

Using Gmail with XOAUTH2

Edit

New Page

[Jump to bottom](#)

Marcus Bointon edited this page on 8 Sep 2020 · 14 revisions

PHPMailer 5.2.11 added support for [Google's XOAUTH2](#) SMTP & IMAP authentication mechanism (see [this PR](#)). Using it is *very* complicated compared with other auth mechanisms, so this page is here to tell you what you need to do.

Note

[PHPMailer 6.0](#) has substantially improved support for OAuth2, including support for many servers other than Google; this document has not been updated for PHPMailer 6.0, but what you need to do is broadly similar.

Background

The XOAUTH2 mechanism replaces the usual username and password combination that allows you to authenticate against an SMTP server, which will then allow you to send messages through it. In December 2014 Google made this their preferred method of authenticating for Gmail for both sending and receiving email, and deprecated other systems. If you're not using XOAUTH2, you may experience authentication failures as described in [the troubleshooting guide](#). You will usually only need to set it up once for your PHP app (not each time you need to send a message), and while it's possible to share the same config across multiple apps, much of the supposed security improvement provided by OAuth stems from using separate client configs for each place you use it, so it's perfectly reasonable to create multiple Client IDs using the process shown below.

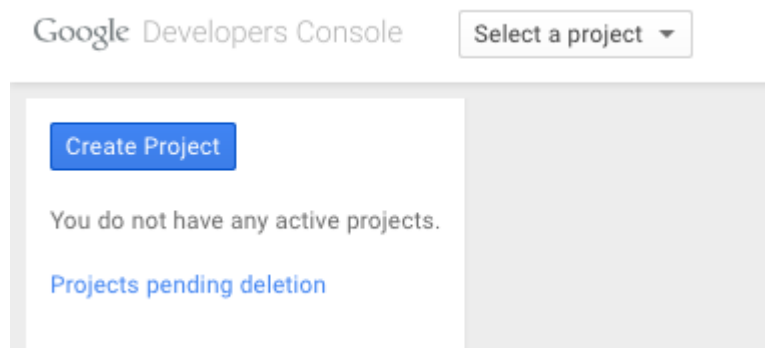
There is an earlier version of this system called `XOAUTH` (which Gmail also supports), but PHPMailer only supports the more recent `XOAUTH2`.

Important!

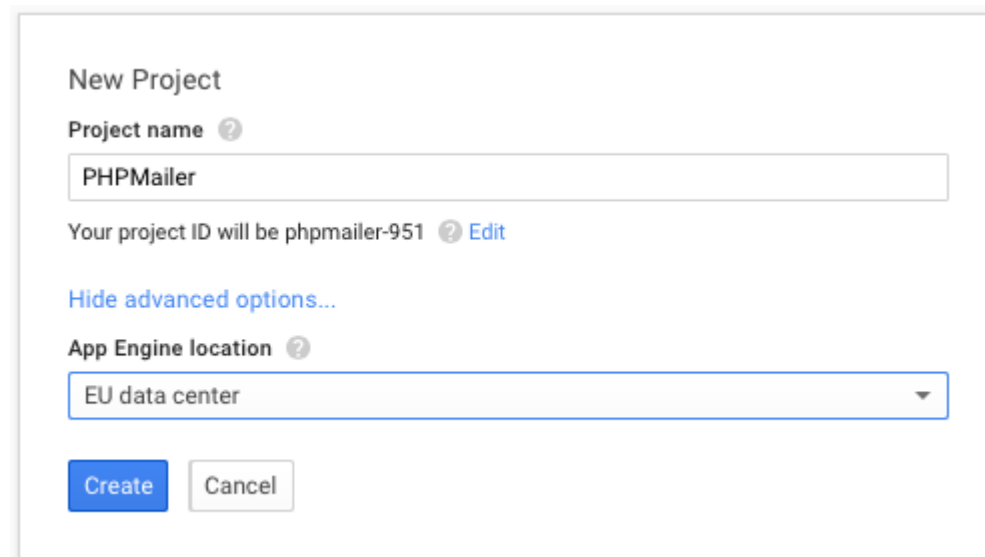
The [OAuth2 libraries this depends upon](#) require **PHP 5.5 or later**, so you need to be running at least that in order to be able to use this authentication system. Because of this requirement, this package is **not** enabled by default in PHPMailer's `composer.json` file, but appears [in the 'suggests' section](#). You should take the suggested package and add it to your own `composer.json` file (the same one you use to install PHPMailer itself in your own project, **not** PHPMailer's own composer file), and then re-run `composer install` to load it. This example requires "league/oauth2-google" package.

