Google Calendar API

Summary:

The Google Calendar Events API allows users to interact with their Google Calendar, enabling them to create, retrieve, update, and delete events. This documentation provides comprehensive information on setting up, running, and using the API.

Version	Date	Developer	Description
0.0	5-Nov-2023	Mahmoud Khaled Ragab	Initial

TABLE OF CONTENTS

Instructions

1. Intro

The Google Calendar Events API allows users to interact with their Google Calendar, enabling them to create, retrieve, update, and delete events. This documentation provides comprehensive information on setting up, running, and using the API.

2. Prerequisites

Before setting up and running the application, ensure you have the following prerequisites:

- .NET SDK installed on your machine.
- Visual Studio (optional but recommended for development).
- Google API project with the Calendar API enabled.
- Google API credentials (client ID and client secret) to access the Google Calendar API.

3. Project Setup

3.1. Obtaining Google API Credentials

To obtain Google API credentials, follow these steps:

1. Visit the [Google API Console] (https://console.developers.google.com/).
2. Create a new project if you don't have one.
3. Enable the "Google Calendar API" for your project.
4. In the "Credentials" tab, create new credentials of type "OAuth 2.0 Client ID."
5. Configure the OAuth consent screen with relevant information.
6. Select the application type (web application or desktop application) and add authorized redirect URIs.
7. After creating credentials, you will receive a `Client ID` and `Client Secret`. Keep these secure.

3.2. Configuring the Application

Now, configure your application to use the obtained credentials:

1. In your application code, locate where you need to configure the Google API credentials.

   Example: [\GoogleCalendarEvents\GoogleCalendarEvents\Manager\GoogleCredintial.cs]

2. Enable the Google Calendar API for your project.
3. Use the `Client ID` and `Client Secret` obtained from the Google API Console to set up OAuth 2.0 authentication.

   Example: [\GoogleCalendarEvents\GoogleCalendarEvents\GoogleAuthention\ClientCredintial.json]

3. Ensure the redirect URI matches the one configured in the Google API Console.

4. Running the Application

Follow these steps to run your application:

1. Clone or download the application source code from the repository.
2. Open the application solution in Visual.
3. Build the solution to restore NuGet packages.
4. Run the application, which will start the API server.

5. API Endpoints

Application exposes the following API endpoints:

Endpoint	HTTP Method	Description
`/api/events`	POST	Create a new event.
`/api/events/{eventId}`	GET	Retrieve an event by ID.
`/api/events/{eventId }`	PUT	Update an existing event.
`/api/events/{eventId}`	DELETE	Delete an event
`/api/events`	GET	Retrieve a list of events with optional filters
`/api/events/Search`	GET	Search for events with optional filtering

Status Codes

All status codes are standard HTTP status codes. The below ones are used in this API.

2XX - Success of some kind
4XX - Error occurred in client's part
5XX - Error occurred in server's part

Status Code	Description
200	OK
201	Created
202	Accepted (Request accepted, and queued for execution)
400	Bad request
401	Authentication failure
403	Forbidden
404	Resource not found
405	Method Not Allowed
409	Conflict
412	Precondition Failed
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

1. Create Event

1.1. End Point

- HTTP Method: POST.
- Endpoint: /api/events.

1.2. Description

- This endpoint allows users to create a new event in their Google Calendar.

1.3. Request

- The request body must contain a JSON object with the following properties:

- `Summary` (string, required): A brief summary of the event.
- `Description` (string): Additional information about the event.
- `Location` (string): The location where the event will take place.
- `Start` (string, required): The start date and time of the event in ISO 8601 format ("2023-11-03T18:37:07.426Z").
- `End` (string, required): The end date and time of the event in ISO 8601 format ("2023-11-03T18:37:07.426Z").
- Attachments (List of EventAttachment): A list of attachments associated with the event. Each attachment contains (string) FileId and (string) FileUrl properties.

1.4. Response

- Success Response (201 Created)
  - The response will contain the details of the created event.
  - The HTTP status code is 201 Created.

