Cal.diy event types, private links, and webhooks

Use this capability family to define bookable event types, issue restricted booking links, and attach webhooks to one event type. The reviewed connector exposes 15 fixed operations. Choose an AIP capability from this page instead of constru

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/cal-diy/capabilities/event-types-and-private-links.md

Event-type CRUD and event-type webhooks use provider API version `2024-06-14`. Private links use `2024-04-15`. The webhook payload version `2021-10-20` is a separate value and does not replace either provider API version.

Choose an operation

Event types

Capability	Provider request	Required input	Risk
`cap:cal_diy:event_type.list`	`GET /v2/event-types`	None	Low
`cap:cal_diy:event_type.get`	`GET /v2/event-types/{event_type_id}`	`event_type_id`	Low
`cap:cal_diy:event_type.create`	`POST /v2/event-types`	`title`, `slug`, `lengthInMinutes`	Medium
`cap:cal_diy:event_type.update`	`PATCH /v2/event-types/{event_type_id}`	`event_type_id`	Medium
`cap:cal_diy:event_type.delete`	`DELETE /v2/event-types/{event_type_id}`	`event_type_id`	High

Private links

Capability	Provider request	Required input	Risk
`cap:cal_diy:event_type.private_link.list`	`GET /v2/event-types/{event_type_id}/private-links`	`event_type_id`	Low
`cap:cal_diy:event_type.private_link.create`	`POST /v2/event-types/{event_type_id}/private-links`	`event_type_id`	Medium
`cap:cal_diy:event_type.private_link.update`	`PATCH /v2/event-types/{event_type_id}/private-links/{link_id}`	`event_type_id`, `link_id`	Medium
`cap:cal_diy:event_type.private_link.delete`	`DELETE /v2/event-types/{event_type_id}/private-links/{link_id}`	`event_type_id`, `link_id`	High

A private link is access-controlled data even when its provider URL can be shared.

Event-type webhooks

Capability	Provider request	Required input	Risk
`cap:cal_diy:event_type.webhook.list`	`GET /v2/event-types/{event_type_id}/webhooks`	`event_type_id`	Low
`cap:cal_diy:event_type.webhook.get`	`GET /v2/event-types/{event_type_id}/webhooks/{webhook_id}`	Both ids	Low
`cap:cal_diy:event_type.webhook.create`	`POST /v2/event-types/{event_type_id}/webhooks`	Event id, active flag, URL, triggers, secret ref	Medium
`cap:cal_diy:event_type.webhook.update`	`PATCH /v2/event-types/{event_type_id}/webhooks/{webhook_id}`	Both ids	Medium
`cap:cal_diy:event_type.webhook.delete`	`DELETE /v2/event-types/{event_type_id}/webhooks/{webhook_id}`	Both ids	High
`cap:cal_diy:event_type.webhook.delete_all`	`DELETE /v2/event-types/{event_type_id}/webhooks`	`event_type_id`	High

`delete_all` removes every webhook scoped to the selected event type; it is not a filtered delete.

List and read event types

`event_type.list` accepts an object with any subset of these query fields:

Field	Constraint
`username`	Non-empty string
`eventSlug`	Non-empty string
`usernames`	Non-empty array of strings
`orgSlug`	Non-empty string
`orgId`	Integer
`sortCreatedAt`	`asc` or `desc`

Additional fields are rejected. With no filters, pass an empty object:

```
JSON
{}
```

A filtered request can be as small as:

```
JSON
{
  "username": "avery",
  "sortCreatedAt": "desc"
}
```

`event_type.get` accepts only an integer provider identifier:

```
JSON
{
  "event_type_id": 42
}
```

The connector percent-encodes path segments and removes path and query fields from any provider JSON body. Applications pass one AIP input object; they do not partition the provider request themselves.

Create or update an event type

Create requires three fields:

```
JSON
{
  "title": "Product consultation",
  "slug": "product-consultation",
  "lengthInMinutes": 30
}
```

Update requires only the event-type id. Include the fields that should change:

```
JSON
{
  "event_type_id": 42,
  "minimumBookingNotice": 120,
  "hidden": false
}
```

The connector sends `event_type_id` in the provider path and omits it from the JSON body. An otherwise empty update is schema-valid but is not a useful read or health probe. Use `event_type.get` to inspect current state.

Additional top-level properties are rejected for both create and update.