Configure an OAuth2 app

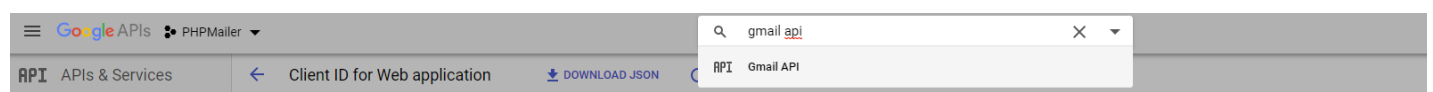
First of all you need to log into your google account using your usual username and password and go to the [developer console](#). This will list any existing Google API projects you are using already and allow you to create new ones. Click the "Create Project" button:



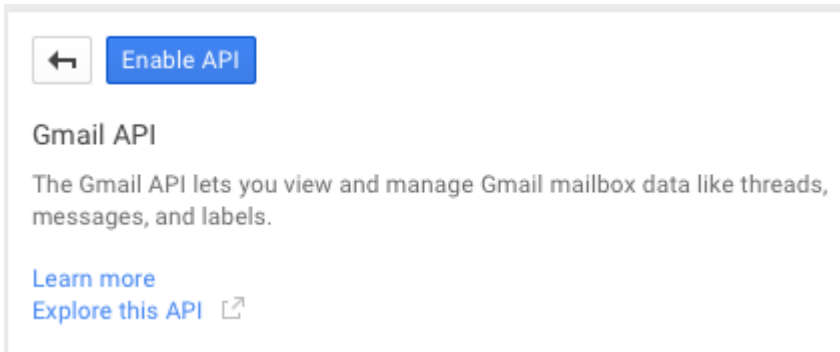
Give your project a name that reflects its use, for example "www.example.com contact form" and select a data center (in advanced options).



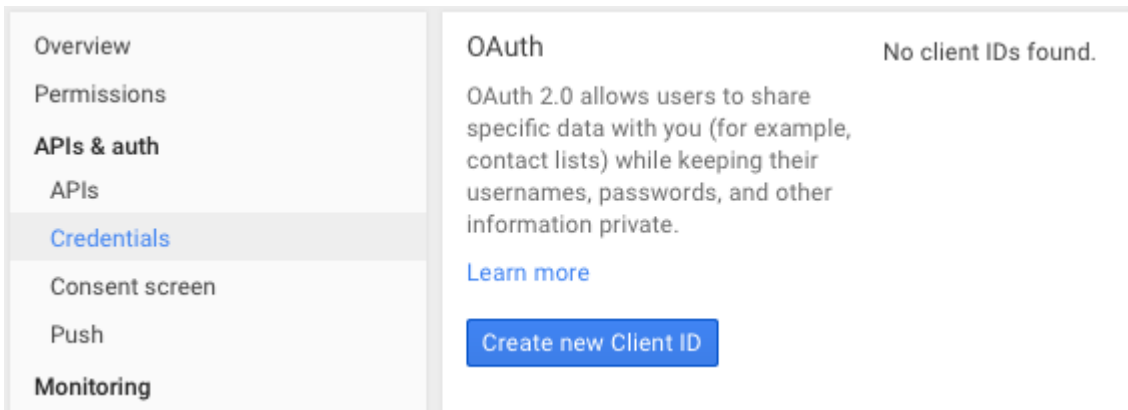
When the project has been created, type "gmail api" in the top search bar, and click on **Gmail API**.



