**Sending Email with OAuth2 and Gmail in Python**

**Part I – Google Cloud Project with OAuth**

Connecting to Gmail API requires creating a Google Cloud Platform project that enables the Gmail API and creates OAUTH credentials.

* 1. Go to <https://console.developers.google.com> you should be asked to sign in with a Google account. After login you should be presented with a screen similar to

this.

Graphical user interface, application

Description automatically generated

* 1. This account already has one project created so the text next to your dropdown arrow will be different. Follow the two steps in this screenshot.

Graphical user interface, application

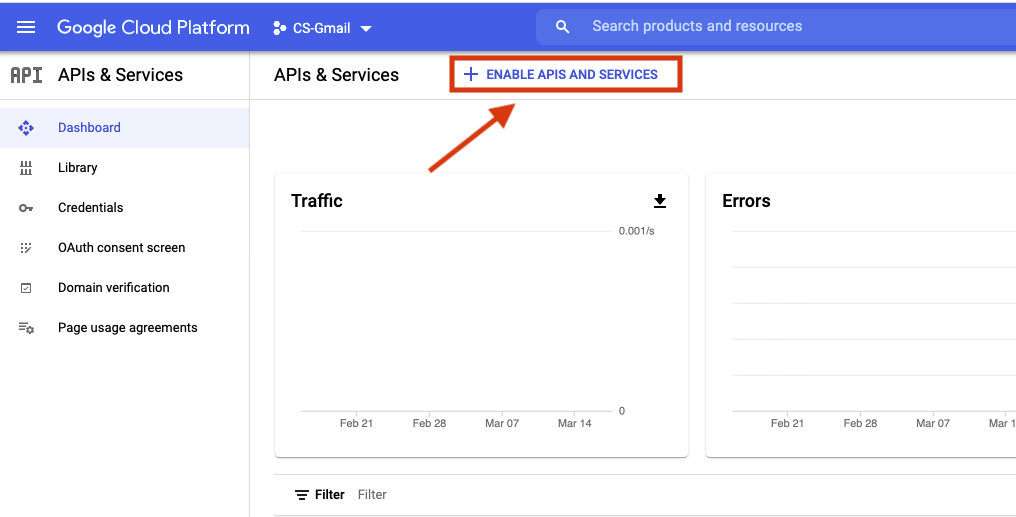
Description automatically generated

* 1. New Project will bring up this web page. Change the Project name to something that identifies what the project’s intended use. Since we will be testing the Gmail API I have chosen to use CS-Gmail. Click Create.

Graphical user interface, text, application

Description automatically generated

* 1. You should now see the name next to the dropdown arrow change to the name of your project and this dashboard page. (Note: If you have any existing projects you will need to use the dropdown arrow to switch the dashboard to the newly created project). Click on Enable API and Services.



* 1. This will open a webpage for Google’s API library. You may browse this page and notice how many API interfaces Google provides for their various services. Each API may work in a slightly different manner with respect to the level of provided. In this case we want the Gmail API so I have typed “Gmail” in the search dialog and my page narrowed down to 2 choices. Click on Gmail API.

Graphical user interface, application, Word, email

Description automatically generated

* 1. Select ENABLE

Graphical user interface, application

Description automatically generated

* 1. Once enabled this webpage is presented. This is where we will choose to create the OAUTH credentials needed for our python code to access the Gmail API. Click the CREATE CREDENTIALS button on the far right.

Graphical user interface, application

Description automatically generated

* 1. On the “Add credentials to your project” page chose client ID in step 1.

Graphical user interface, text, application, email

Description automatically generated

* 1. Select Configure Consent Screen – this allows configuring text on the web page that users will see when your app is asking for permission to access their Google account.

Graphical user interface, text, application, email

Description automatically generated

* 1. Select External as it is the only option available with a general Gmail account and click Create.

