative tests, and qualification scope before it can join the public catalog.

Do not add an arbitrary passthrough connector as a shortcut. Dynamic long-tail catalogs still need explicit authority, path, schema, side-effect, credential, and evidence boundaries.

Related documentation

- Profiles and connectors
- Connector-fleet quickstart
- Examples and reference deployments
- Build a connector
- Installation