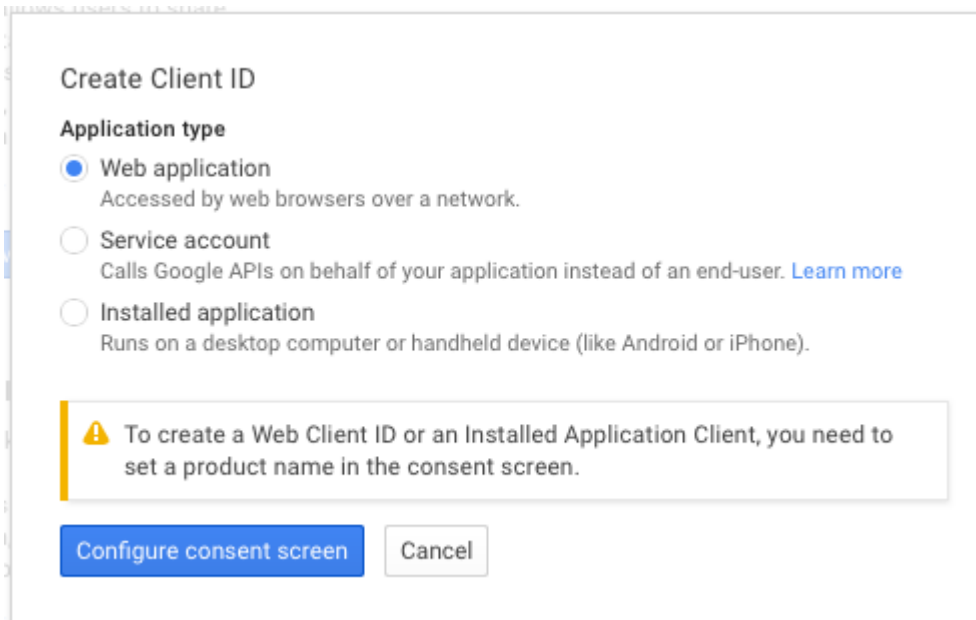
Clicking the **Gmail API** link will ask you to enable it, so click the **Enable API** button:



Now select **Credentials** from the left-hand menu, then click the **Create new Client ID** button:



There are various different ways you can use Google's OAuth systems, but we will use the most straightforward type: **Web application**, so leave that selected and click **Configure consent screen**.



This authentication is going to use your Gmail login, so select the email address you want to use, and give this Client app a name (it doesn't have to be PHPMailer). All the other settings can be left at defaults. Then click save.

Consent screen

The consent screen will be shown to users whenever you request access to their private data using your client ID

Note: This screen will be shown for all of your applications registered in this project

Email address

Choose your email ▼

Product name

PHPMailer|

Homepage URL (Optional)

Product logo (Optional) ?



This is how your logo will look to end users

Max size: 120x120 px

Privacy policy URL (Optional)

Terms of service URL (Optional)

Google+ page ID (Optional) ?

plus.google.com/ page ID

Save

Reset

To make the client ID, select the same **Web application** option. For the purposes of this example, we don't need to worry about Javascript origins, so that can be left blank. The Authorized redirect URLs must contain a pointer to a script that will validate your OAuth request, and here we're using the script that is bundled with PHPMailer called `get_auth_token.php`, which you need to configure to be run by a web server. Here I've just used localhost - you need to be able to run the script yourself, it does not need to be accessible to the outside world, but it will need to connect to google. Click **Create client ID** and it will create the ID (amazing!):

HOW TO SHARE

Create Client ID

Application type

☒ **Web application**
Accessed by web browsers over a network.

☐ **Service account**
Calls Google APIs on behalf of your application instead of an end-user. [Learn more](#)

☐ **Installed application**
Runs on a desktop computer or handheld device (like Android or iPhone).

Authorized JavaScript origins
Cannot contain a wildcard (http://*.example.com) or a path (http://example.com/subdir).

Authorized redirect URIs
One URI per line. Needs to have a protocol, no URL fragments, and no relative paths. Can't be a public IP Address.

http://localhost/phpmailer/get_oauth_token.php

[Create Client ID](#) [Cancel](#)

It will then show you the properties of the client ID:

OAuth

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.

