Google APIs and Credentials

Version 1.0 [2017-07-31]

By Chris Koeritz

Accounts

You are probably already familiar with the google account. You must have a google account to use the google developer console and any google APIs. Any google account can be used, such as your gmail identity.

All of the projects and other items you create at the developer console are owned by the google account that is currently logged in. You may want to create new google account per customer if you intend to bill the customer for any charges incurred by their site when using google's APIs; this might simplify tracking for the bills. Alternatively, if you are given access to the customer's own google account, you could set up all of the google assets on it, and the customer's overage charges would go to their own account. The best approach for this probably varies per business.

Setting up a new google account is documented here: https://support.google.com/mail/answer/56256?hl=en

Google may require you to register a credit card for billing, if you use certain services. The billing management site can be found here: https://console.developers.google.com/billing

Projects

The google 'project' is the outermost container for API information and credentials at the developer console. To set a project up, first go to the google developer console API dashboard after logging in as the appropriate google account: https://console.developers.google.com/apis/dashboard

If no projects exist yet, the dashboard offers you a 'create' button to start a new project. If there are already existing projects, then to create a new project, bring up the selection dialog by clicking on the current project's name (top-left of window). The resulting 'Select' dialog lets you switch between your projects, but it will also create a new project if you click the large '+' button on the top right.

There seems to be a limit on the number of projects a google developer can create (twelve or so). I am not sure if handing google more money would remove that limit, but it seems likely.

Google APIs

Google groups its various services under the umbrella of a set of named APIs. Each major API category (such as google drive) can require several specific APIs to be enabled on your project before

you can use the associated services. For example, the maps service offers several APIs: some users may only want the javascript maps API, while others may need the static maps API.

You can see all of the APIs that are currently enabled for your project on the dashboard: https://console.developers.google.com/apis/dashboard The dashboard also has an "Enable API" button that allows you to enable new APIs.

If you would like more information about the APIs and their requirements, then visit the API library: https://console.developers.google.com/apis/library Each link provides information about the API, including authentication requirements and documentation.

API Keys

The google API key is used as a form of bearer credential that represents your project. They are used for APIs that need to be passed a key during requests for google services, for example when requesting a map from the google maps API. The API key reflects all of the APIs that you have enabled for your project, and are 'active' in that, if you add more enabled APIs or remove APIs from your project, the key will reflect whatever APIs are enabled at the time of the request for service.

The existing API keys can be viewed from this page: https://console.developers.google.com/apis/credentials A new API key can be created using the 'Create Credentials' button.

This page has a little more information about creating API keys for the Google Maps service: https://developers.google.com/maps/documentation/javascript/get-api-key Google tends to provide API key creation info per major service, so there may be more appropriate API key information pages available for your specific services (search on 'create google api key' to see some of them).

It is very important to limit the API key to lock it in to a specific site or application. This prohibits someone from stealing the API key and using it to get free google services on your dime. The limits are set by clicking on the API key in the credentials list and setting the 'Key Restriction' appropriately. Probably setting the 'referrers' is the most common restriction used, since it allows only your set of specific web sites to use your API key.

OAuth 2.0 Credentials

Oauth 2.0 credentials are used to store successful authorizations gained by our web site (or application) to use google services at our user's behest. In general, oauth allows service A to ask service B for permission to user U's data that is held on service B. Effectively, service A can now impersonate user U within the narrow bounds of the granted authorization on service B.

For example, a new web-based file management service 'bebox' might be designed for storing your data (similar to dropbox, etc). The bebox service might enable you to link your personal google drive to bebox in order to extend your awesome data management powers. To do this, bebox has to ask google for permission to use the stuff google manages for you, which is where the oauth authorization

comes in. Google issues your application an oauth ‘access token’ which it can present to google from then on. You are not exposing your google password to the bebox service, but merely granting that bebox can look at your google files on your behalf without bugging you again.

Any authorizations that you grant on google services to various applications can also be revoked at your google ‘myaccount’ site: https://myaccount.google.com/permissions Revoking permissions is always an option for the user, so the developer must be aware of this when designing code that relies on the returned oauth access tokens. There’s always the chance a seemingly valid token will stop working if the user revoked access, and at that point authorization must be reattempted.

Creating oauth credentials to use in your application is luckily pretty straightforward. On the developer console, it is located here: https://console.developers.google.com/apis/credentials You may have to choose the appropriate project since google will open the last one you were using. The ‘Create Credentials’ button can be clicked to create a new OAuth credential that your application can use to initiate the OAuth authorization process.

This guide more fully describes the process of creating credentials in its ‘Create authorization credentials’ section: https://developers.google.com/api-client-library/php/auth/web-app

The redirect URI(s) that you configure per credential are very important. During OAuth 2.0 authorization, google will jump off site to its own authorization page to interact with the user, and the redirect URI tells google where to jump back into your site after authorization. Picking the appropriate URI depends on how you’ve designed your web site, including which controller is going to manage the google login process. For example, if my Users controller has a ‘googleLogin’ method that google should come back to after authorization, then the URL might be “https://blorp.tv/users/google_login”. Only the URIs configured on the OAuth credential can be used for the redirect, and in fact only one specific registered URI can be used per authorization request. If you registered more than one redirect URI, you will have to call setRedirectUri() on your google client before authorization and provide just one of the registered URIs for redirection.

To enable a CakePHP application for OAuth 2.0 authorization, we are currently relying on the downloadable ‘client_secret_….json’ file that can be downloaded from the credentials list. In the credentials list, click the down arrow to download the appropriate client secret file from the list. Rename the downloaded file to ‘client_secret.json’ and store it in the ‘config’ directory of your CakePHP application. Given the set of information in that file, our oauth code can use the google PHP client to take the user through the authorization process. What actually gets authorized totally depends on the APIs enabled in your project, and on the ‘scopes’ requested at authorization time.

This is a list of the available google scopes:
https://developers.google.com/identity/protocols/googlescopes Knowing which scopes to request will be defined by your intended application, and some more research in the google guides is probably required.

Required APIs for Google Maps

To use the features we've relied on for google maps so far, these APIs need to be enabled:

- Google Maps JavaScript API
- Google Maps Geocoding API
- Google Maps Distance Matrix API
- Google Static Maps API

Required APIs and Scopes for Google Calendar

- Google Calendar API
- Google_Service_Calendar::CALENDAR_READONLY scope (does not allow editing of user's calendar events)