Rotate connector credentials

Use this guide to replace one provider credential used by a standalone connector host. The procedure creates a new tenant-owned connector instance, starts replicas with the new material, changes new route assignments to the new credential r

SECTION	Deploy and Operate
TYPE	Guide
PROTOCOL	AIP 1.0
SOURCE	docs/guides/connector-credentials-and-rotation.md

This is the safe default for Cal.diy, Hermes Agent, Chatwoot, Dify, CrewAI, and Twenty at source revision `97be86e9efedf07ecf1783b03800f683f107fb04`. Their host binaries read provider API, OAuth, sidecar, or webhook secrets during process startup. The common credential revision policy reloads dynamically, but it does not reload or select that product-owned secret material.

Use the emergency procedure when the old credential may be compromised. It stops new work and fails old revisions closed before availability or graceful completion.

Understand the three credential layers

Do not begin a rotation until the change record distinguishes these objects:

Layer	Stored where	Purpose	Contains secret material
Provider material	Product-owned secret files or external secret provider	Authenticates the connector to the external product	Yes
Opaque credential handle	Host configuration	Names an issuer, handle, tenant, and verified scopes	No
Credential revision reference	Tenant binding, durable route assignment, and host policy	Fences which credential generation may perform a provider side effect	No

The registry copies a binding's `credential_revision_ref` into each new durable route assignment. Resolving the same action ID again returns the existing assignment, including its original credential revision and exact replica. A later binding update affects new actions only.

Immediately before a provider side effect, the connector host checks the revision from the signed route against its current policy. An authorized revision permits execution. Authorization does not fetch a secret, replace a file, or prove that the mounted material matches the label.

Choose the supported rotation shape

Use a blue/green instance rotation for the six public hosts:

```
TEXT
old binding -> old instance -> old replicas -> old provider material
new binding -> new instance -> new replicas -> new provider material
```

Keep the new binding disabled while the new replicas register and become ready. At cutover, stop admission of new actions, disable the old binding, enable the new binding, verify the catalog, and resume admission. Existing action IDs stay pinned to old replicas; fresh action IDs receive the new revision.

Do not place old-secret and new-secret replicas under one instance during this rotation. The registry can send any new assignment for that instance to any ready replica. Because the six current hosts load one product credential set at startup, a mixed instance could label work with one revision while a replica uses material from the other.

A same-instance rolling rotation is suitable only for a separately reviewed connector whose credential provider resolves material by the pinned revision and can serve both revisions concurrently. The six current public host entrypoints do not establish that behavior.

Prerequisites

Before a routine rotation, require:

- the exact active connector type and immutable version;
- the old instance, replica, binding, policy, and credential revision IDs;
- a new opaque revision ID containing 1–256 bytes;
- a new provider credential created through the provider's approved process;
- owner-controlled secret files or an external secret-provider revision;
- new instance and replica IDs with reviewed endpoints and signing identities;
- a complete signed admission package stream and its trust policy;
- short-lived connector-registry administrator access;
- a way to pause new action admission for the affected tenant and capability;
- a non-mutating provider check and expected result;
- visibility into active assignments, host readiness, errors, and provider

audit records;

- a rollback window during which the old credential and old replicas remain

available.

Use a distinct revision name that cannot be confused with a file version, deployment timestamp, or mutable secret alias. Record a cryptographic or provider-owned secret version identifier in the private change record, but do not place it in public logs if it reveals provider internals.

Record the current boundary

Capture a secret-free baseline before changing the provider or registry:

Item	Required value
Tenant and capability	Exact binding key
Old instance	Tenant owner, external account, immutable connector version, and status
Old binding	Priority, policy revision, credential revision, quota policy, and enabled state
Old replicas	Endpoint, peer DID, trust domain, topology, status, lease, and active assignments
Old host policy	Current, accepted previous, and revoked revision sets
Product boundary	Account or workspace and required scopes, without secret bytes
Evidence	Last known safe read, provider audit marker, and observation time

Stop if the binding revision, instance owner, external account, or host policy does not match the approved record. First reconcile the unknown change; do not hide it inside a credential rotation.

Create and protect the new provider material

Create the replacement in the external provider with the smallest scopes required by the admitted capabilities. Keep the old credential active. Store the new material through the product-specific mechanism documented for that connector.