[Learn more](#)

[Create new Client ID](#)

Client ID for web application

Client ID	237644427849-g8d0pnkd1jh3idcjbopvkse2hvj0tdp.apps.googleusercontent.com
Email address	237644427849-g8d0pnkd1jh3idcjbopvkse2hvj0tdp@developer.gserviceaccount.com
Client secret	mkIHhrns6eF-qjwuiLpSB4DL
Redirect URIs	http://localhost/phpmailer/get_oauth_token.php
JavaScript origins	none

[Edit settings](#) [Reset secret](#) [Download JSON](#) [Delete](#)

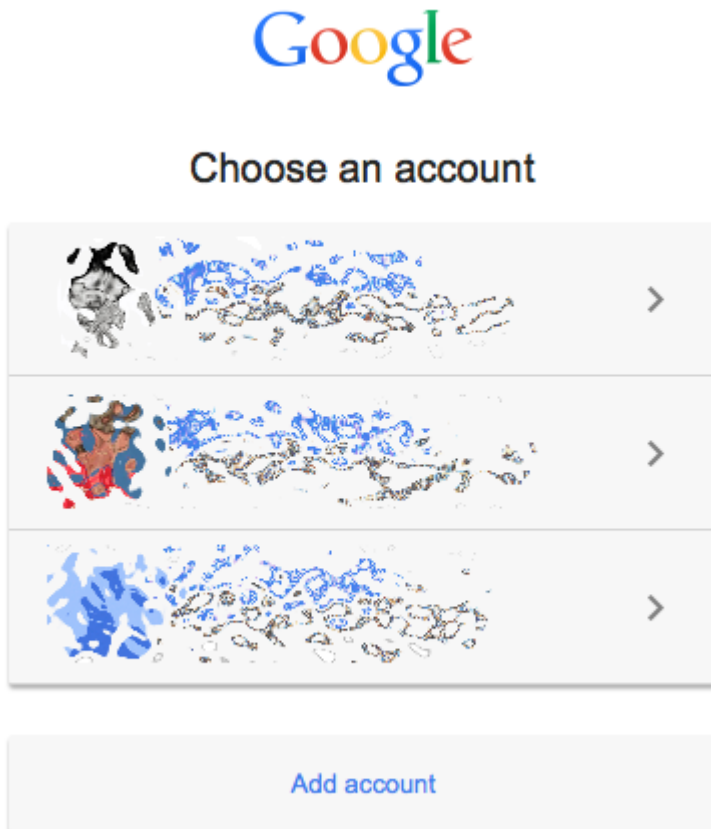
Now you need to take these values and copy and paste them into the `get_auth_token.php` script, like this:

```
$redirectUri = 'http://localhost/phpmailer/get_oauth_token.php';
$clientId = '237644427849-g8d0pnkd1jh3idcjbopvkse2hvj0tdp.apps.googleusercontent.com';
$clientSecret = 'mkIHhrns6eF-qjwuiLpSB4DL';
```

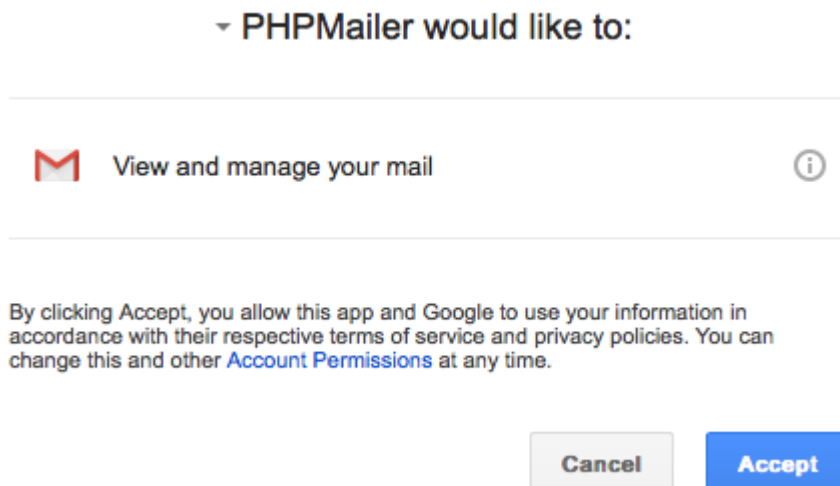
The redirect URL needs to be set to point at the script too (yes, it does need to point at itself).

