

# Using OAuth2 for Authorization

## OAuth Credentials

OAuth credentials can be generated in several different ways using the [oauth2client](#) library provided by Google. If you are editing spreadsheets for yourself then the easiest way to generate credentials is to use *Signed Credentials* stored in your application (see example below). If you plan to edit spreadsheets on behalf of others then visit the [Google OAuth2 documentation](#) for more information.

## Using Signed Credentials

1. Head to [Google Developers Console](#) and create a new project (or select the one you have.)
2. Under “API & auth”, in the API enable “Drive API”.


---

[API Library](#)   [Enabled APIs \(2\)](#)

---

Some APIs are enabled automatically. You can disable them if you're not using their services.

[API](#) ^

	Quota	
Drive API	<div><div></div></div> 0%	<a href="#">Disable</a>
Drive SDK		<a href="#">Disable</a> 

3. Go to “Credentials” and hit “Create new Client ID”.

< Projects

---

**gsread-april**

Overview

Permissions

Billing & settings

**APIs & auth**

APIs

[Credentials](#)

Consent screen

Push

**Monitoring**

**Source Code**

**Compute**

**Networking**

**Storage**

**Big Data**

## OAuth

No client IDs found.

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](#)

---

## Public API access

No keys found.

Use of this key does not require any user action or consent, does not grant access to any account information, and is not used for authorization.

[Learn more](#)

[Create new Key](#)

4. Select "Service account". Hitting "Create Client ID" will generate a new Public/Private key pair.

OAuth

No client IDs found.

OAuth 2.0 allows users to

### Create Client ID

**Application type**

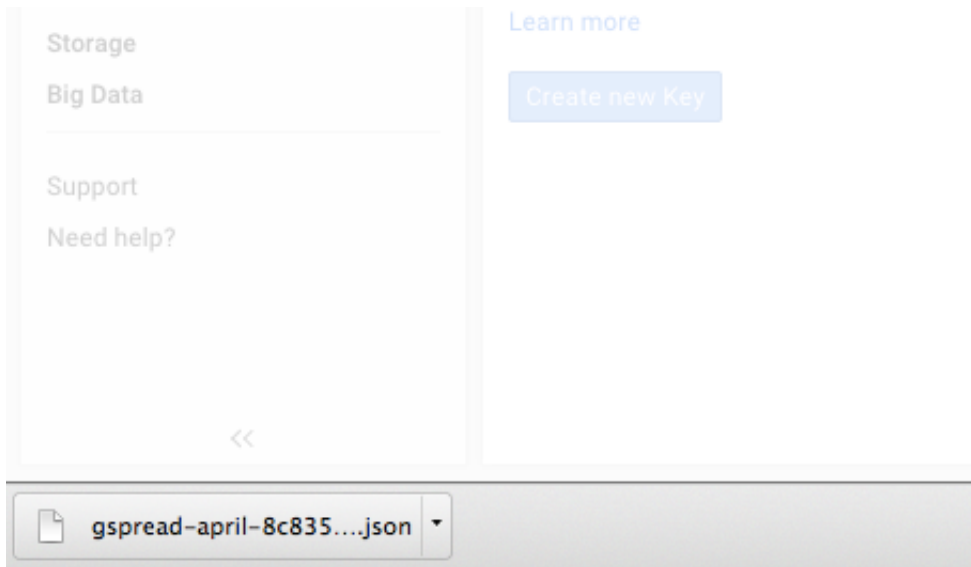
☐ Web application  
Accessed by web browsers over a network.

☒ **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).

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

You will automatically download a JSON file with this data.



This is how this file may look like:

```
{
  "private_key_id": "2cd ... ba4",
  "private_key": "-----BEGIN PRIVATE KEY-----\nNrDyLw ... jINQh/9\n-----END PRIVATE KEY-----\n",
  "client_email": "473 ... hd@developer.gserviceaccount.com",
  "client_id": "473 ... hd.apps.googleusercontent.com",
  "type": "service_account"
}
```

You'll need *client\_email* and *private\_key*.

5. Now you can read this file, and use the data when constructing your credentials:

```
import json
import gspread
from oauth2client.client import SignedJwtAssertionCredentials

json_key = json.load(open('gsread-april-2cd ... ba4.json'))
scope = ['https://spreadsheets.google.com/feeds']

credentials = SignedJwtAssertionCredentials(json_key['client_email'],
json_key['private_key'], scope)

gc = gspread.authorize(credentials)

wks = gc.open("Where is the money Lebowski?").sheet1
```

**Note:** Python 3 users need to cast `json_key['private_key']` to `bytes`. Otherwise you'll get `TypeError: expected bytes, not str` exception. Replace the line with `SignedJwtAssertionCredentials` call with this:

```
credentials = SignedJwtAssertionCredentials(json_key['client_email'],
bytes(json_key['private_key'], 'utf-8'), scope)
```

6. Go to Google Sheets and share your spreadsheet with an email you have in your `json_key['client_email']`. Otherwise you'll get a `SpreadsheetNotFound` exception when trying to open it.

## Troubleshooting

### `oauth2client.client.CryptoUnavailableError: No crypto library available`

If you're getting the "No crypto library available" exception, make sure you have `PyOpenSSL` library installed in your environment.

## Custom Credentials Objects

If you have another method of authenticating you can easily hack a custom credentials object.

```
class Credentials (object):
    def __init__ (self, access_token=None):
        self.access_token = access_token

    def refresh (self, http):
        # get new access_token
        # this only gets called if access_token is None
```