- Example response:

```
json
    {
      "Id": "1234",
      "Summary": "Event Summary",
```

```
        "Description": "Event Description",
        "Location": "Event Location",
        "Start": "2023-11-03T18:37:07.426Z",
        "End": "2023-11-03T18:37:07.426Z",
         "Attachments": [
            {
               "FileId": " file1234567",
               "FileUrl": "https://example.com/file1234567"
             }
         ]
}
```


1.5. Error Response

- If required fields are missing or formatted incorrectly, the endpoint will return a 400 Bad Request response with an error message.
- If the event's date is in the past or falls on a Friday or Saturday, the endpoint will return a 400 Bad Request response with an error message.
- If an unexpected error occurs during event creation, the endpoint will return a 500 Internal Server Error response with an error message.

1.6. Usage

- To create a new event, send a POST request to “/api/events” with a JSON request body containing the event details.
- Ensure that the event's start and end dates are in the future and not on a Friday or Saturday.

2. Get Event

1.1. End Point

- HTTP Method: GET.
- Endpoint: /api/events/{eventId}.

1.2. Description

- This endpoint allows users to retrieve details about a specific event from their Google Calendar.

1.3. Request

- URL Parameters

  - `eventId` (string, required): The unique identifier of the event to retrieve.

1.4. Response

- Success Response (200 OK)

  - The response will contain the details of the created event.
  - The HTTP status code is 201 Created.

- Example response:

```
json
{
    "Id": "1234567",
    "Summary": "Event Summary",
    "Description": "Event Description",
    "Location": "Event Location",
    "Start": "2023-11-03T18:37:07.426Z",
    "End": "2023-11-03T18:37:07.426Z",
    "Attachments": [
        {
            "FileId": "file1234567",
            "FileUrl": "https://example.com/file1234567"
```

```
        }
    ]
}
```

1.5. Error Response

- If the requested event does not exist, the endpoint will return a 404 Not Found response.

1.6. Usage

- To retrieve event details, send a GET request to “/api/events/{eventId}”, where `{eventId}` is the unique identifier of the event you want to retrieve.

3. Update Event

1.1. End Point

- HTTP Method: PUT.
- Endpoint: /api/events//{eventId}.

1.2. Description

- This endpoint allows users to update an existing event in their Google Calendar.

1.3. Request

- URL Parameters

  - `eventId` (string, required): The unique identifier of the event to update.

- Example request body:

  json

```
{
  "Summary": "Updated Event Summary",
  "Description": "Updated Event Description",
  "Location": "Updated Event Location",
  "Start": "2023-11-03T18:37:07.426Z",
  "End": "2023-11-03T18:37:07.426Z"
 "Attachments": [
      {
        "FileId": " file1234567",
        "FileUrl": "https://example.com/file1234567"
       }
            ]
}
```

1.4. Response

- Success Response (200 OK)

  - The response will contain the updated details of the event.

  - The HTTP status code is 200 OK.

- Example response:

  json

```
{
   "Id": "1234",
   "Summary": "Updated Event Summary",
   "Description": "Updated Event Description",
   "Location": "Updated Event Location",
   "Start": "2023-11-03T18:37:07.426Z",
   "End": "2023-11-03T18:37:07.426Z",
   "Attachments": [
      {
            "FileId": " file1234567",
             "FileUrl": "https://example.com/file1234567"
      }
   ]
}
```

1.5. Error Response

- If the requested event does not exist, the endpoint will return a 404 Not Found response.

1.6. Usage

- To update an event, send a PUT request to “/api/events/{eventId}”, where `{eventId}` is the unique identifier of the event you want to update. The updated event details should be included in the request body.
- Ensure that the event's start and end dates are in the future and not on a Friday or Saturday.

4. Delete Event

1.1. End Point

- HTTP Method: DELETE.
- Endpoint: /api/events/{eventId}.

1.2. Description

- This endpoint allows users to delete an event from their Google Calendar.

1.3. Request

- URL Parameters

  - `eventId` (string, required): The unique identifier of the event to delete.

1.4. Response

- Success Response (204 No Content)

  - The response will have no content.

  - The HTTP status code is 204 No Content, indicating a successful deletion.

1.5. Error Response

- If the `eventId` is missing, the endpoint will return a 400 Bad Request response.
- If the event is not found or other errors occur during deletion, the endpoint will return a 500 Internal Server Error response.

