Migrate bundled connectors to the fleet

Use this guide to move one product connector from an aipd process into an admitted standalone connector host. It is for operators and connector owners who can control the gateway, registry, host, provider credentials, ingress, and rollback

SECTION	Connectors
TYPE	Guide
PROTOCOL	AIP 1.0
SOURCE	docs/guides/migrate-bundled-connectors.md

The procedure preserves the AIP 1.0 capability contract while changing where the provider operation executes. It keeps one authoritative mutation path at a time, verifies the new path with non-mutating work, and switches one tenant binding only after the old path has drained.

The pinned source contains a compatibility binary named `aipd-legacy-bundled`, a product-neutral `aipd`, and standalone hosts for the six public connectors: Cal.diy, Hermes Agent, Chatwoot, Dify, CrewAI, and Twenty. It does not contain a generic command that imports bundled connector state into a host database.

This guide reflects source revision `97be86e9efedf07ecf1783b03800f683f107fb04`.

Use this procedure for one execution boundary

Apply this migration when a connector implementation or trusted local module runs in the same process as the gateway and the target connector already has:

- a frozen connector implementation;
- a standalone host binary;
- an immutable host artifact and exact digest;
- a reviewed manifest and implementation support map;
- the evidence required for fleet admission;
- a tenant-owned provider account or endpoint boundary.

If the connector still uses legacy SDK traits, complete Upgrade a connector to the frozen SDK first. If no standalone host or admission package exists, follow Build a connector.

Do not combine this migration with capability renaming, schema changes, provider account consolidation, credential-scope expansion, or a wire-version change. Each would make it harder to distinguish a deployment failure from a contract change.

Migrate one connector type, tenant, and provider boundary at a time. A deployment may contain fewer than the six public connectors; do not create an unused instance merely to make the fleet catalog look complete.

Prepare authority and a rollback window

Assign an owner for each control before changing traffic:

Control	Required authority
Bundled process	Remove or restore the connector configuration and restart safely
Product-neutral gateway	Configure registry access, signing, trust, and limits
Connector registry	Plan and apply a signed admission package and change bindings
Standalone host	Deploy the admitted digest, inspect readiness, drain, and stop it
Provider account	Rotate credentials and change webhook or callback destinations
Durable stores	Retain, inspect, and restore gateway, host, and ingress state
Client traffic	Pause or bound mutations and run an approved read-only check
Evidence	Record identities, revisions, timestamps, outcomes, and rollback decisions

Choose a window long enough for the connector's maximum action duration, provider job duration, retry window, callback delay, and webhook replay window. Stop before migration if active work cannot be enumerated or if the provider cannot distinguish completed, failed, and unknown mutations.

Retain these rollback inputs outside the deployment:

- the current gateway artifact and complete non-secret configuration;
- canonical bundled manifest and capability schemas;
- provider credential references and their revisions, without secret values;
- provider ingress destinations and signing-secret references;
- durable-store backup identities and restoration procedure;
- the signed fleet admission package and trust policy;
- the proposed tenant binding revision;
- a list of active and recently completed action IDs.

Do not put database URLs, signing seeds, provider tokens, OAuth secrets, webhook secrets, or private trust roots in the migration record.

Inventory the bundled boundary

Capture a baseline from the exact deployment you will migrate. The `aipd-legacy-bundled` source preserves the older command-line surface, normalized readiness fields, default manifest, and default MCP tool catalog as compatibility fixtures. That source-level compatibility is useful for a baseline, but it does not prove that a particular deployment is healthy or that a fleet artifact is admitted.