Core and timing fields

Field	Constraint
`title`	Non-empty string; required on create
`slug`	Non-empty string; required on create
`lengthInMinutes`	Integer at least `1`; required on create
`lengthInMinutesOptions`	Non-empty, unique array of integers at least `1`
`description`	String
`slotInterval`	Integer
`minimumBookingNotice`	Integer at least `0`
`beforeEventBuffer`	Integer
`afterEventBuffer`	Integer
`offsetStart`	Integer at least `0`
`scheduleId`	Integer

The schema intentionally does not add a non-negative minimum to `slotInterval`, `beforeEventBuffer`, or `afterEventBuffer`. Do not infer an application range that the published contract does not declare; Cal.diy may still apply provider-side business rules.

Limits, recurrence, and confirmation

Field	Accepted shape
`bookingLimitsCount`	Disabled, or at least one of `day`, `week`, `month`, `year` with integer value at least `1`
`bookingLimitsDuration`	Disabled, or the same period keys with integer value at least `15`
`bookerActiveBookingsLimit`	Disabled, or `maximumActiveBookings` at least `1`, `offerReschedule`, or both
`bookingWindow`	Disabled; rolling `businessDays` or `calendarDays`; or an explicit `range`
`recurrence`	Disabled, or positive `interval` and `occurrences` with `yearly`, `monthly`, or `weekly` frequency
`confirmationPolicy`	Disabled, `always`, or time-threshold policy
`seats`	Disabled, or an active seat policy

Every disabled form is exactly:

```
JSON
{
```

```
  "disabled": true
}
```

An active period-limit form uses at least one period key and may set `disabled` to `false`:

```
JSON
{
  "week": 5,
  "disabled": false
}
```

A rolling booking window requires `type` and a non-negative numeric `value`. An explicit range requires a non-empty `value` array whose items are ISO dates or date-times.

An active confirmation policy always requires `type` and `blockUnconfirmedBookingsInBooker`. Type `time` also requires `noticeThreshold`, with unit `minutes` or `hours` and a positive integer `count`. Type `always` permits that threshold but does not require it.

An active seat policy requires `seatsPerTimeSlot` from `1` through `1000`, `showAttendeeInfo`, and `showAvailabilityCount`.

The schema rejects an active recurrence together with an active `bookerActiveBookingsLimit`. Disable one of those features before sending the other.

Booking page and presentation fields

Field	Constraint
`bookingFields`	Non-empty array of supported field variants
`disableGuests`	Boolean
`onlyShowFirstAvailableSlot`	Boolean
`bookerLayouts`	Default layout plus a non-empty unique enabled-layout list
`requiresBookerEmailVerification`	Boolean
`lockTimeZoneToggleOnBookingPage`	Boolean
`color`	Six-digit light and dark hex colors
`customName`	String
`successRedirectUrl`	URI string
`hidden`	Boolean
`bookingRequiresAuthentication`	Boolean
`interfaceLanguage`	Supported interface-language enum
`showOptimizedSlots`	Boolean

Layout values are `month`, `week`, and `column`. The default layout must be one of those values; the schema does not separately require it to appear in the enabled list.

Built-in booking-field variants use `type` values `name`, `splitName`, or `email`, or fixed `slug` values `title`, `location`, `notes`, `guests`, and `rescheduleReason`. Custom variants use these `type` values:

```
TEXT
phone | address | text | url | number | textarea | select | multiselect
multiemail | checkbox | radio | boolean
```

Every custom variant requires `type`, a non-empty `slug`, `label`, and `required`. `select`, `multiselect`, `checkbox`, and `radio` also require a non-empty `options` array. Each variant rejects fields outside its published shape.

The interface-language enum is:

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

The empty string is an explicit enum member; it is not a documentation placeholder.

Calendar, privacy, and media fields

Field	Constraint
`destinationCalendar`	Object with non-empty `integration` and `externalId`
`useDestinationCalendarEmail`	Boolean
`hideCalendarNotes`	Boolean
`hideCalendarEventDetails`	Boolean
`hideOrganizerEmail`	Boolean
`calVideoSettings`	Bounded object of recording, transcription, email, and exit-redirect settings
`disableCancelling`	Object with optional Boolean `disabled`
`disableRescheduling`	Object with optional `disabled` and `minutesBefore` at least `1`
`allowReschedulingPastBookings`	Boolean
`allowReschedulingCancelledBookings`	Boolean
`locations`	Non-empty array of supported location variants

`calVideoSettings.redirectUrlOnExit` accepts a URI string or null. Its other fields are Booleans for organizer or guest recording and transcription, automatic recording, transcription email, and guest or organizer controls. Additional fields are rejected.

