

# Manage focus time, out of office, and working location events

This page explains how to use the Google Calendar API to create events that show the status of Google Calendar users. Status events describe where users are or what they're doing, including whether they're in focus time, out of office, or working from a certain location.

In Google Calendar, users can create focus time, out of office and working location events to indicate their custom status and location. These features are only available on primary calendars, and to some Google Calendar users.

For more details, go to [Use focus time in Google Calendar](https://support.google.com/calendar/answer/11190973)

(<https://support.google.com/calendar/answer/11190973>) and [Turn working location on or off for users](https://support.google.com/a/answer/10918567) (<https://support.google.com/a/answer/10918567>).

## Read and list Calendar status events

You can read and list Calendar status events in the [Events](#)

(</workspace/calendar/api/v3/reference/events>) resource of the Calendar API.

To read a status event, use the [events.get](#) (</workspace/calendar/api/v3/reference/events/get>) method, specifying the [eventId](#) (</workspace/calendar/api/v3/reference/events/get#eventId>) of the event.

To list status events, use the [events.list](#) (</workspace/calendar/api/v3/reference/events/list>) method, specifying one or more of the following values in the [eventTypes](#) (</workspace/calendar/api/v3/reference/events/list#eventTypes>) field:

- 'focusTime'
- 'outOfOffice'
- 'workingLocation'

Then, in the returned Event objects, inspect that the [eventType](#)

(</workspace/calendar/api/v3/reference/events#eventType>) field has the requested value, and refer to the corresponding field for details about the status created by the user in Google Calendar:

- **focusTimeProperties** (/workspace/calendar/api/v3/reference/events#focusTimeProperties)
- **outOfOfficeProperties** (/workspace/calendar/api/v3/reference/events#outOfOfficeProperties)
- **workingLocationProperties**  
(/workspace/calendar/api/v3/reference/events#workingLocationProperties)

## Subscribe to changes on status events

You can subscribe to changes on status events in the **Events**  
(/workspace/calendar/api/v3/reference/events) resource of the Calendar API.

Use the **events.watch** (/workspace/calendar/api/v3/reference/events/watch) method, specifying the **calendarId** (/workspace/calendar/api/v3/reference/events/watch#calendarId) of the Calendar to subscribe to and one or more of the following values in the **eventTypes**  
(/workspace/calendar/api/v3/reference/events/watch#eventTypes) field:

- 'focusTime'
- 'outOfOffice'
- 'workingLocation'

## Create and update Calendar status events

To create a status event, you create an instance of the **Events**  
(/workspace/calendar/api/v3/reference/events) resource using the **events.insert**  
(/workspace/calendar/api/v3/reference/events/insert) method, setting the required fields for the event type.

If you update the status event using **events.update**  
(/workspace/calendar/api/v3/reference/events/update) method, the event must maintain the required fields.

### Create focus time

To create a focus time event:

- Set eventType (/workspace/calendar/api/v3/reference/events#eventType) to 'focusTime'.
- Include the focusTimeProperties (/workspace/calendar/api/v3/reference/events#focusTimeProperties) field.
- Set the transparency (/workspace/calendar/api/v3/reference/events#transparency) field to 'opaque'.
- Set the event's start (/workspace/calendar/api/v3/reference/events#start) and end (/workspace/calendar/api/v3/reference/events#end) fields to be a timed event (with start and end times specified).  
Focus times cannot be all-day events.

For feature details, go to Use focus time in Google Calendar  
(<https://support.google.com/calendar/answer/11190973>)

## Create out of office

To create an out of office event:

- Set eventType (/workspace/calendar/api/v3/reference/events#eventType) to 'outOfOffice'.
- Include the outOfOfficeProperties (/workspace/calendar/api/v3/reference/events#outOfOfficeProperties) field.
- Set the transparency (/workspace/calendar/api/v3/reference/events#transparency) field to 'opaque'.
- Set the event's start (/workspace/calendar/api/v3/reference/events#start) and end (/workspace/calendar/api/v3/reference/events#end) fields to be a timed event (with start and end times specified).  
Out of office events cannot be all-day events.

For feature details, go to Show when you're out of office  
(<https://support.google.com/calendar/answer/7638168>)

## Create working location

To create a working location event:

- Set eventType

(/workspace/calendar/api/v3/reference/events#eventType) to 'workingLocation'.