For the current six host binaries, secret inputs are regular non-symlink files with no group or world permission bits on Unix. Mount a new immutable secret version rather than overwriting the path used by a running old replica. Do not print, hash into a public log, pass on a command line, or include the material in an admission package.

API credentials and inbound webhook verification secrets are separate trust boundaries. Rotate only the intended boundary. If both must change, use the connector-local runbook to preserve any provider-supported webhook overlap and record two independent verification results.

Prepare the new host policy

Configure each new replica with the complete common credential tuple:

```
TEXT
credential id       = opaque handle for the new material
credential issuer   = approved internal issuer
credential scopes   = exact verified provider scopes
credential revision = credential-revision-new
tenant              = the new instance tenant
```

Set `--credential-revision-policy-file` to a non-secret JSON file that authorizes the new revision:

```
JSON
{
  "current_revision_ref": "credential-revision-new",
  "accepted_previous_revisions": [],
  "revoked_revisions": []
}
```

The file rejects unknown fields and overlapping sets. Replace it atomically on the same filesystem; a partial, unreadable, or invalid document makes provider work unavailable. The common host validates at startup that this policy authorizes the configured revision and reloads it before every provider side effect.

Do not add the old revision to a new replica's policy merely to make a health check pass. A new replica that lacks the old provider material must not claim it can execute old-revision work.

Admit the green instance with traffic disabled

Create the next monotonic admission package revision. It remains a complete, signed package with the immutable connector version and required evidence; it is not a secret-rotation patch format. Include:

- the new tenant-owned instance with the same intended external account

boundary;

- new pre-provisioned replicas in `offline` state with zero active assignments;
- one binding per affected capability for the new instance;
- a higher binding `policy_revision` than any existing record for the same

tenant, capability, and instance key;

- `credential_revision_ref` set to the new opaque revision;
- `enabled` set to `false`;
- the reviewed priority and quota policy.

Plan the signed package without writing:

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

Compare the returned package ID, revision, digest, instance count, replica count, and binding count with the approved change. Apply only those exact bytes:

```
SH
aipctl connector registry apply \
  --package rotation-stage-signed.json \
  --trust-policy admission-trust-policy.json \
  --database-url-file /run/operator/registry-admin.url
```

Then read the durable operation record:

```
SH
aipctl connector registry status \
  --package-id "$PACKAGE_ID" \
  --revision "$PACKAGE_REVISION" \
  --database-url-file /run/operator/registry-admin.url
```

Require an applied state. Close and unmount administrator access after the short-lived operation. The disabled binding must not appear in tenant-visible capability discovery.

Start and verify the green replicas

Start the new hosts with their new instance, replica, configuration revision, credential handle, credential revision, policy file, and provider secret paths. Do not reuse an old replica ID or signing seed.

For every new replica, verify:

1. `/health` identifies an AIP 1.0 connector host;
2. `/ready` reports a valid lease, no drain, durable storage ready, and product

connector ready;

3. registry state matches the admitted endpoint, peer DID, trust domain,

artifact digest, topology, and configuration revision;

4. the deployment secret mount contains the intended private version without

exposing its bytes;

5. the provider's approved non-mutating credential check succeeds;
6. a deliberately invalid or under-scoped credential check fails as expected;
7. no production tenant binding routes new work to the green instance yet.

Host readiness is necessary but insufficient. A product health implementation may not exercise every scope or operation. Retain the provider audit record and the connector-local check result as separate evidence.

Prepare the cutover package

Create another signed admission package revision containing both binding transitions for every affected capability:

Binding	Instance	Credential revision	Enabled	Policy revision
Old	Old instance	Old revision	`false`	Greater than its current value
New	New instance	New revision	`true`	Greater than its staged value

Include both referenced instances in the package. Keep the same exact tenant, capability IDs, quota intent, and reviewed priority semantics. A missing old binding does not delete or disable it; the package must explicitly write its

new disabled state.

Run `aipctl connector registry plan` and compare the signed digest with the change record. Do not use `connector registry revoke` for routine rotation: that command revokes the entire connector version represented by an applied package, not one credential.