Record the following for the selected connector:

Boundary	Record before cutover
Contract	Connector ID, capability IDs, kinds, schemas, risk, side effects, and manifest digest
Implementation	Source revision, artifact identity, and supported operation claims
Provider scope	Tenant, account, workspace, application, or endpoint set
Configuration	Non-secret values, defaults, limits, and owning process
Credentials	Opaque reference, issuer, scopes, account, rotation, and expiry
Runtime state	Active actions, idempotency reservations, transactions, checkpoints, and results
Product state	Provider operation IDs and unresolved or asynchronous jobs
Ingress state	Destination, signatures, timestamp policy, replay store, and pending deliveries
Egress state	Callbacks, streams, queued events, retries, and delivery acknowledgements

Boundary	Record before cutover
Routing	Clients, tenant, capability, and the current authoritative execution path

Compare the bundled manifest with the manifest generated by the standalone connector under the target configuration class. Capability IDs, schemas, side-effect declarations, identity requirements, idempotency behavior, approval requirements, and transaction behavior should remain unchanged.

If the old contract overclaims behavior, handle that correction as a separate reviewed release. Do not hide the contract change inside a process migration.

Separate gateway and product configuration

The target gateway is product-neutral. It owns AIP edge authentication, verified tenant identity, runtime state, registry data-plane access, remote dispatch policy, scheduling limits, and central event ingress. It does not own provider credentials or product-specific endpoints.

The standalone host owns the connector's bounded provider configuration. All six public host binaries flatten the common `ConnectorHostBootstrapArgs` and then add product-specific arguments. Move values according to ownership:

Bundled value	Target owner
Provider base URL or product endpoint inventory	Product host
Provider token, OAuth secret, sidecar token, or webhook secret	Owner-controlled host file
Tenant-to-provider account mapping	Host configuration plus admitted instance identity
Webhook replay storage	Product host durable boundary
Gateway edge identity and client trust	Product-neutral `aipd`
Connector registry data-plane URL	Product-neutral `aipd` owner-only file
Gateway fleet signing seed	Product-neutral `aipd` owner-only file
Host, control-plane, and gateway DIDs	Deployment trust configuration
Connector type, version, instance, replica, and binding	Signed registry admission package

Do not translate a secret file into an environment value or command-line literal. Preserve the credential's scope and provider account while changing which process reads the file. Rotate the credential if the old process cannot be prevented from using it after cutover.

The common host boundary also requires exact public and control-plane endpoints, a runtime database URL file, and a signing-seed file. It also requires connector type and version, instance and replica IDs, a verified tenant, central gateway identity, artifact digest, and a non-secret secret-provider reference. These values must match the admitted records; they are not labels that the host may invent at startup.

Prepare and admit the standalone host

Build the host from the reviewed source and retain its immutable digest. Use the manifest and implementation support generated for the same artifact and configuration class.

Prepare a signed admission package containing:

- the connector type and immutable version;
- the exact artifact and manifest digests;
- implementation support for each callable capability;
- a tenant-owned connector instance;

- a pre-provisioned offline replica;
- an initially disabled tenant capability binding;
- bounded admission policy;
- valid OCI signature, SBOM, provenance, conformance, vulnerability-policy,

license-policy, and revocation-observation evidence.

Verify the package without writing:

```
SH
aipctl connector registry plan \
  --package signed-admission.json \
  --trust-policy admission-trust-policy.json
```

Apply that exact package only after the plan output matches the intended type, version, instance, replica, tenant, capabilities, and disabled binding:

```
SH
aipctl connector registry apply \
  --package signed-admission.json \
  --trust-policy admission-trust-policy.json \
  --database-url-file registry-database-url
```

Do not give the host or gateway registry-administrator credentials. The gateway uses the registry data plane for catalog and route resolution. The standalone control-plane service owns the narrow signed register, heartbeat, drain, and offline lifecycle boundary.

Prepare the product-neutral gateway

Configure the product-neutral `aipd` with:

- an owner-only connector-registry data-plane URL file;
- an owner-only Ed25519 signing-seed file;
- either an explicit host allowlist or trust in admitted registry endpoints;
- TLS trust for private PKI when applicable;
- explicit HTTP or private-network exceptions only for controlled development;
- request timeout, response bound, retry budget, client-pool bound, topology,

and admission limits appropriate to the deployment;

- a central callback URL when remote streaming is enabled;
- bounded connector-event ingress limits.

