Authorizing API requests

Every request that your application sends to the Google+ API needs to identify your application to Google. You can use the *API key* you get when defining your project, or you can use an OAuth 2.0 *access token*. You should use an access token when you are making calls on behalf of a given user.

Acquiring and using an API key

For Google+ API calls that do not need to identify a particular user, you can use an application API key. This is useful for server-side applications, or web applications that do not require the user to sign in with Google.

To create an API key:

1. Go to the Google Developers Console.
2. Create or select a project.
3. In the sidebar on the left, select **APIs & auth**.
4. In the displayed list of APIs, find the **Google+ API** and set its status to **ON**.
5. In the sidebar on the left, select **Credentials**.
6. Create an API key by clicking **Create New Key**. Select the appropriate kind of key: **Server key**, **Browser key**, **Android key**, or **iOS key**. Then click **Create**.

After you have an API key, your application can append the query parameter `key=yourAPIKey` to all request URLs. The API key is safe for embedding in URLs; it doesn't need any encoding.

For more information, see API keys. To keep your API keys secure, follow the best practices for securely using API keys.

Authorizing requests with Google+ Sign-In

Requests that your app makes to the Google+ API for non-public user data must be authorized by an authenticated user. Google recommends using Google+ Sign-In.

Use the Google+ Sign-In button to authenticate users and enable them to authorize your application. Under the covers, this button uses OAuth 2.0 to access the Google+ API. Detailed instructions for integrating the Sign-In button are available for web, Android, and iOS platforms.

You can use the Google APIs client libraries to handle your OAuth 2.0 flows. These libraries are available in most programming languages, as demonstrated in our quick-start sample apps. For purely server-side API access, we provide examples that use those libraries in many languages.

Incremental authorization

Google supports *incremental authorization*, which enables your app to request initial permissions at sign-in and later can request additional permissions, typically just before they are needed. For example, if your app allows users to save music playlists to Google Drive, you can ask for basic user information at sign-in, and later ask just for Google Drive permissions when the user is ready to save their first playlist. At this point, the consent dialog box asks the user only for the new permissions, which gives them a simpler, in-context decision to make.

You might use this technique if you suspect users are not signing in because your consent screen is overwhelming, or are confused as to why they are being asked for certain permissions. This feature can improve your sign-in conversion rate if you configure your initial scopes to be only the minimum that your app requires to get the user started. For implementation, see incremental authorization for web, Android, or iOS.

Authorization scopes

Scopes are strings that enable access to particular resources, such as user data. You include a scope in certain authorization requests, which then displays appropriate permissions text in a consent dialog that is presented to a user. Once the user consents to the permissions, Google sends your app *tokens*, which identify the specific authorization grant. In other words, the scopes and tokens determine what user data the user gives your app permission to access.

An app that makes unauthenticated calls (where no scope is requested) can access only user data that is public on Google+. For example, if your app does an unauthenticated search for public posts, the search response includes a user ID for each person who posted, and your app can then can access the user's name and photo URL, which are always public, and a birthday or gender if the user has made them public.

Try out all of the Google APIs, including the Google+ API, and view their scopes at the OAuth 2.0 Playground.

Scopes that conform to the OpenID Connect standard have full names that are short: `profile`, `email` and `openid`—they are *not* in the form of a URI. On the other hand, Google-specific scopes are in the form of a URI, such as `https://www.googleapis.com/auth/plus.login`.

The following scopes, with user consent, provide access to otherwise restricted user data.

Scope - fully qualified name	Description
Login scopes	
profile	This is the basic login scope. This scope does the following: • It requests that your app be given access to the authenticated user's basic profile information. • It lets you know who the currently authenticated user is by letting you replace a Google+ user ID with "`me`", which represents the authenticated user, in any call to the Google+ API. • It lets your web app access over-the-air Android app installs.
https://www.googleapis.com/auth/**plus.login**	This is the recommended login scope providing access to social features. This scope implicitly includes the `profile` scope and also requests that your app be given access to: • the age range of the authenticated user • the list of circled people that the user has granted your app access to know • the methods for reading, writing and deleting app activities (moments) to Google on behalf of the user In addition, this scope enables cross-platform single sign-on.
Email scopes	
email	This scope requests that your app be given access to: • the user's Google account email address. You access the email address by calling `people.get`, which returns the `emails` array (or by calling `people.getOpenIdConnect`, which returns the `email` property in OIDC-compliant format). • the name of the Google Apps domain, if any, that the user belongs to. The domain name is returned as the `domain` property from `people.get` (or `hd` property from `getOpenIdConnect`). This `email` scope is equivalent to and replaces the `https://www.googleapis.com/auth/userinfo.email` scope. Also see the related scope: `https://www.googleapis.com/auth/plus.profile.emails.read`.
https://www.googleapis.com/auth/**plus.profile.emails.read**	This scope requests that your app be given access to: • the user's Google account email address, as well as any public, verified email addresses in the user's Google+ profile. You access the email addresses by calling `people.get`, which returns the `emails` array. • the name of the Google Apps domain, if any, that the user belongs to. For more details about access to the domain name, see the `email` scope. Also see the related `email` scope.
Other scopes	
openid	This scope informs the authorization server that the client is making an OpenID Connect request, and requests access to the authenticated user's ID. You must include this scope with the other OpenID Connect scopes. The getOpenIdConnect method returns the user's profile in OIDC-compliant format—use the following HTTP request path: `https://www.googleapis.com/plus/v1/people/me/openIdConnect`

	Learn more about OpenID Connect for sign-in and OAuth 2.0 for Login (OpenID Connect).
https://www.googleapis.com/auth/**plus.me**	For login purposes, use the `profile` or `https://www.googleapis.com/auth/plus.login` scope. The `https://www.googleapis.com/auth/plus.me` scope is not recommended as a login scope because, for users who have not upgraded to Google+, it does not return the user's name or email address. This scope does the following: • It lets you know who the currently authenticated user is by letting you replace a Google+ user ID with "`me`", which represents the authenticated user, in any call to the Google+ API.
Deprecated scopes	
https://www.googleapis.com/auth/**userinfo.profile**	***Deprecated*** *- replace with the equivalent profile scope.* This scope is equivalent to the profile scope and requests access to the same data. **Note:** This scope is deprecated; however, it will be maintained and kept available for backward compatibility. For an explanation about this change, see Migrating to Google+ Sign-In.
https://www.googleapis.com/auth/**userinfo.email**	***Deprecated*** *- replace with the equivalent* `email` *scope.* This scope requests access to the user's Google account email address. Google generates new tokens with this scope for the `people.get` endpoint. This scope also requests access from the user to the `userinfo` endpoint for backward compatibility. Also see the related scope: `https://www.googleapis.com/auth/plus.profile.emails.read`. **Note:** This scope is deprecated; however, it will be maintained and kept available for backward compatibility. For an explanation about this change, see Migrating to Google+ Sign-In.

Revoking access to a token or application

At any time, a user can revoke access to any application:

- For applications that create app activities, users can revoke actions using the App Settings page.
- For *all* applications, including those that write app activities, a user can revoke authorization by following the "Websites authorized to access the account" link from the Google Dashboard. Users can go directly to the authorization manager at https://accounts.google.com/issuedauthsubtokens and revoke access.

To programmatically revoke access for any given access token, see the instructions for web, Android and iOS. This will also revoke any associated refresh token.

Last updated September 10, 2014.