Cal.diy schedules

Use these capabilities to list or read schedules and to create, update, or delete weekly availability with dated overrides. This page owns all six schedule operations.

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

Schedule operations use provider API version `2024-06-11`, rather than the general Cal.diy version used by most other capability families.

Choose an operation

Capability	Provider request	Risk	Approval
`cap:cal_diy:schedule.list`	`GET /v2/schedules`	Low	No
`cap:cal_diy:schedule.default.get`	`GET /v2/schedules/default`	Low	No
`cap:cal_diy:schedule.get`	`GET /v2/schedules/{schedule_id}`	Low	No
`cap:cal_diy:schedule.create`	`POST /v2/schedules`	Medium	Required
`cap:cal_diy:schedule.update`	`PATCH /v2/schedules/{schedule_id}`	Medium	Required
`cap:cal_diy:schedule.delete`	`DELETE /v2/schedules/{schedule_id}`	High	Required

List or read schedules

List and default get both accept an empty object:

```
JSON
{}
```

Read one owned schedule with an integer provider id:

```
JSON
{
  "schedule_id": 41
}
```

The public get schema does not set a minimum for `schedule_id`; Cal.diy can reject an integer outside the provider's domain. Additional fields are rejected.

All three reads are low risk and approval-free at the capability level. The connector does not publish the provider response field shape.

Create a schedule

Create requires a name, time zone, and explicit default choice:

```
JSON
{
  "name": "Customer calls",
  "timeZone": "Europe/London",
  "isDefault": false,
  "availability": [
    {
      "days": ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"],
      "startTime": "09:00",
      "endTime": "17:00"
    }
  ],
  "overrides": [
    {
      "date": "2026-08-10",
      "startTime": "11:00",
      "endTime": "15:00"
    }
  ]
}
```

Field	Constraint
`name`	Required non-empty string
`timeZone`	Required non-empty string
`isDefault`	Required boolean
`availability`	Optional array of weekly intervals
`overrides`	Optional array of dated intervals

Both arrays may be empty. Additional fields are rejected at the schedule and interval object boundaries.

Weekly availability

Each `availability` entry requires:

Field	Constraint
`days`	Non-empty array with unique weekday values
`startTime`	`HH:MM` in the range `00:00` through `23:59`
`endTime`	The same `HH:MM` format

The exact weekday enum is:

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

Weekday values are case-sensitive.

Dated overrides

Each `overrides` entry requires an ISO date `date`, plus `startTime` and `endTime` in the same 24-hour format. One object represents one dated interval.

The connector schema does not verify that a start precedes its end, that intervals do not overlap, that override dates are unique, or that a named time zone makes a local time valid. Validate those rules before approval and rely on Cal.diy for provider-owned schedule semantics.

Create is medium risk. It names `schedule.delete` as best-effort compensation. Compensation requires separate approval and is not automatic.

Update a schedule

Update requires only `schedule_id`; every schedule property is optional:

```
JSON
{
  "schedule_id": 41,
  "name": "Customer calls and demos",
  "isDefault": true
}
```

The optional fields use the same schemas as create: `name`, `timeZone`, `isDefault`, `availability`, and `overrides`. The connector sends supplied fields as a provider patch but does not define provider merge or replacement semantics for the two arrays.

Unlike get and delete, update accepts `schedule_id` as either a string or an integer. Its string branch has no minimum length. An identifier-only update or an empty string id can pass the published schema without expressing a useful provider change; use an id obtained from a trusted schedule read and include an intentional patch field.

Before update, the connector reads the schedule selected by the same id. A successful preflight does not lock the schedule or validate the proposed availability.

Update is medium risk and has no compensating capability.

Delete a schedule

Delete uses the same integer-only input as public get:

```
JSON
{
  "schedule_id": 41
}
```

The connector reads the selected schedule before deletion. Delete is high risk and destructive, and it has no rollback contract. Creating another schedule later is a separate provider mutation; the connector declares no restoration of the original id, default assignment, or availability.

Shared read contract

The three reads are synchronous, retry-safe, and eligible for connector retry. Their idempotency key is optional, action-scoped, and collision-checked against the input hash. No capability-level human approval is required.

Shared mutation contract

All three mutations require:

Contract field	Value
Human approval	Required
Approval selector	Tenant policy
Approval validity window	900,000 ms
Approval evidence	Reason, input snapshot, policy decision
Idempotency	Required; external-account scope
Collision behavior	Revalidate input hash
Connector retry	Not supported
Retry safety	Unsafe
Execution	Synchronous only
Transaction modes	`dry_run`, `plan`, `commit`, `reconcile`
Plan before commit	Not required
Dry-run fidelity	Policy and schema only

Create and update are medium risk. Delete is high risk, destructive, and uses the destructive approval reason. None carries `send_message`.

Only create declares best-effort delete compensation. Update and delete declare rollback as not supported.

Data and result contract

The family is classified as confidential. Its contract does not declare PII and does not require redaction. Deployment policy may apply stricter handling to provider output or tenant-specific schedule names.

The connector returns the validated provider JSON body. A non-empty string `status` is required. `data`, `pagination`, and additional provider fields are optional and provider-owned.

The common response scrubber still replaces secret-, key-, and token-named values with `[REDACTED]`. It is not a general confidential-data filter.

Failures and recovery

Failure	Relevant cause	Safe response
`connector.cal_diy.invalid_action`	Invalid id, weekday, time, date, or unknown field	Correct the exact input before another call
`connector.cal_diy.policy`	Mutation omitted its required idempotency key	Preserve approved input and add the missing key
`connector.cal_diy.idempotency_collision`	One key owns different mutation input	Stop and inspect the original action
`connector.cal_diy.in_flight`	A matching schedule mutation is executing	Read durable state and wait
`connector.cal_diy.outcome_unknown`	Prior provider dispatch cannot be proved	Reconcile; do not create a replacement key
`connector.cal_diy.authentication`	Provider returned `401` or `403`	Verify account binding, credentials, and schedule ownership
`connector.cal_diy.state_conflict`	Provider returned `409` or `412`	Refresh the schedule and review the change
`connector.cal_diy.rate_limited`	Provider returned `429`	Retry reads within policy; reconcile mutations
`connector.cal_diy.remote_temporary`	Provider returned a server error	Retry reads within policy; reconcile mutations
`connector.cal_diy.transport`	Provider exchange failed	Retry reads only; treat dispatched mutations as uncertain
`connector.cal_diy.invalid_provider_output`	Successful response lacked valid `status`	Preserve evidence and reconcile a mutation

Permanent rejection, invalid credentials, oversized responses, and durable settlement failures use the common Cal.diy error family. Gateway and lifecycle errors are described in the global error reference.

Verify schedule handling

For a read, retain the capability, schedule selector, actor, tenant, external account, response status, and provider request id when present.

For a mutation, also retain approval evidence and the original idempotency key. Wait for a completed durable result, then read the exact schedule or list schedules and compare only the intended fields. For uncertain dispatch, reconcile the original operation before another mutation.

These checks validate application handling. They do not prove bookability, time-zone correctness, non-overlap, event-type propagation, live-provider qualification, or production readiness.

Related documentation

- Cal.diy capability index
- Event types, private links, and webhooks

- Calendars and free/busy
- Approvals and policy
- Transactions and compensation