Fetch the token

Now paste the script URL into your browser and it will ask you to select which Google ID you want to use (if you have more than one):



Then it will ask you for permission to allow access to this app:



Finally it will use your redirect to reload the script, and produce a refresh token:

Refresh Token: 1/7Jt8_RHX86Pk09VTfQd4O_ZqKbmuV7HpMNz-rqJ4KdQMEudVrK5jSpoR30zcRFq6

This is what we did all this work for: this value is what you need to perform XOAUTH2 authentication when connecting to Gmail's SMTP server.

Google won't reissue a refresh token unless the old one has been revoked or has expired – you will just get an empty token. If that happens, revoke the refresh token via the Google account's permissions screen, then run `get_oauth_token.php` again and reauthorise the permissions, and this time you should get a refresh token.

I have shown the real values used in this example, but this Client ID has now been deleted so it won't work any more (the ability to do this is an OAuth feature) - you must generate your own!

🔗 Configure your email script

In the examples folder you will find a script called `gmail_xoauth.phps` you can use as the basis for this process. How you use PHPMailer remains largely unchanged relative to the other examples; The differences are that it uses the `PHPMailerOAuth` class instead of the simpler `PHPMailer`, and you need to set the `AuthType` property to `XOAUTH2` and fill in the authentication values you obtained above.

Your gmail address goes into `oauthUserEmail` (and will generally need to be the same as your *From* address, though you can also use the **Email address** value that is shown with the Client ID above), **Client ID** goes into `oauthClientId`, **Client secret** into `oauthClientSecret` and the refresh token in `oauthRefreshToken`, like this:

```
$mail->oauthUserEmail = "<your gmail address>@gmail.com";
$mail->oauthClientId = "237644427849-g8d0pnkd1jh3idcjd bopvkse2hv j0tdp.apps.googleusercontent.
$mail->oauthClientSecret = "mklHhrns6eF-qjwu iLpSB4DL";
$mail->oauthRefreshToken = "1/7Jt8_RHX86Pk09VTfQd4O_ZqKbmuV7HpMNz-rqJ4KdQMEudVrK5jSpoR30zcRFq6"
```

If you run the script with debug output enabled, you should look for a sequence like this after `STARTTLS`:

```
2015-05-21 00:45:46      CLIENT -> SERVER: EHLO Oc.local
2015-05-21 00:45:46      SERVER -> CLIENT: 250-mx.google.com at your service,
[2a02:8424:2a38:e00:a102:53e4:c91f:4cff]
                        250-SIZE 35882577
                        250-8BITMIME
                        250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN
```

XOAUTH

250-ENHANCEDSTATUSCODES

250-PIPELINING

250-CHUNKING

250 SMTPUTF8

2015-05-21 00:45:47 CLIENT -> SERVER: AUTH XOAUTH2

dXNlcj1tYXJjdXMuYm9pbnRvbkbBnbWFpbC5jb20BYXV0aD1CZWYyZXIgeWEyOS5lZ0h5NGpXbkZaZFpMaEctV3g1ZUVt

2015-05-21 00:45:47 SERVER -> CLIENT: 235 2.7.0 Accepted

You can see the client specifying XOAUTH2, and the subsequent `235 2.7.0 Accepted` from the server tells us that the authentication was successful, and now we can go ahead with sending our message as usual.

Credit where it's due

The XOAUTH2 support for PHPMailer was written by [@sherryl4george](#), so a big round of applause please!

© 2021 by wiki contributors.



▼ Pages 13

[Home](#)

[About the CVE 2016 10033 and CVE 2016 10045 vulnerabilities](#)

[Diagnosing email sending twice \(or more\)](#)

[Gmail XOAUTH smtp connect\(\) failed. Trouble shooting](#)

[Gmail XOAUTH2 Using Google API Client](#)

[Improving delivery rates, avoiding spam filters](#)

[Overriding the SMTP class](#)

[Sending to lists](#)

[SMTP Debugging](#)

[Troubleshooting](#)

[TroubleshootingSMTP](#)

[Tutorial](#)

Using Gmail with XOAUTH2

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/PHPMailer/PHPMailer.wiki.git>