1.6. Usage

- To delete an event, send a DELETE request to `/api/events/{eventId}`, where `{eventId}` is the unique identifier of the event you want to delete.

5. Get All Events By Filter

1.1. End Point

- HTTP Method: GET.
- Endpoint: /api/events.

1.2. Description

- This endpoint allows users to retrieve events from their Google Calendar with optional filtering and pagination.

1.3. Request

- Query Parameters
  - `startDate` (DateTime, optional): Filter events with a start date greater than or equal to this date.
  - `endDate` (DateTime, optional): Filter events with an end date less than or equal to this date.
  - `searchQuery` (string, optional): Search events by summary (case-insensitive).

1.4. Response

- Success Response (200 OK).
  - The response includes a JSON object containing the following properties:
  - `TotalEvents` (int): The total number of events that match the filter criteria.
  - `TotalPages` (int): The total number of pages based on the specified page size.
  - `CurrentPage` (int): The current page number.
  - `Events` (array of GoogleCalendarEventDto): An array of event objects matching the filter criteria.

1.5. Error Response

- If the provided query parameters are invalid, the endpoint will return a 400 Bad Request response.

1.6. Usage

- To retrieve events, send a GET request to “/api/events”.
- You can include query parameters for filtering events based on `startDate`, `endDate`, and `searchQuery`.
- The response will include a paginated list of events that match the filter criteria along with pagination metadata.

6. Search Event (like Filtering)

1.1. End Point

- HTTP Method: GET.
- Endpoint: /api/events/search.

1.2. Description

- This endpoint allows users to search for events in their Google Calendar with optional filtering.

1.3. Request

- Query Parameters

  - `summary` (string, optional): Search for events with a summary that contains the specified text.
  - `description` (string, optional): Search for events with a description that contains the specified text.
  - `startDate` (DateTime, optional): Filter events with a start date greater than or equal to this date.
  - `endDate` (DateTime, optional): Filter events with an end date less than or equal to this date.

1.4. Response

- Success Response (200 OK)

  - The response includes a JSON array containing event objects that match the search criteria. Each event object has the following properties:
  - `Id` (string): The unique identifier of the event.
  - `Summary` (string): The event's summary.
  - `Description` (string): The event's description.
  - `Location` (string): The event's location.
  - `Start` (DateTime): The event's start date and time.
  - `End` (DateTime): The event's end date and time.

- `Attachments` (array of EventAttachment): An array of event attachments with `FileId` and `FileUrl` properties.

1.5. Error Response

- If no events match the search criteria, the endpoint will return a 404 Not Found response.

1.6. Usage

- To search for events, send a GET request to `/api/events/Search`.
- You can include query parameters to filter events by `summary`, `description`, `startDate`, and `endDate`.
- The response will include a list of events that match the search criteria.

Note: About Attachment

- Attach files from Google Drive to Google Calendar events, you will need to integrate the Google Drive API into your application. The Google Drive API allows you to access and manipulate files stored in a user's Google Drive account.
- To achieve this, follow these general steps:

  1. Set up the Google Drive API: You need to create a project in the Google Developers Console, enable the Google Drive API, and obtain API credentials (client ID and client secret).

  2. Authenticate your application: Implement OAuth 2.0 authentication to allow your application to access a user's Google Drive. You'll use the credentials obtained in step 1.

  3. Use the Google Drive API: Once authenticated, you can use the API to search for files in the user's Google Drive, obtain file IDs, and work with the files, such as downloading or attaching them to Google Calendar events.

  4. Associate Google Drive files with Google Calendar events: When creating or updating Google Calendar events, you can specify the file IDs or URLs of the files from Google Drive that should be associated with those events.

Conclusion

- The API provides a set of endpoints to manage Google Calendar events. It allows users to create, retrieve, update, and delete events. The API also supports filtering and searching for events. Proper validation and error handling are in place to ensure data integrity and provide meaningful responses.

- To set up and run the application, users need to obtain Google API credentials and configure the necessary settings for authentication and authorization. The endpoints are clearly documented, making it easy for developers to understand and use the API. Additionally, the API supports pagination and handles events' attachments seamlessly.

- By following the provided instructions, developers can create a robust and user-friendly application for managing Google Calendar events while adhering to best practices for API design and development.

By: Mahmoud Khaled – Software Developer