Fleet configuration is atomic at startup. The pinned implementation rejects a partial fleet configuration, requires the registry URL and signing seed, and requires either a host allowlist or registry-endpoint trust.

When enabled, the gateway installs the registry-backed capability catalog, remote connector handler, fair admission scheduler, fleet status provider, and signed connector-event ingress as one product-neutral composition. Product connector code is not part of that composition.

Start or restart the product-neutral gateway without enabling the new tenant binding. Confirm its normal readiness and registry connectivity before you start the product host. Do not infer host eligibility from gateway health.

Decide what happens to existing state

The standalone host opens its own durable PostgreSQL runtime. The pinned source does not expose a generic bundled-to-host state importer. Do not copy tables, serialized files, replay databases, or runtime directories between unrelated stores unless the selected connector has a separately reviewed migration tool and rollback procedure.

Use this decision table:

State class	Migration action
Completed gateway action history	Keep in the existing gateway store; do not replay provider work
Active synchronous action	Let it reach a known terminal state before switching
Active provider job	Retain its provider operation ID and reconcile it through the old owner
Unknown mutation outcome	Stop cutover until the provider state is reconciled
Idempotency reservation	Keep the old path authoritative until its result is known or its policy expires
Transaction or compensation state	Finish or explicitly abort under the original execution owner
Pending stream or callback	Drain and verify acknowledgement before disabling the old path
Webhook replay record	Keep the old ingress authoritative until the destination switch is complete
New host runtime	Start empty unless a connector-specific, reviewed importer says otherwise

An identical idempotency key in two independent stores is not a shared reservation. It can result in two provider calls. Pausing new mutations and draining the old owner is therefore a safety control, not an optional cleanup.

Start the host without routing traffic

Start one replica with the exact admitted identity and artifact digest. The common host preparation validates its configuration and trust, discovers the connector manifest, and opens durable storage. Serving recovers durable work, registers the replica, renews its lease, and exposes:

Route	Cutover use
`GET /health`	Confirms process and protocol identity
`GET /ready`	Confirms valid lease, storage, connector, and non-draining state
`GET /metrics`	Shows bounded host lifecycle and request signals
`GET /aip/v1/manifest`	Exposes the exact running manifest for comparison
`POST /aip/v1/messages`	Accepts signed, route-pinned AIP execution from the gateway

Require `/ready`, not `/health`, before considering the replica eligible. Also confirm that the registry reports the expected type, version, instance, replica, artifact digest, endpoint, tenant, lease, and capacity.

Keep the tenant binding disabled. Registration and readiness prove host lifecycle state; they do not authorize tenant traffic.

Shadow only non-mutating behavior

Compare the old and new boundaries before cutover without giving both authority to mutate the provider.

Safe comparisons include:

1. canonical manifests and capability schemas;
2. connector discovery and implementation support;
3. `/health`, `/ready`, and bounded metrics;
4. deterministic provider fixtures;

5. provider reads that the capability contract classifies as non-mutating;
6. error mapping for rejected or malformed inputs;
7. tenant, account, credential issuer, and trust identities in retained traces.

Do not shadow creates, updates, deletes, messages, bookings, long-running jobs, or any operation whose side effects are uncertain. A dry-run label is not sufficient unless the connector and provider contract both guarantee that no provider effect occurs.

Record differences by contract field. Stop if the new host publishes a missing or extra capability, a changed schema, weaker approval or idempotency behavior, an unexpected provider account, or an implementation claim that the host does not exercise.

Cut over one tenant binding

Use one controlled sequence:

1. Pause new mutations for the selected tenant and capabilities.
2. Let bundled actions, callbacks, streams, and provider jobs reach known

states.

3. Reconcile every unknown provider outcome under the bundled owner.
4. Disable the bundled connector configuration or replace the compatibility

process with product-neutral `aipd` for this boundary.

5. Restart as required and confirm the bundled capability is no longer handled

locally.

6. Change provider webhook or callback delivery to the standalone host when the

connector uses ingress.