Location variants are:

- `address` with non-empty `address` and Boolean `public`;
- `link` with URI `link` and Boolean `public`;
- `phone` with E.164-like `phone` and Boolean `public`;
- `integration` with one supported integration name;
- marker-only `attendeeAddress`, `attendeePhone`, or `attendeeDefined`.

The phone pattern is `+` followed by a nonzero digit and 6 through 14 further digits. Supported integration names are:

```
TEXT
cal-video | google-meet | zoom | whereby-video | whatsapp-video | webex-video
telegram-video | tandem | sylaps-video | skype-video | sirius-video
signal-video | shimmer-video | salesroom-video | roam-video | riverside-video
ping-video | office365-video | mirotalk-video | jitsi | jelly-video
jelly-conferencing | huddle | facetime-video | element-call-video
eightxeight-video | discord-video | demodesk-video | campfire-video
```

Delete an event type

`event_type.delete` accepts only:

```
JSON
{
  "event_type_id": 42
}
```

This is a high-risk destructive mutation. Before dispatch, the connector reads the same event type. The preflight verifies current provider reachability and object visibility; it does not make deletion reversible or retry-safe.

The delete capability has no rollback contract. A later create operation would produce new provider state and is not an atomic restoration of the deleted event type.

Manage private links

Private-link list requires only `event_type_id`. Create may also set an expiry, a usage limit, or both:

```
JSON
{
  "event_type_id": 42,
  "expiresAt": "2030-01-01T00:00:00Z",
  "maxUsageCount": 20
}
```

`expiresAt` must have date-time format. `maxUsageCount` is an integer at least `1`. Both are optional, so policy should decide whether an unbounded private link is acceptable before approval.

Update requires the parent id and a non-empty link id:

```
JSON
{
  "event_type_id": 42,
  "link_id": "link-7",
  "maxUsageCount": 10
}
```

Delete accepts the same two identifiers and no update fields. It is the high-risk revocation operation.

Create, update, and delete first read the parent event type. There is no individual private-link get capability in this catalogue. Use list to inspect the provider-owned collection, and keep returned links under restricted-data handling.

Private-link create names `event_type.private_link.delete` as a best-effort compensation. The compensation requires approval and has a 24-hour contract window. It is neither automatic nor guaranteed.

Manage event-type webhooks

List requires `event_type_id` and optionally accepts `skip` at least `0` and `take` from `1` through `250`. Get and delete require an additional non-empty `webhook_id`.

Create requires all of these fields:

```
JSON
{
  "event_type_id": 42,
  "active": true,
  "subscriberUrl": "https://hooks.example.com/cal/cal-hook-01",
  "triggers": ["BOOKING_CREATED"],
  "webhook_secret_ref": "cal-hook-01"
}
```

In this example, the deployment-owned destination prefix would be `https://hooks.example.com/cal/`. The final path segment must equal the secret reference exactly.

Update requires only `event_type_id` and `webhook_id`. It may change `active`, `subscriberUrl`, `triggers`, `webhook_secret_ref`, or the fixed `version`. Sending a new secret reference without the bound subscriber URL is rejected. Changing the subscriber URL without a valid secret reference is also rejected.

Create and update may include `version`, but its only valid value is `2021-10-20`. The connector inserts that value when it is omitted.

Trigger catalogue

`triggers` is a non-empty, unique array selected from these 19 values:

```
TEXT
BOOKING_CREATED | BOOKING_PAYMENT_INITIATED | BOOKING_PAID
BOOKING_RESCHEDULED | BOOKING_REQUESTED | BOOKING_CANCELLED
```

```
BOOKING_REJECTED | BOOKING_NO_SHOW_UPDATED | FORM_SUBMITTED
MEETING_ENDED | MEETING_STARTED | RECORDING_READY
RECORDING_TRANSCRIPTION_GENERATED | OOO_CREATED
AFTER_HOSTS_CAL_VIDEO_NO_SHOW | AFTER_GUESTS_CAL_VIDEO_NO_SHOW
FORM_SUBMITTED_NO_EVENT | DELEGATION_CREDENTIAL_ERROR
WRONG_ASSIGNMENT_REPORT
```

Secret and destination boundary

`webhook_secret_ref` is an opaque deployment selector, not secret material. It must contain 1 through 128 URL-segment-safe characters, begin with an ASCII letter or digit, and then use only letters, digits, `.`, `_`, `~`, or `-`.

The connector rejects raw `secret` input and custom `payloadTemplate` input. It resolves the opaque reference through the deployment secret provider, injects the raw secret only into the provider request, and removes the reference from that request body. Returned secret-like fields are recursively redacted.

