Google+ API

The Google+ API is the programming interface to Google+. You can use the API to integrate your app or website with Google+. This enables users to connect with each other for maximum engagement using Google+ features from within your application.

API
Quota
Authorization
API Calls
Data Formats
Pagination
Partial Responses

API

View the Google+ API reference overview with resources People, Activities, Comments and Moments.

> **Note:** The Google+ API currently provides read-only access to public data. All API calls require either an OAuth 2.0 token or an API key.

Quota

Your application is limited to the number of API calls it can make by a usage courtesy quota. To view the courtesy limit and to request additional quota for your application, in the Google Developers Console, choose Quotas from the left menu for your project. On that page are two groups of API calls:

- **Google+ API (Sign-in)** - For methods that are allowed by the scope `https://www.googleapis.com/auth/plus.login` (people.get, people.list, moments.insert, moments.remove and moments.list)
- **Google+ API** - For all other methods

Each group has its own limit, which applies to the total of all API calls in that group.

Authorization

Many API calls require that the user of your application grant permission to access their data. Google uses the OAuth 2.0 protocol to allow authorized applications to access user data. To learn more, see OAuth.

API Calls

Most of the Google+ API follows a RESTful API design, meaning that you use standard HTTP methods to retrieve and manipulate resources. For example, to get the profile of a user, you might send an HTTP request like:

```
GET https://www.googleapis.com/plus/v1/people/userId
```

Common Parameters

Different API methods require parameters to be passed either as part of the URL path or as query parameters. Additionally, there are a few parameters that are common to all API endpoints. These are all passed as optional query parameters.

Parameter Name	Value	Description
`callback`	`string`	Specifies a JavaScript function that will be passed the response data for using the API with JSONP.
`fields`	`string`	Selector specifying which fields to include in a partial response.
`key`	`string`	API key. Your API key identifies your project and provides you with API access, quota, and reports. Required unless you provide an OAuth 2.0 token.
`access_token`	`string`	OAuth 2.0 token for the current user. Learn more about OAuth.
`prettyPrint`	`boolean`	If set to "true", data output will include line breaks and indentation to make it more readable. If set to "false", unnecessary whitespace is removed, reducing the size of the response. Defaults to "true".
`userIp`	`string`	Identifies the IP address of the end user for whom the API call is being made. This allows per-user quotas to be enforced when calling the API from a server-side application. Learn more about Capping Usage.

Data Formats

Resources in the Google+ API are represented using JSON data formats. For example, retrieving a user's profile may result in a response like:

```
{
  "kind": "plus#person",
  "id": "118051310819094153327",
  "displayName": "Chirag Shah",
  "url": "https://plus.google.com/118051310819094153327",
  "image": {
    "url": "https://lh5.googleusercontent.com/-XnZDEoiF09Y/AAAAAAAAAAI/AAAAAAAAYCI/7fow4a2UTM
U/photo.jpg"
  }
}
```

Common Properties

While each type of resource will have its own unique representation, there are a number of common properties that are found in almost all resource representations.

Property Name	Value	Description
`displayName`	`string`	This is the name of the resource, suitable for displaying to a user.
`id`	`string`	This property uniquely identifies a resource. Every resource of a given kind will have a unique `id`. Even though an `id` may sometimes look like a number, it should always be treated as a string.
`kind`	`string`	This identifies what kind of resource a JSON object represents. This is particularly useful when programmatically determining how to parse an unknown object.
`url`	`string`	This is the primary URL, or permalink, for the resource.

Pagination

In requests that can respond with potentially large collections, such as theh collection returned by `activities.list`, each response contains a limited number of items set by maxResults (default: 20). Each response also contains a `nextPageToken` property. To obtain the next page of items, you pass this value of `nextPageToken` to the `pageToken` property of the next request. Repeat this process to page through the full collection. For the last page, `nextPageToken` will be absent.

> **Note:** Page tokens become stale over time — if new items have been added to a list since you started paginating, they might not appear in the results. If you held a `pageToken` for a long period of time and want to continue paging, it might be better to restart pagination by repeating the original request by omitting the `pageToken`.

For an example of pagination, calling an `activities.list` method returns a response with `nextPageToken`:

```
{
  "kind": "plus#activityFeed",
  "title": "Plus Public Activities Feed",
  "nextPageToken": "CKaEL",
  "items": [
    {
      "kind": "plus#activity",
      "id": "123456789",
      ...
    },
    ...
  ]
  ...
}
```

To get the next page of activities, pass the value of this token in with your next `activities.list` request:

```
https://www.googleapis.com/plus/v1/people/me/activities/public?pageToken=CKaEL
```

As before, the response to this request includes `nextPageToken`, which you can pass in to get the next page of results. You can continue this cycle to get new pages.