Graphical user interface, text, application

Description automatically generated

* 1. App information set the following 3 fields:
     + *Under App information* – set App Name to something recognizable to a user as they will see this in the consent screen. This example uses “CS Gmail API Test”
     + *Under App information* – set “User support email” to your Gmail account
     + *Under Developer Contact information* – set Email addresses to your Gmail account

Then click “Save and Continue”

Graphical user interface, application, email

Description automatically generated

* 1. Scope does not need to be added to this project, click Save and Continue. No changes are needed for Test Users so also click Save and Continue.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* 1. You will be presented with a summary screen. At this point you can click Credentials in the left navigation which should take you to this screen. Click Create Credentials and select OAuth client ID

Graphical user interface, text, application, email

Description automatically generated

* 1. For Application type select Desktop app. Changing the name for the client ID is optional. Click Create.

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* 1. Click Ok in the dialog that displays the created credentials.

Graphical user interface, text, application

Description automatically generated

* 1. Click the download icon on the credentials screen to download a json file with the credential information to your local machine.

Graphical user interface, text, application, email

Description automatically generated

* 1. At this point you can copy this .json file from your Downloads directory to the directory in which your python program resides. Usually, you name the file credentials.json.

When you run your python program for the first time the code will recognize that there is no authorization token and will use the credentials file to prompt the user to select a Google account and then to approve the permission scope for which the program is asking.

Here are a series of screenshots showing the process:

* Prompts user for access to a Google account.

Graphical user interface, text, application

Description automatically generated

* Notifying user that this app is not verified and asking for permission to continue.

Graphical user interface, text, application

Description automatically generated

* Tells user what permissions the app is asking to use. Click Allow to allow app access.

Graphical user interface, application

Description automatically generated

* Final confirmation screen.

Graphical user interface, text, application

Description automatically generated

* Last web page that indicates authentication flow is complete.

Text

Description automatically generated with low confidence

At this point your python program should have successfully sent the email to the desired email address. Looking in the directory of the program you will now see a token.json file that was not there before. If you were to run the program again it would send the message without this access authorization procedure until such time that the token will expire < *should look for token expiration time to place here>*

**Part II – Python Code**

Follow steps on this webpage to configure python to have the required libraries to connect to the Google API.

https://developers.google.com/gmail/api/quickstart/python

This is the sample code from the page edited to include code to create a message to be sent. This webpage <https://developers.google.com/gmail/api/guides/sending> includes information on how a message should be crafted for use with the API. This is a very simple example for demonstration purposes.

from \_\_future\_\_ import print\_function

import os.path

from googleapiclient.discovery import build

from google\_auth\_oauthlib.flow import InstalledAppFlow

from google.auth.transport.requests import Request

from google.oauth2.credentials import Credentials

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import base64

# If modifying these scopes, delete the file token.json.

SCOPES = ['https://www.googleapis.com/auth/gmail.modify']

def main():

"""Shows basic usage of the Gmail API.

Lists the user's Gmail labels.

"""

creds = None

# The file token.json stores the user's access and refresh tokens, and is

# created automatically when the authorization flow completes for the first

# time.

if os.path.exists('/Users/lweihl/Documents/python\_projects/gmail\_oauth/token.json'):

creds = Credentials.from\_authorized\_user\_file('token.json', SCOPES)

# If there are no (valid) credentials available, let the user log in.

if not creds or not creds.valid:

if creds and creds.expired and creds.refresh\_token:

creds.refresh(Request())

else:

flow = InstalledAppFlow.from\_client\_secrets\_file('/Users/lweihl/Documents/python\_projects/gmail\_oauth/credentials.json', SCOPES)

creds = flow.run\_local\_server(port=0)

# Save the credentials for the next run

with open('/Users/lweihl/Documents/python\_projects/gmail\_oauth/token.json', 'w') as token:

token.write(creds.to\_json())

service = build('gmail', 'v1', credentials=creds)

# Call the Gmail API

emailMsg = 'Testing Gmail OAUTH - version 2 of the code'

mimeMessage = MIMEMultipart()

# <email addr here> should be replaced with ‘user@somewhere’

mimeMessage['to'] = <email addr here>

mimeMessage['subject'] = 'Successful Gmail OAUTH Test'

mimeMessage.attach(MIMEText(emailMsg, 'plain'))

raw\_string = base64.urlsafe\_b64encode(mimeMessage.as\_bytes()).decode()

message = service.users().messages().send(userId='me', body={'raw': raw\_string}).execute()

print(message)

if \_\_name\_\_ == '\_\_main\_\_':

main()