Webhook destinations fail closed. A subscriber URL must:

- use HTTPS and contain a host;
- contain no username, password, query, or fragment;
- match a configured prefix by scheme, host, and effective port; and
- have a path equal to the configured prefix plus `webhook_secret_ref`.

The schema's generic URI format is therefore not the complete authorization rule. A schema-valid URL still fails when it is outside the deployment-owned prefix and selector binding.

Create preflights the parent event type. Update and delete preflight the exact webhook. `delete_all` preflights the parent event type, then removes the whole event-type webhook collection.

Webhook create names `event_type.webhook.delete` as a best-effort, approval-required compensation with a 24-hour contract window. It does not undo webhook deliveries that have already occurred.

Shared mutation contract

All ten mutations require human approval and an external-account-scoped idempotency key. Medium-risk creates and updates use the `customer_visible_change` reason class. The four deletes use `destructive_state_change`.

Approval uses the tenant-policy selector with a 900,000 ms validity window and requires a reason, input snapshot, and policy decision. Approval does not grant provider ownership, bypass the destination policy, or make a stale input valid.

Every mutation is synchronous, declares retry unsafe, and advertises no connector retry support. Preserve the same key only for the same capability, external account, and canonical input. If the provider outcome is uncertain, reconcile durable and provider evidence instead of sending the mutation again.

All mutations advertise `dry_run`, `plan`, `commit`, and `reconcile`. None of these 15 operations requires a plan before commit. Dry-run fidelity is limited to policy and schema checks; it does not prove that Cal.diy will accept a provider-side change.

Event-type, private-link, and webhook creates have named best-effort delete compensations. Updates and deletes declare rollback unsupported. A compensation is a new governed action, not an automatic transaction rollback.

Read, data, and output contract

The five reads are low risk, synchronous, retry-safe, and eligible for connector retry. Their idempotency key is optional and action-scoped.

Event-type CRUD output is confidential. Private-link and event-type-webhook output is restricted. Every operation in this family is marked as containing PII and requiring redaction. Apply the stronger tenant policy when returned provider fields contain URLs, booking questions, identity data, or webhook configuration.

The connector returns the provider JSON body as AIP output after validation and recursive secret-field redaction. The output requires a non-empty string `status`; `data`, `pagination`, and additional provider fields are optional. The connector does not publish a complete provider response-field schema.

Failures and recovery

Failure	Relevant cause	Safe response
`connector.cal_diy.invalid_action`	Schema mismatch, raw webhook secret, unsupported version, unbound URL, or destination-policy failure	Correct input or trusted deployment policy; do not weaken the boundary
`connector.cal_diy.policy`	Missing mutation idempotency key or unresolved webhook secret reference	Preserve governed input; correct key or deployment secret mapping
`connector.cal_diy.idempotency_collision`	One key owns different canonical input	Stop and inspect the original action
`connector.cal_diy.in_flight`	The matching mutation is still owned	Read durable state and wait
`connector.cal_diy.outcome_unknown`	Prior provider dispatch cannot be proved	Reconcile; never invent a new retry key
`connector.cal_diy.authentication`	Provider returned `401` or `403`	Verify account binding, credential revision, and provider ownership
`connector.cal_diy.state_conflict`	Provider returned `409` or `412`	Refresh the event type or webhook and review the intended 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 a dispatched mutation as uncertain
`connector.cal_diy.state`	Durable state or webhook-secret resolution failed	Restore the deployment-owned dependency before another mutation
`connector.cal_diy.invalid_provider_output`	Successful provider body lacked a valid `status`	Preserve request evidence and reconcile any mutation

The common connector error set also covers permanent provider rejection, oversized responses, invalid credentials, invalid requests, and durable pre-dispatch or post-dispatch settlement failures. Use the global error reference for gateway and lifecycle errors.

Verify an integration

For every operation, verify the discovered capability id, selected external account, completed AIP action id, and non-empty output `status`.

For event-type writes, also verify the final provider state with `event_type.get`. For private-link changes, list the parent collection and handle returned links as restricted data. For webhook changes, verify the event-type webhook record without exposing the secret, then separately test signed ingress under the deployment's webhook-security procedure.

For every mutation, retain the approval, input snapshot, external-account idempotency key, provider request id when present, and durable settlement state. A follow-up read is confirmation evidence; it does not convert an uncertain mutation into a safe retry.

These checks validate integration behavior. They do not establish live Cal.diy qualification or production readiness for a specific deployment.

Related documentation

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