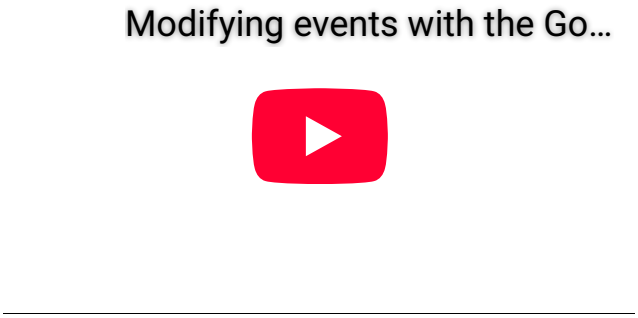
Recurring events

This document describes how to work with recurring events (/workspace/calendar/concepts/events-calendars#recurring_events) and their instances.



Create recurring events

Creating recurring events is similar to creating (/workspace/calendar/v3/reference/events/insert) a regular (single) event with the `event` (/workspace/calendar/v3/reference/events) resource's `recurrence` (/workspace/calendar/v3/reference/events#recurrence) field set.

Protocol (#protocol) Java (#java) .NET (#.net) Python (#python) PHP (#php) Ruby (#ruby)

```
POST /calendar/v3/calendars/primary/events
...

{
  "summary": "Appointment",
  "location": "Somewhere",
  "start": {
    "dateTime": "2011-06-03T10:00:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  },
  "end": {
    "dateTime": "2011-06-03T10:25:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  },
  "recurrence": [
    "RRULE:FREQ=WEEKLY;UNTIL=20110701T170000Z",
  ],
  "attendees": [
```

```
    {
      "email": "attendeeEmail",
      # Other attendee's data...
    },
    # ...
  ],
}
```

Access instances

To see all the instances (/workspace/calendar/concepts/events-calendars#instances_and_exceptions) of a given recurring event you can use the events.instances() (/workspace/calendar/v3/reference/events/instances) request.

The `events.list()` (/workspace/calendar/v3/reference/events/list) request by default only returns single events, recurring events, and exceptions (/workspace/calendar/concepts/events-calendars#instances_and_exceptions); instances that are not exceptions are not returned. If the `singleEvents` (/workspace/calendar/v3/reference/events/list#singleEvents) parameter is set `true` then all individual instances appear in the result, but underlying recurring events don't. When a user who has free/busy permissions queries `events.list()`, it behaves as if `singleEvent` is `true`. For more information about access control list rules, see Acl (/calendar/v3/reference/acl).

Warning: Do not modify instances individually when you want to modify the entire recurring event, or "this and following" (#modifying_all_following_instances) instances. This creates lots of exceptions that clutter the calendar, slowing down access and sending a high number of change notifications to users.

Individual instances are similar to single events. Unlike their parent recurring events, instances do not have the `recurrence` (/workspace/calendar/v3/reference/events#recurrence) field set.

The following event fields are specific to instances:

- `recurringEventId` (/workspace/calendar/v3/reference/events#recurringEventId) — the ID of the parent recurring event this instance belongs to
- `originalStartTime` (/workspace/calendar/v3/reference/events#originalStartTime) — the time this instance starts according to the recurrence data in the parent recurring event. This

can be different from the actual `start` (/workspace/calendar/v3/reference/events#start) time if the instance was rescheduled. It uniquely identifies the instance within the recurring event series even if the instance was moved.

Modify or delete instances

To modify a single instance (creating an exception), client applications must first retrieve the instance and then update it by sending an authorized PUT request to the instance edit URL with updated data in the body. The URL is of the form:

```
https://www.googleapis.com/calendar/v3/calendars/calendarId/events/instanceId
```

Use appropriate values in place of ***calendarId*** and ***instanceId***.

Note: The special ***calendarId*** value `primary` can be used to refer to the authenticated user's primary calendar.

Upon success, the server responds with an HTTP 200 OK status code with the updated instance. The following example shows how to cancel an instance of a recurring event.

Protocol (#protocol) Java (#java) .NET (#.net) Python (#python) PHP (#php) Ruby (#ruby)

```
PUT /calendar/v3/calendars/primary/events/instanceId
...

{
  "kind": "calendar#event",
  "id": "instanceId",
  "etag": "instanceEtag",
  "status": "cancelled",
  "htmlLink": "https://www.google.com/calendar/event?eid=instanceEid",
  "created": "2011-05-23T22:27:01.000Z",
  "updated": "2011-05-23T22:27:01.000Z",
  "summary": "Recurring event",
  "location": "Somewhere",
```

```
  "creator": {
    "email": "userEmail"
  },
  "recurringEventId": "recurringEventId",
  "originalStartTime": "2011-06-03T10:00:00.000-07:00",
  "organizer": {
    "email": "userEmail",
    "displayName": "userDisplayName"
  },
  "start": {
    "dateTime": "2011-06-03T10:00:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  },
  "end": {
    "dateTime": "2011-06-03T10:25:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  },
  "iCalUID": "eventUID",
  "sequence": 0,
  "attendees": [
    {
      "email": "attendeeEmail",
      "displayName": "attendeeDisplayName",
      "responseStatus": "needsAction"
    },
    # ...
    {
      "email": "userEmail",
      "displayName": "userDisplayName",
      "responseStatus": "accepted",
      "organizer": true,
      "self": true
    }
  ],
  "guestsCanInviteOthers": false,
  "guestsCanSeeOtherGuests": false,
  "reminders": {
    "useDefault": true
  }
}
```

Modify all following instances

In order to change all the instances of a recurring event on or after a given (target) instance, you must make two separate API requests. These requests split the original recurring event into two: the original one which retains the instances without the change and the new recurring event having instances where the change is applied:

1. Call **`events.update()`** (/workspace/calendar/v3/reference/events/update) to trim the original recurring event of the instances to be updated. Do this by setting the **`UNTIL`** component of the **`RRULE`** to point before the start time of the first target instance. Alternatively, you can set the**`COUNT`** component instead of **`UNTIL`**.
2. Call **`events.insert()`** (/workspace/calendar/v3/reference/events/insert) to create a new recurring event with all the same data as the original, except for the change you are attempting to make. The new recurring event must have the start time of the target instance.

This example shows how to change the location to "Somewhere else", starting from the third instance of the recurring event from the previous examples.

Protocol (#protocol)

```
# Updating the original recurring event to trim the instance list:

PUT /calendar/v3/calendars/primary/events/recurringEventId
...

{
  "summary": "Appointment",
  "location": "Somewhere",
  "start": {
    "dateTime": "2011-06-03T10:00:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  },
  "end": {
    "dateTime": "2011-06-03T10:25:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  },
  "recurrence": [
    "RRULE:FREQ=WEEKLY;UNTIL=20110617T065959Z",
  ],
  "attendees": [
    {
```

```
      "email": "attendeeEmail",
      # Other attendee's data...
    },
    # ...
  ],
}


# Creating a new recurring event with the change applied:

POST /calendar/v3/calendars/primary/events
...

{
  "summary": "Appointment",
  "location": "Somewhere else",
  "start": {
    "dateTime": "2011-06-17T10:00:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  },
  "end": {
    "dateTime": "2011-06-17T10:25:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  },
  "recurrence": [
    "RRULE:FREQ=WEEKLY;UNTIL=20110617T065959Z",
  ],
  "attendees": [
    {
      "email": "attendeeEmail",
      # Other attendee's data...
    },
    # ...
  ],
}
```

Note: Changing all following instances resets any exceptions happening after the target instance.

Last updated 2025-07-30 UTC.