Configure Cal.diy authentication and tenant routing

Use this procedure to give one aip-host-cal-diy instance access to one Cal.diy account and make that account reachable only through its admitted AIP tenant binding. The provider secret stays in the host deployment; an AIP action cannot sele

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/cal-diy/getting-started/authentication-and-tenant-routing.md

The reviewed standalone host uses a one-tenant, one-account model. Deploy a separate admitted instance when another tenant or Cal.diy account needs a different ownership boundary.

Understand the three trust layers

Do not treat these identities as one credential:

Layer	Establishes	Does not contain
AIP caller authentication	Which principal reached product-neutral `aipd`	Cal.diy token or client secret
Fleet and host identity	Which admitted tenant, account, instance, replica, and gateway may use the host	Raw provider secret
Cal.diy provider authentication	Which Cal.diy account authorizes the fixed provider request	Caller-selectable tenant or account

The central gateway authenticates the caller and chooses an eligible instance from the registry. The connector host accepts messages only from its configured gateway principal and signing identity. It then replaces identity assertions from the action payload with its own verified tenant membership, external account reference, and optional non-secret credential metadata.

Finally, the Cal.diy client reads the provider credential chosen by the host operator. Changing `identity`, `tenant`, `account_id`, or credential-looking fields in action input does not change that client credential.

1. Choose the provider authentication mode

The standalone host accepts exactly one of two modes.

Mode	Configure	Provider headers	Appropriate source material
Bearer	Token file	`Authorization: Bearer …`	API key, managed-user token, or OAuth access token
Cal platform client	Public client id and client-secret file	`x-cal-client-id` and `x-cal-secret-key`	Cal platform client credentials

The client-header mode is not an OAuth token exchange. The connector sends the two configured headers directly. It does not open a browser flow, call the provider token endpoint, or refresh an OAuth credential.

Choose the smallest provider identity and permission set that covers the admitted capability families. Do not use a platform administrator credential merely because the connector's frozen catalogue prevents arbitrary routes.

2. Provision the provider secret file

Create the secret through the deployment's secret manager and mount it as an owner-controlled file. The host requires a regular, non-symlink file. On Unix, group or world permissions cause startup to fail.

For a pre-provisioned mount point, the expected permission shape is:

```
TEXT
-rw------- 1 <host-uid> <host-gid> ... /run/secrets/cal-diy-provider
```

The file must contain one non-empty UTF-8 value after surrounding whitespace is trimmed. The Cal.diy host reads at most 16 KiB. The bootstrap reader keeps the temporary string in zeroizing memory, and the connector redacts secret material from its debug representation and errors.

Do not place the secret in:

- an action payload, manifest, admission package, or registry record;
- a command-line argument or ordinary environment variable;
- an image layer, Compose file, example, log, or retained qualification

artifact;

- a provider base URL as user information.

3. Configure the Cal.diy client boundary

Set a fixed provider origin and stable account id for the instance.

For Bearer authentication:

```
TEXT
AIPD_CAL_DIY_BASE_URL=https://cal.example.test
AIPD_CAL_DIY_ACCOUNT_ID=calendar_primary
AIPD_CAL_DIY_BEARER_TOKEN_FILE=/run/secrets/cal-diy-provider
```

For Cal platform client headers:

```
TEXT
AIPD_CAL_DIY_BASE_URL=https://cal.example.test
AIPD_CAL_DIY_ACCOUNT_ID=calendar_primary
AIPD_CAL_DIY_OAUTH_CLIENT_ID=client_public_id
AIPD_CAL_DIY_OAUTH_CLIENT_SECRET_FILE=/run/secrets/cal-diy-provider
```

Do not set the Bearer file with either client setting. The client id and secret file are a required pair. A partial pair, an empty client id, or both modes causes startup to fail.

The account id partitions connector state. It must contain 1 to 128 ASCII letters, digits, dots, underscores, or hyphens. Treat it as immutable ownership metadata for the instance; do not derive it from action input.

The provider URL must:

- use HTTPS, except for an explicit loopback development origin;
- contain an authority and no embedded username or password;
- identify the fixed Cal.diy deployment rather than an action-selected host.

The client removes a configured query and fragment, refuses non-loopback plain HTTP, does not follow redirects with credentials, and applies a 30-second request timeout. These checks do not replace deployment DNS, egress, or TLS policy.

4. Bind the host to one tenant and external account

Configure the fleet identity separately from the provider secret:

```
TEXT
AIP_CONNECTOR_HOST_TENANT_ID=tenant-calendar-production
AIP_CONNECTOR_HOST_MEMBERSHIP_ID=membership-cal-primary
AIP_CONNECTOR_HOST_EXTERNAL_ACCOUNT_ID=calendar_primary
AIP_CONNECTOR_HOST_EXTERNAL_ACCOUNT_SYSTEM=cal_diy
```

The external account id and system must be present together or both absent. For a Cal.diy production instance, set both and make the external account id match `AIPD_CAL_DIY_ACCOUNT_ID` by deployment policy.

