Cal.diy profile capabilities

Use these capabilities to read or update the Cal.diy profile owned by the credential configured for the selected external account. Both operations use the provider route /v2/me, but they have different authorization, approval, idempotency,

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

The connector exposes exactly two profile operations at the reviewed source revision. It does not expose an arbitrary pass-through profile endpoint.

Operations

Capability	Provider request	Risk	Approval	Retry
`cap:cal_diy:profile.get`	`GET /v2/me`	Low	No	Safe
`cap:cal_diy:profile.update`	`PATCH /v2/me`	Medium	Required	Unsafe

Both requests set `cal-api-version: 2024-08-13`. The connector also sends the configured provider credential and trusted AIP request metadata. Applications invoke the AIP capability; they do not construct this provider request directly.

Read the profile

`cap:cal_diy:profile.get` reads the profile associated with the selected credential. Its input is an object with no properties:

```JSON
{}
```

Additional properties are rejected by the published input schema. The operation is a synchronous, low-risk read with these contract markers:

Contract field	Value
Side effects	`read`, `external_network`
Human approval	Not required by the capability
Idempotency	Optional; action scope; revalidate input hash on collision
Connector retry support	Yes
Retry safety	Safe
Data sensitivity	Restricted; contains PII; redaction required
Credential scope	`cal_diy:profile.get`
Expected latency hint	2,000 ms
Provider timeout	30,000 ms

“No human approval” is not “no authorization.” The gateway and connector host still enforce the authenticated actor, tenant, admitted route, credential revision, and policy for the request.

Update the profile

`cap:cal_diy:profile.update` changes fields on the same authenticated profile. Pass only the fields that should change:

```
JSON
{
  "name": "Avery Patel",
  "timeFormat": 24,
  "weekStart": "Monday",
  "timeZone": "Europe/London",
  "locale": "en",
  "bio": "Scheduling product consultations"
}
```

The input schema technically accepts an empty object because every field is optional. Do not send an empty update as a probe. Use `profile.get` for a read, and include at least one intentional field in an update.

Update fields

Field	Type and constraint	Meaning
`email`	String with email format	Profile email
`name`	String	Display name
`timeFormat`	Integer: `12` or `24`	Clock format
`defaultScheduleId`	Integer	Default Cal.diy schedule identifier
`weekStart`	Weekday enum	First day of the displayed week
`timeZone`	Non-empty string	Profile time-zone identifier
`locale`	Supported locale enum	Profile interface locale
`avatarUrl`	String with URI format	Profile avatar URL
`bio`	String	Profile biography
`metadata`	Object with bounded scalar values	Provider profile metadata

Additional top-level properties are rejected. The schema does not add business rules beyond the constraints shown here; provider-side account policy can still reject a schema-valid value.

`weekStart` accepts exactly:

```
TEXT
Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday
```

`locale` accepts exactly:

```
TEXT
ar | ca | de | es | eu | he | id | ja | lv | pl | ro | sr | th | vi | az
cs | el | es-419 | fi | hr | it | km | nl | pt | ru | sv | tr | zh-CN
bg | da | en | et | fr | hu | iw | ko | no | pt-BR | sk | ta | uk | zh-TW
bn
```

The `metadata` object accepts at most 50 properties. Each property name has at most 40 characters. Each value must be a string, Boolean, or number; a string value has at most 500 characters. Nested objects, arrays, and null values are not accepted.

Update safety contract

Contract field	Value
Side effects	`write`, `external_network`, `identity`
Human approval	Required
Approval reason class	`customer_visible_change`
Idempotency	Required; external-account scope
Collision behavior	Revalidate input hash
Connector retry support	No
Retry safety	Unsafe
Data sensitivity	Restricted; contains PII; redaction required
Credential scope	`cal_diy:profile.update`
Expected latency hint	5,000 ms
Provider timeout	30,000 ms

The approval policy uses the tenant-policy selector and a 900,000 ms validity window. Approval evidence must include a reason, an input snapshot, and the policy decision. Bind the approval to the exact actor, tenant, capability, and input being executed. A changed field or value requires a new decision.

The identity side-effect marker is additive: it tells policy that this write changes account identity or presentation data. It does not replace provider authorization or grant permission to change an email address.

