Handle API errors

The Calendar API returns two levels of error information:

- HTTP error codes and messages in the header
- A JSON object in the response body with additional details that can help you determine how to handle the error.

The rest of this page provides a reference of Calendar errors, with some guidance on how to handle them in your app.

We strongly recommend testing (/workspace/calendar/api/guides/quota#test_quota_limit_handling) error handling of your application in a real environment.

Implement exponential backoff

The Cloud APIs documentation (https://cloud.google.com/storage/docs/retry-strategy) has a good explanation of exponential backoff and how to use it with the Google APIs.

Errors & suggested actions

This section provides the complete JSON representation of each listed error and suggested actions you might take to handle it.

400: Bad Request

User error. This can mean that a required field or parameter has not been provided, the value supplied is invalid, or the combination of provided fields is invalid.

```
{
 "error": {
  "errors": [
```

```
   {
    "domain": "calendar",
    "reason": "timeRangeEmpty",
    "message": "The specified time range is empty.",
    "locationType": "parameter",
    "location": "timeMax",
   }
  ],
  "code": 400,
  "message": "The specified time range is empty."
 }
}
```

Suggested action: Because this is a permanent error, do not retry. Read the error message instead and change your request accordingly.

401: Invalid Credentials

Invalid authorization header. The access token you're using is either expired or invalid.

```
{
 "error": {
  "errors": [
   {
    "domain": "global",
    "reason": "authError",
    "message": "Invalid Credentials",
    "locationType": "header",
    "location": "Authorization",
   }
  ],
  "code": 401,
  "message": "Invalid Credentials"
 }
}
```

Suggested actions:

- Get a new access token using the long-lived refresh token.

- If this fails, direct the user through the OAuth flow, as described in Authorizing requests with OAuth 2.0 (/workspace/calendar/auth).
- If you are seeing this for a service account, check that you have successfully completed all the steps in the service account page (/identity/protocols/OAuth2ServiceAccount).

403: User Rate Limit Exceeded

One of the limits from the Developer Console has been reached.

```
{
 "error": {
  "errors": [
   {
    "domain": "usageLimits",
    "reason": "userRateLimitExceeded",
    "message": "User Rate Limit Exceeded"
   }
  ],
  "code": 403,
  "message": "User Rate Limit Exceeded"
 }
}
```

Suggested actions:

- Make sure your app follows best practices from manage quotas (/workspace/calendar/api/guides/quota).
- Raise the per-user quota in the Developer Console project.
- If one user is making a lot of requests on behalf of many users of a Google Workspace account, consider using a service account with domain-wide delegation (/workspace/calendar/api/guides/quota#proper_accounting_with_service_accounts) and setting the `quotaUser` parameter.
- Use exponential backoff (#exponential-backoff).

403: Rate Limit Exceeded

The user has reached Google Calendar API's maximum request rate per calendar or per authenticated user.

```
{
 "error": {
  "errors": [
   {
    "domain": "usageLimits",
    "reason": "rateLimitExceeded",
    "message": "Rate Limit Exceeded"
   }
  ],
  "code": 403,
  "message": "Rate Limit Exceeded"
 }
}
```

Suggested action: `rateLimitExceeded` errors can return either 403 or 429 error codes—currently they are functionally similar and should be responded to in the same way, by using exponential backoff (#exponential-backoff). Additionally make sure your app follows best practices from manage quotas (/workspace/calendar/api/guides/quota).

403: Calendar usage limits exceeded

The user reached one of the Google Calendar limits in place to protect Google users and infrastructure from abusive behavior.

```
{
 "error": {
  "errors": [
   {
    "domain": "usageLimits",
    "message": "Calendar usage limits exceeded.",
    "reason": "quotaExceeded"
   }
  ],
  "code": 403,
  "message": "Calendar usage limits exceeded."
 }
```

```
}
```

Suggested actions:

- Read more on the Calendar usage limits in the Google Workspace Administrator help (https://support.google.com/a/answer/2905486).

403: Forbidden for non-organizer

The event update request is attempting to set one of the shared event properties in a copy that isn't the organizer's. Shared properties (for example, `guestsCanInviteOthers`, `guestsCanModify`, or `guestsCanSeeOtherGuests`) can only be set by the organizer.


```
{
 "error": {
  "errors": [
   {
    "domain": "calendar",
    "reason": "forbiddenForNonOrganizer",
    "message": "Shared properties can only be changed by the organizer of the ev
   }
  ],
  "code": 403,
  "message": "Shared properties can only be changed by the organizer of the even
 }
}
```


Suggested actions:

- If you're using Events: insert (/workspace/calendar/api/guides/reference/events/insert), Events: import (/workspace/calendar/api/guides/reference/events/import), or Events: update (/workspace/calendar/api/guides/reference/events/update), and your request doesn't include any shared properties, this is equivalent to trying to set them to their default values. Consider using Events: patch (/workspace/calendar/api/guides/reference/events/patch) instead.
- If your request has shared properties, make sure that you're only trying to change these properties if you're updating the organizer's copy.