Cut over new assignments

Admission package catalog writes are individually durable and idempotent, but the complete binding list is not committed as one database transaction. Therefore perform the binding transition inside a bounded admission pause:

1. stop acceptance of new actions for the affected tenant and capabilities;
2. allow already accepted requests to obtain or reuse their durable

assignments;

3. apply the exact cutover package with short-lived administrator access;
4. require the package operation to report `applied`;
5. query tenant-scoped capability discovery and confirm that only the green

enabled binding is eligible;

6. submit one new non-mutating action with a fresh action ID;
7. confirm its durable assignment names the green instance, a green replica,

and the new credential revision;

8. confirm the external provider observed the intended account and scope;
9. resume new action admission.

Stop and keep admission paused if discovery is empty, both instances remain eligible, the route carries the wrong revision, or the provider result cannot be attributed safely. Do not test cutover with a mutation.

Preserve and retire the old path

Keep the old replicas, host policy, and provider credential available while old durable assignments can still resume. Disabling the old binding prevents new assignments; it does not rewrite or delete an existing action's pinned route.

Track old assignments until every item is terminal or explicitly reconciled. Include queued, running, awaiting-approval, retryable, delegated, and streaming work in that inventory. A low current in-flight gauge alone does not prove that no durable retry can return.

After the retention boundary is satisfied:

1. drain each old replica through the signed lifecycle service;
2. verify it accepts no new work and reaches `offline`;
3. preserve the secret-free assignment and provider audit records;
4. revoke or delete the old provider credential through the provider's

approved control plane;

5. remove the old secret mount and backups according to retention policy;
6. retain the old binding as disabled audit state unless a separate registry

lifecycle procedure owns deletion;

7. close the rotation record with the new binding, route, provider, and

retirement evidence.

Roll back a routine rotation

Rollback is safe only while the old credential and old replicas remain usable. Pause new admission again and apply a new package revision that disables the green binding and re-enables the old binding with higher policy revisions. Do not reuse a previously applied package revision or edit its signed bytes.

Verify that a fresh non-mutating action pins the old instance and old credential revision before resuming admission. Actions already assigned to the green instance remain pinned there. Keep green replicas available until those actions settle, or classify each unresolved provider effect before containment.

A binding rollback does not undo an external provider mutation. Use action idempotency, receipts, provider audit data, and the connector-local recovery procedure to decide whether an uncertain action completed. Never retry an ambiguous mutation with a new action ID merely because routing changed.

Revoke a suspected compromised credential

When confidentiality or integrity may be lost, prioritize containment:

1. pause affected new action admission;
2. atomically replace the revision policy on every reachable old host so the

old revision appears only in `revoked_revisions`;

3. apply a signed package revision that disables every binding carrying the old

revision;

4. revoke the provider credential through the provider control plane;
5. drain or isolate old replicas;
6. inventory actions that may already have passed the pre-side-effect revision

check;

7. reconcile their provider outcomes before retry or compensation;
8. deploy and enable a green instance only after new material and scopes are

verified.

A revoked revision returns HTTP 403 with `connector_host.credential_revision_denied`; it is non-retryable. An unreadable or invalid policy returns HTTP 503 with `connector_host.credential_revision_unavailable`; it is temporary and suggests a 1,000 ms retry. During a compromise, do not treat an unavailable policy as permission to bypass the check.

The policy check occurs immediately before the host passes the action into the gateway. Work already beyond that boundary may have reached the provider. Record those actions as potentially ambiguous until provider evidence resolves them.

Use package-wide revocation only when the entire admitted connector version must stop. It changes the version status to `revoked` and affects unrelated instances using that exact version.

Completion criteria

Close the rotation only when all of the following are true:

- fresh actions pin the green instance, replica, and credential revision;

- the provider confirms the expected account and least-privilege scopes;
- tenant discovery exposes no enabled old binding;
- old assignments are terminal or individually reconciled;
- old replicas are offline;
- the old provider credential is revoked;
- no secret value appears in packages, logs, metrics, receipts, or retained

screenshots;

- rollback or emergency decisions and their evidence are retained;
- qualification and production-readiness claims remain limited to their actual

evidence.

For exact common fields and policy-file constraints, see Connector host configuration. Use Deploy the connector fleet for admission and host startup, Identity and trust for the credential boundary, Deploy AIP in production for operational ownership, and Connector catalog for the six product-specific configuration and recovery paths.