- Include the **workingLocationProperties** (/workspace/calendar/api/v3/reference/events#workingLocationProperties) field.
- Set the **visibility** (/workspace/calendar/api/v3/reference/events#visibility) field to 'public'.
- Set the **transparency** (/workspace/calendar/api/v3/reference/events#transparency) field to 'transparent'.
- Set the event's **start** (/workspace/calendar/api/v3/reference/events#start) and **end** (/workspace/calendar/api/v3/reference/events#end) fields to be either:
  - A timed event (with start and end times specified);
  - An all-day event (with start and end dates specified) which spans exactly one day.

All-day working location events cannot span multiple days, but timed events can.

The following fields are optional but recommended for the best user experience when inserting an **officeLocation**

(/workspace/calendar/api/v3/reference/events#workingLocationProperties.officeLocation):

- **workingLocationProperties.officeLocation.buildingId**  
(/workspace/calendar/api/v3/reference/events#workingLocationProperties.officeLocation.buildingId)  
: This should match a **buildingId** in the organization's **resources** (/workspace/admin/directory/reference/rest/v1/resources.buildings#resource:-building) database. This helps users benefit from all Calendar features, for example room suggestions.
- **workingLocationProperties.officeLocation.label**  
(/workspace/calendar/api/v3/reference/events#workingLocationProperties.officeLocation.label):  
This is the label that's shown on the Calendar Web and mobile clients. You can fetch building IDs and building labels using the **resources.buildings.list** (/workspace/admin/directory/reference/rest/v1/resources.buildings/list) method.

Creating and updating working location events through the batch endpoints isn't supported.

For feature details, go to [Set your working hours & location](https://support.google.com/calendar/answer/7638168)

(<https://support.google.com/calendar/answer/7638168>) and [Turn working location on or off for users](https://support.google.com/a/answer/10918567) (<https://support.google.com/a/answer/10918567>)

## How to show overlapping working location events

A user can have multiple working location events on their calendar at the same time which overlap, meaning that any given time could have multiple working locations set for it. In circumstances when only a single location can be shown to the user, they should be shown that location consistently across multiple applications. When doing this, use the following guidelines to choose which event to show:

- [Timed events](/workspace/calendar/api/concepts/events-calendars#types_of_events) (/workspace/calendar/api/concepts/events-calendars#types\_of\_events) take precedence over all-day events.
- Single events take precedence over recurring events and their [exceptions](/workspace/calendar/api/concepts/events-calendars#instances_exceptions) (/workspace/calendar/api/concepts/events-calendars#instances\_exceptions).
- Events which start later take precedence over events which start earlier.
- Events with shorter durations take precedence over those with longer durations.
- More recently created events take precedence over events that were created earlier.
- Partially overlapping events should be shown as two different events each with their own working location.

## Create status events in Google Apps Script

[Google Apps Script](/apps-script/overview) (/apps-script/overview) is a JavaScript-based cloud scripting language that lets you build business applications that integrate with Google Workspace. Scripts are developed in a browser-based code editor, and they are stored and run on Google's servers. See also [Google Apps Script quickstart](/workspace/calendar/api/quickstart/apps-script) (/workspace/calendar/api/quickstart/apps-script) to start using Apps Script to send requests to the Google Calendar API.


The following instructions describe how to manage status events using the [Google Calendar API](/apps-script/advanced/calendar) (/apps-script/advanced/calendar) as an advanced service in Google Apps Script. For a complete list of Google Calendar API resources and methods, see the [reference documentation](/workspace/calendar/api/guides/overview) (/workspace/calendar/api/guides/overview).

## Create and set up the script

1. Create a script by going to [script.google.com/create](https://script.google.com/create) (<https://script.google.com/create>).
2. On the left pane next to **Services**, click Add a service **+** .
3. Select **Google Calendar API** and click **Add**.
4. After enabled, the API appears on the left pane. Available methods and classes in the API can be listed using the *Calendar* keyword in the editor.

## (Optional) Update the Google Cloud project

Each Google Apps Script project has an associated Google Cloud project. Your script can use the default project that Google Apps Script automatically creates. If you want to use a custom Google Cloud project, take the following steps to update the project associated with your script.

1. On the left side of the editor, click Project Settings .
2. Under **Google Cloud Platform (GCP) Project**, click **Change project**.
3. Enter the project number of the Google Cloud project that's in the Developer Preview Program, and click **Set project**.
4. On the left side, select Editor **< >** to navigate back to the code editor.

## Add code to the script

The following code sample shows how to create, read, and list status events on your primary calendar.