Idempotency and settlement

Every profile update requires an idempotency key. Scope it to the selected external account and preserve the same key only for the same canonical input. Reusing a key with different update content is an idempotency collision.

Before provider dispatch, the connector records a durable claim. A repeated matching request can observe one of three relevant states:

Durable state	Meaning	Application action
Completed	The original result was stored	Accept the stored result
In progress	Another invocation owns the claim	Read durable action state; do not create a new key
Uncertain	Dispatch may have crossed the provider boundary	Reconcile evidence; do not repeat the update

`profile.update` supports `dry_run`, `plan`, `commit`, and `reconcile` modes. A plan is not required before commit, and dry-run fidelity is limited to policy and schema checks. A successful dry run does not prove that Cal.diy will accept the update.

The capability declares no rollback or compensating capability. If a completed update must be reversed, submit a new, separately approved update using values confirmed from authoritative state. Do not treat that later write as an atomic rollback.

Result contract

The connector returns the validated Cal.diy JSON response as the AIP action output. The common output schema requires a non-empty string `status` and may contain `data`, `pagination`, or additional provider fields:

```
JSON
{
  "status": "success",
  "data": {}
}
```

This example demonstrates only the connector envelope. The connector does not publish an exhaustive provider profile schema for `data`. Consumers must tolerate additional fields and must not assume a particular `data` shape.

Before returning output, the connector recursively replaces values whose normalized field names are `secret`, `apiKey`, `hashedKey`, `token`, `accessToken`, `refreshToken`, or `clientSecret` with `[REDACTED]`. Profile data remains classified as restricted even after this targeted redaction.

An empty successful provider body is normalized to:

```
JSON
{
  "status": "success"
}
```

Failures and recovery

Failure	Typical cause	Safe response
`connector.cal_diy.invalid_action`	Input does not match the published operation schema	Correct the input; do not resend unchanged data
`connector.cal_diy.policy`	A mutation lacks its required idempotency key	Supply the missing key while preserving the original governed input
`connector.cal_diy.idempotency_collision`	The key already owns different input	Stop and inspect the original action; never force reuse
`connector.cal_diy.in_flight`	A matching mutation is still executing	Read durable state and wait for settlement
`connector.cal_diy.outcome_unknown`	The provider outcome cannot be proved	Reconcile using durable and provider evidence; do not retry
`connector.cal_diy.authentication`	Cal.diy returned `401` or `403`	Verify account binding and credential revision before another call
`connector.cal_diy.state_conflict`	Cal.diy returned `409` or `412`	Refresh authoritative profile state and review the intended change
`connector.cal_diy.rate_limited`	Cal.diy returned `429`	Honor the retry delay only when the action contract permits retry
`connector.cal_diy.remote_temporary`	Cal.diy returned a server error	Retry reads within policy; reconcile mutations
`connector.cal_diy.remote_rejected`	Cal.diy rejected another client request	Correct the request or account state before retrying
`connector.cal_diy.transport`	The provider exchange failed	Retry a read within policy; treat a dispatched mutation as uncertain
`connector.cal_diy.response_too_large`	The response exceeded the configured bound	Do not bypass the bound casually; inspect provider output and configuration
`connector.cal_diy.invalid_provider_output`	A successful response lacked a valid non-empty `status`	Preserve request evidence; reconcile an update before investigating provider drift

Errors can also include a validated provider request ID and remote status. Keep those values with the AIP action ID and external-account identity during diagnosis. Error details do not contain raw provider response bodies.

For global lifecycle and gateway errors, use the error reference.

Verify an integration

For a profile read, verify that:

- discovery returns `cap:cal_diy:profile.get` for the intended account route;
- the completed result belongs to the submitted action ID;
- the output has a non-empty `status`;
- returned PII is handled under the tenant's restricted-data policy.

For a profile update, additionally verify that:

- the approval evidence binds the exact update input;
- the external-account idempotency key is retained with the action record;
- the durable result is completed before relying on the new profile value;
- a follow-up read confirms the intended fields without exposing unrelated PII;
- an uncertain result is reconciled instead of repeated.

These checks validate application handling. They do not establish live-provider qualification or production readiness for a particular deployment.

Related documentation

- Cal.diy capability index
- Authentication and tenant routing
- Approvals and policy
- Transactions and compensation
- Error reference