Partial Responses

By default, the server sends back the full representation of a resource after processing requests. For better performance, you can ask the server to send only the fields you really need and get a *partial response* instead.

To request a partial response, use the `fields` request parameter to specify the fields you want returned. You can use this parameter with any request that returns a response body.

Example

The following example shows the use of the `fields` parameter with the Google+ API.

Simple request: This HTTP `GET` request omits the `fields` parameter and returns the full activity resource, with its dozens of fields.

```
https://www.googleapis.com/plus/v1/activities/z12gtjhq3qn2xxl2o224exwiqruvtda0i?key=YOUR-API-
KEY
```

Request for a partial response: This HTTP `GET` request for the above resource that uses the `fields` parameter significantly reduces the amount of data returned.

```
https://www.googleapis.com/plus/v1/activities/z12gtjhq3qn2xxl2o224exwiqruvtda0i?fields=url,ob
ject(content,attachments/url)&key=YOUR-API-KEY
```

In response to the above request, the server sends back a JSON response that contains only the `url` field and the pared-down `object` that includes only `content` and `attachments.url`.

```
{
 "url": "https://plus.google.com/102817283354809142195/posts/F97fqZwJESL",
 "object": {
  "content": "A picture... of a space ship... launched from earth 40 years ago.",
  "attachments": [
   {
    "url": "http://apod.nasa.gov/apod/ap110908.html"
   }
  ]
 }
}
```

Note that the response is a JSON object that includes only the selected fields and their enclosing parent objects.

Syntax of the `fields` parameter is covered next, followed by more detail about what gets returned in the response.

Fields parameter syntax summary

The format of the `fields` request parameter value is loosely based on XPath syntax. The supported syntax is summarized below; additional examples are provided in the following section.

- Use a comma-separated list to select multiple fields.
- Use `a/b` to select a field `b` that is nested within field `a`; use `a/b/c` to select a field `c` nested within `b`.
- Specify field sub-selectors to request only specific sub-fields by placing expressions in parentheses "`( )`" after any selected field.
  For example: `fields=items(id,object/content)` returns only the item `id` and object's `content`, for each `items` array element.

- Use wildcards in field selections, if needed.
  For example: `fields=items/object/*` selects all items in an object.

More examples of using the fields parameter

The examples below include descriptions of how the `fields` parameter value affects the response.

> **Note:** As with all query parameter values, the `fields` parameter value must be URL encoded. For better readability, the examples in this document omit the encoding.

***Field selections*: Identify the fields you want returned.**

The `fields` request parameter value is a comma-separated list of fields, and each field is specified relative to the root of the response. Thus, if you are performing a list operation, the response is a collection, and it generally includes an array of resources. If you are performing an operation that returns a single resource, fields are specified relative to that resource. If the field you select is (or is part of) an array, the server returns the selected portion of all elements in the array.

Here are some collection-level examples from activities.list:

Examples	Effect
`items`	Returns all elements in the `items` array, including all fields in each element, but no other fields.

`updated,items`	Returns both the `updated` field and all elements in the `items` array.
`items/title`	Returns only the `title` field for all elements in the `items` array. Whenever a nested field is returned, the response includes the enclosing parent objects. The parent fields do not include any other child fields unless they are also selected explicitly.

Here are some resource-level examples from activities:

Examples	Effect
`title`	Returns the `title` field of the requested resource.
`actor/displayName`	Returns the `displayName` sub-field of the `actor` object in the requested resource.
`object/attachments/url`	Returns only the `url` field for all members of the `attachments` array, which is itself nested under the `object` object.

***Field sub-selections*: Request only parts of the selected elements.**

By default, if your request specifies particular objects, the server returns the objects in their entirety. You can specify a response that includes only certain sub-fields within the selected objects. You do this using "`( )`" sub-selection syntax, as in the example below:

Example	Effect
`items(id,url)`	Returns only the values of the `id` and `url` fields for each element in the `items` array.

Handling partial responses

After a server processes a valid request that includes the `fields` query parameter, it sends back an HTTP `200 OK` status code, along with the requested data. If the `fields` query parameter has an error or is otherwise invalid, the server returns an HTTP `400 Bad Request` status code, along with an error message telling the user what was wrong with their fields selection, for example, "`Invalid field selection a/b`".

> **Note:** Whenever possible, use `maxResults` judiciously to reduce the results of each query to a manageable size. Otherwise, the performance gains possible with partial response might not be realized.

Last updated May 3, 2014.