1. Paste the following into the code editor.

★ *primary* is a keyword to access the primary calendar of the signed-in user. To retrieve events on a different calendar, set the **calendarId** parameter to an email address. This lets you read status events on the calendars of other users you have at minimum free/busy access to. Secondary calendars can't have status events.

```

/** Creates a focus time event. */
function createFocusTime() {
  const event = {
    start: { dateTime: '2023-11-14T10:00:00+01:00' },
    end: { dateTime: '2023-11-14T12:00:00+01:00' },
    eventType: 'focusTime',
    focusTimeProperties: {
      chatStatus: 'doNotDisturb',
      autoDeclineMode: 'declineOnlyNewConflictingInvitations',
      declineMessage: 'Declined because I am in focus time.',
    }
  }
  createEvent(event);
}

/** Creates an out of office event. */
function createOutOfOffice() {
  const event = {
    start: { dateTime: '2023-11-15T10:00:00+01:00' },
    end: { dateTime: '2023-11-15T18:00:00+01:00' },
    eventType: 'outOfOffice',
    outOfOfficeProperties: {
      autoDeclineMode: 'declineOnlyNewConflictingInvitations',
      declineMessage: 'Declined because I am on vacation.',
    }
  }
  createEvent(event);
}

/** Creates a working location event. */
function createWorkingLocation() {
  const event = {
    start: { date: "2023-06-01" },
    end: { date: "2023-06-02" },
    eventType: "workingLocation",
    visibility: "public",
    transparency: "transparent",
    workingLocationProperties: {
      type: 'customLocation',
      customLocation: { label: "a custom location" },
    }
  }
  createEvent(event);
}

```

```

/**
 * Creates a Calendar event.
 * See https://developers.google.com/workspace/calendar/api/v3/reference/e
 */
function createEvent(event) {
  const calendarId = 'primary';

  try {
    var response = Calendar.Events.insert(event, calendarId);
    var event = (response.eventType === 'workingLocation') ? parseWorkingLo
    console.log(event);
  } catch (exception) {
    console.log(exception.message);
  }
}

/**
 * Reads the event with the given eventId.
 * See https://developers.google.com/workspace/calendar/api/v3/reference/e
 */
function readEvent() {
  const calendarId = 'primary';

  // Replace with a valid eventId.
  const eventId = "sample-event-id";

  try {
    var response = Calendar.Events.get(calendarId, eventId);
    var event = (response.eventType === 'workingLocation') ? parseWorkingLo
    console.log(event);
  } catch (exception) {
    console.log(exception.message);
  }
}

/** Lists focus time events. */
function listFocusTimes() {
  listEvents('focusTime');
}

/** Lists out of office events. */
function listOutOfOffices() {
  listEvents('outOfOffice');
}

```



```

/** Lists working location events. */
function listWorkingLocations() {
  listEvents('workingLocation');
}

/**
 * Lists events with the given event type.
 * See https://developers.google.com/workspace/calendar/api/v3/reference/events
 */
function listEvents(eventType = 'default') {
  const calendarId = 'primary'

  // Query parameters for the list request.
  const optionalArgs = {
    eventTypes: [eventType],
    showDeleted: false,
    singleEvents: true,
    timeMax: '2023-04-01T00:00:00+01:00',
    timeMin: '2023-03-27T00:00:00+01:00',
  }
  try {
    var response = Calendar.Events.list(calendarId, optionalArgs);
    response.items.forEach(event =>
      console.log(eventType === 'workingLocation' ? parseWorkingLocation(event) : ''));
  } catch (exception) {
    console.log(exception.message);
  }
}

/**
 * Parses working location properties of an event into a string.
 * See https://developers.google.com/workspace/calendar/api/v3/reference/events
 */
function parseWorkingLocation(event) {
  if (event.eventType !== "workingLocation") {
    throw new Error("'" + event.summary + "' is not a working location event");
  }

  var location = 'No Location';
  const workingLocation = event.workingLocationProperties;
  if (workingLocation) {
    if (workingLocation.type === 'homeOffice') {
      location = 'Home';
    }
  }
}

```

```
    if (workingLocation.type === 'officeLocation') {  
        location = workingLocation.officeLocation.label;  
    }  
    if (workingLocation.type === 'customLocation') {  
        location = workingLocation.customLocation.label;  
    }  
}  
return `${event.start.date}: ${location}`;  
}
```

## Run the code sample

1. Above the code editor, select the function to run from the drop-down menu, and click **Run**.
2. In the first execution, it prompts you to authorize access. Review and allow Apps Script to access your calendar.
3. You can inspect the results of the script execution in the **Execution Log** that appears at the bottom of the window.

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-07-30 UTC.