7. Enable the reviewed tenant capability binding at a new policy revision.
8. Confirm that registry discovery exposes the capability only to the intended

tenant and selects the expected instance.

9. Resume one approved non-mutating request through the normal client path.
10. Resume bounded mutations only after the read path, lease, metrics, and

retained trace identify the admitted replica.

The tenant capability binding is the fleet routing authorization. A route assignment then pins the action to an exact instance, ready replica, endpoint, peer identity, connector version, manifest digest, catalog revision, binding policy revision, credential revision, and assignment fence.

Do not leave the same capability active as both a local bundled handler and a fleet binding. Local and remote state do not share an idempotency reservation, and traffic precedence is not a migration control.

Verify the cutover

Verify the result at four boundaries:

Boundary	Required observation
Contract	Manifest digest and capability behavior match the reviewed target

Boundary	Required observation
Routing	Tenant binding and retained assignment identify the intended instance and replica
Lifecycle	Host is ready, lease is current, and active assignments settle
Provider	The intended account records one effect for one mutation and no duplicate from the old path

Also confirm:

- the bundled process no longer reads the migrated provider credential;
- only the selected ingress destination receives new provider deliveries;
- old callback and webhook queues are empty or explicitly retained;
- gateway and host logs contain no secret values or unbounded provider bodies;
- retry, cancellation, approval, and reconciliation follow the published

contract;

- an action status lookup reads stored state without repeating provider work;
- the migration record distinguishes source review, artifact admission,

conformance, deployment observation, and live-provider evidence.

Observe for at least the longest relevant action, provider job, callback, and webhook replay window before declaring the old boundary removable.

Resolve cutover failures

Symptom	Decision
Fleet options are rejected at gateway startup	Complete the registry, signing, and host-trust set; do not bypass validation
Host health succeeds but readiness fails	Inspect storage, connector health, lease, identity, artifact, and control-plane trust
No tenant binding is found	Verify the tenant, capability, instance, enabled flag, and policy revision
No ready replica is available	Keep mutations paused and inspect lease, drain state, capacity, and endpoint admission
Manifest differs from the baseline	Stop; determine whether configuration or contract changed
First request reaches the wrong provider account	Disable the binding and rotate or correct the credential mapping
A mutation times out after dispatch	Reconcile the provider operation; do not retry through the bundled path
Both ingress paths receive one event	Disable one destination and reconcile replay state before resuming delivery
The old process still has active work	Restore its authority only long enough to drain or reconcile that work

Do not repair a failed migration by deleting route assignments, replay records, idempotency state, or provider operation IDs. Those records are evidence needed to prevent a second effect.

Roll back without double execution

Rollback changes future routing; it does not erase provider effects already created by the new host.

1. Pause new mutations.
2. Disable the fleet binding so no new assignment selects the host.
3. Begin host drain and wait for active assignments to settle.
4. Reconcile every non-terminal or unknown provider operation through the host

that created it.

5. Mark the replica offline through the lifecycle boundary and stop it.
6. Restore the previous provider ingress destination only after the new ingress

queue is drained or retained.

7. Restore the exact bundled artifact, configuration, credential revision, and

durable store.

8. Confirm the local capability and provider account before resuming a

non-mutating request.

9. Resume mutations only after the rollback owner confirms there is no

overlapping work.

Keep the admitted fleet records and failed migration evidence until the review is complete. Revoke an artifact or credential when the failure concerns its integrity or exposure; disabling a tenant binding alone does not invalidate either.

Repeat with a fresh boundary

Accept the first connector migration before starting another. Create a new inventory, state decision, signed admission package, binding revision, verification record, and rollback decision for each connector and tenant boundary.

The six public connectors have separate host packages and product-specific configuration. A result for Cal.diy does not prove Hermes Agent, Chatwoot, Dify, CrewAI, or Twenty. Carry forward only the process controls; collect new contract and provider evidence for every host artifact.

Related documentation

- Profiles and connectors
- Identity and trust
- Connector catalog