404: Not Found

The specified resource was not found. This can happen in several cases. Here are some examples:

- when the requested resource (with the provided ID) has never existed
- when accessing a calendar that the user can not access

  { "error": { "errors": [{ "domain": "global", "reason": "notFound", "message": "Not Found" }], "code": 404, "message": "Not Found" } }

Suggested action: Use exponential backoff (#exponential-backoff).

409: The requested identifier already exists

An instance with the given ID already exists in the storage.

```
{
 "error": {
  "errors": [
   {
    "domain": "global",
    "reason": "duplicate",
    "message": "The requested identifier already exists."
   }
  ],
  "code": 409,
  "message": "The requested identifier already exists."
 }
}
```

Suggested action: Generate a new ID if you want to create a new instance, otherwise use the update (/workspace/calendar/v3/reference/events/update) method call.

409: Conflict

A batched item inside an `events.batch` (https://developers.google.com/workspace/calendar/api/guides/batch) operation can't be executed

due to an operational conflict with other requested batched items.

```
{
 "error": {
  "errors": [
   {
    "domain": "global",
    "reason": "conflict",
    "message": "Conflict"
   }
  ],
  "code": 409,
  "message": "Conflict"
 }
}
```

Suggested action: Exclude all successfully finished and all definitely failed batched items and retry the remaining ones in a different **`events.batch`** or corresponding single event operations.

410: Gone

The **`syncToken`** or **`updatedMin`** parameters are no longer valid. This error can also occur if a request attempts to delete an event that has already been deleted.

```
{
 "error": {
  "errors": [
   {
    "domain": "calendar",
    "reason": "fullSyncRequired",
    "message": "Sync token is no longer valid, a full sync is required.",
    "locationType": "parameter",
    "location": "syncToken",
    }
  ],
  "code": 410,
  "message": "Sync token is no longer valid, a full sync is required."
 }
}
```


or


```
{
 "error": {
  "errors": [
   {
    "domain": "calendar",
    "reason": "updatedMinTooLongAgo",
    "message": "The requested minimum modification time lies too far in the past
    "locationType": "parameter",
    "location": "updatedMin",
   }
  ],
  "code": 410,
  "message": "The requested minimum modification time lies too far in the past."
 }
}
```


or


```
{
 "error": {
```

```
  "errors": [
   {
    "domain": "global",
    "reason": "deleted",
    "message": "Resource has been deleted"
   }
  ],
  "code": 410,
  "message": "Resource has been deleted"
 }
}
```

Suggested action: For the `syncToken` or `updatedMin` parameters, wipe the store and re-sync. For more details see Synchronize Resources Efficiently (/workspace/calendar/v3/sync). For already deleted events, no further action is necessary.

412: Precondition Failed

The etag supplied in the If-match header no longer corresponds to the current etag of the resource.

```
{
 "error": {
  "errors": [
   {
    "domain": "global",
    "reason": "conditionNotMet",
    "message": "Precondition Failed",
    "locationType": "header",
    "location": "If-Match",
    }
  ],
  "code": 412,
  "message": "Precondition Failed"
 }
}
```

Suggested action: Re-fetch the entity and re-apply the changes. For more details see Get specific versions of resources (/workspace/calendar/api/guides/version-resources).

429: Too many requests

A `rateLimitExceeded` error occurs when the user has sent too many requests in a given amount of time.

```
{
  "error": {
    "errors": [
      {
        "domain": "usageLimits",
        "reason": "rateLimitExceeded",
        "message": "Rate Limit Exceeded"
      }
    ],
    "code": 429,
    "message": "Rate Limit Exceeded"
  }
}
```

Suggested action: `rateLimitExceeded` errors can return either 403 or 429 error codes—currently they are functionally similar and should be responded to in the same way, by using exponential backoff (#exponential-backoff). Additionally make sure your app follows best practices from manage quotas (/workspace/calendar/api/guides/quota).

500: Backend Error

An unexpected error occurred while processing the request.

```
{
 "error": {
  "errors": [
   {
    "domain": "global",
    "reason": "backendError",
    "message": "Backend Error",
   }
  ],
  "code": 500,
  "message": "Backend Error"
```

```
  }
}
```

Suggested action: Use exponential backoff (#exponential-backoff).

Last updated 2025-07-30 UTC.