The reviewed code validates each field independently; it does not promise an automatic equality check between those two account-id settings. Verify the equality in the admission material and deployment configuration before enabling the tenant binding.

The registry binding must associate only the intended tenant and admitted Cal.diy capabilities with this instance. The host's verified membership then reasserts the same tenant on every invocation. An action cannot redirect the request to another instance merely by naming another tenant or account.

5. Configure credential identity and revision metadata

The generic host can attach a non-secret credential handle to its verified execution context. Configure its four required parts together:

```
TEXT
AIP_CONNECTOR_HOST_CREDENTIAL_ID=cal-primary-provider
AIP_CONNECTOR_HOST_CREDENTIAL_ISSUER=cal_diy_deployment
AIP_CONNECTOR_HOST_CREDENTIAL_SCOPES=cal_diy:profile.get
AIP_CONNECTOR_HOST_CREDENTIAL_REVISION=cal-primary-2026-07-26-01
AIP_CONNECTOR_HOST_CREDENTIAL_POLICY_FILE=/run/config/cal-credential-policy.json
AIP_CONNECTOR_HOST_SECRET_PROVIDER_REF=secret://production/cal-diy/primary
```

Use the exact operation scopes granted to the admitted instance. Add scopes as a comma-delimited list; do not copy a wildcard from a qualification fixture into production.

The handle id, issuer, scopes, revision, and secret-provider reference are identity and revocation metadata. They are not the raw provider secret and do not cause the reviewed standalone Cal.diy binary to resolve a different credential for each action. The actual client credential still comes from the single Cal.diy secret file read at host startup.

The reloadable credential-revision policy can deny a revoked revision before a provider side effect. A missing, unreadable, invalid, or non-authorizing policy fails closed. Use it with registry and deployment rotation; do not mistake it for hot reload of the provider secret itself.

6. Verify routing before enabling mutations

Keep the new tenant capability binding disabled during admission and host startup. Verify:

1. the host registers with the expected connector type, version, instance,

replica, artifact digest, tenant, external account, and credential revision;

2. the host readiness probe succeeds against the configured profile endpoint;
3. discovery for the intended tenant contains the expected Cal.diy read

capability and no binding for another tenant;

4. `cap:cal_diy:profile.get` completes through the normal signed gateway path;
5. the provider request is associated with the expected Cal.diy account;

6. an unauthorized tenant has no eligible route rather than falling back to

this instance.

Use the Cal.diy quickstart for the deterministic read flow. Follow the production fleet deployment guide for the disabled-binding, admission, enablement, and rollback sequence. Do not start verification with a booking mutation.

7. Rotate a provider credential

The Cal.diy secret is read once when a host process starts. Replacing the file under a running process is not documented as a credential reload.

Rotate through a bounded replica rollout:

1. create a new provider credential with the required Cal.diy permissions;
2. mount it at a new owner-controlled secret path without changing the old

replica;

3. assign a new credential revision and configuration revision;
4. update the revision policy and admission or route material through its

reviewed control path;

5. start a replacement replica with the new secret file and verify a read;
6. route new assignments to the replacement and drain the old replica;
7. revoke the old provider credential only after in-flight work and rollback

requirements are resolved;

8. retain the identities, timestamps, verification, and revocation evidence.

Do not rotate by editing an action, reusing a replica identity for two active processes, or changing the provider account behind an unchanged instance.

Library-level tenant credential routing

The Cal.diy connector library contains an optional `with_tenant_credential_routing` composition. It maps a verified runtime tenant to an account and resolves a tenant-bound opaque Bearer handle with the exact `cal_diy:` scope. Missing membership, wrong issuer, wrong tenant, expired handle, missing scope, or resolution failure is denied.

The reviewed `aip-host-cal-diy` binary does not enable that library mode. Its public model is one account and one provider secret per admitted instance. Do not document the library feature as a standalone-host setting, and do not deploy the migration-only bundled daemon as the target fleet architecture.

Resolve authentication and routing failures

Symptom	Decision
Host requires one Bearer file or one complete client pair	Remove the partial or conflicting auth mode; never weaken the startup check
Secret file is rejected	Check regular-file type, ownership, mode, UTF-8 content, and 16 KiB bound without printing the secret
Provider URL is rejected	Use HTTPS with an authority and no embedded credentials; plain HTTP is loopback-only
Account id is rejected	Use the documented 1-to-128-character safe identifier and keep it stable
External account configuration is rejected	Set id and system together, then compare the id with the Cal.diy account setting
Credential handle configuration is rejected	Provide id, issuer, non-empty scopes, and revision together
Host registers but tenant discovery is empty	Compare the admitted binding, tenant, instance, capability, and route policy before touching the provider secret
Provider returns `401` or `403`	Verify credential type, account, expiry, and permissions; do not retry a mutation with a different secret
Provider returns a redirect	Correct the fixed base URL or upstream configuration; the client deliberately does not follow it with credentials

Related documentation

- Cal.diy connector overview
- Cal.diy quickstart
- Rotate connector credentials
- Connector registry and routing
- Security model