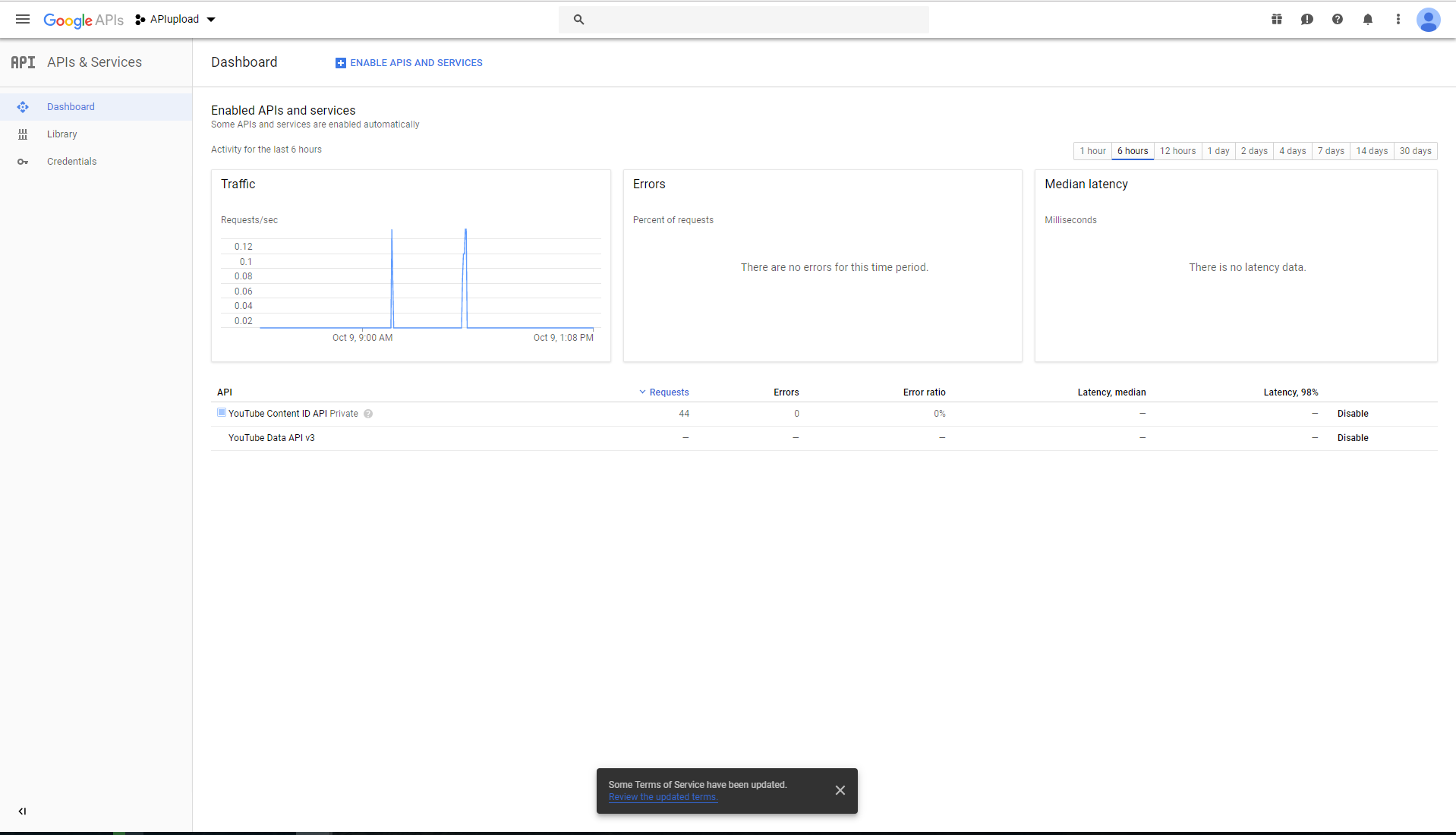
# YouTube Data API Overview

## Introduction

This document is intended for developers who want to write applications that interact with YouTube. It explains basic concepts of YouTube and of the API itself. It also provides an overview of the different functions that the API supports.

1. You need a [Google Account](https://www.google.com/accounts/NewAccount) to access the Google API Console, request an API key, and register your application.
2. Create a project in the [Google Developers Console](https://console.developers.google.com/)and [obtain authorization credentials](https://developers.google.com/youtube/registering_an_application) so your application can submit API requests.



1. After creating your project, make sure the YouTube Data API is one of the services that your application is registered to use:
   1. Go to the [API Console](https://console.developers.google.com/) and select the project that you just registered.
   2. Visit the [Enabled APIs page](https://console.developers.google.com/apis/enabled). In the list of APIs, make sure the status is **ON** for the **YouTube Data API v3**.
2. If your application will use any API methods that require user authorization, read the [authentication](https://developers.google.com/youtube/v3/guides/authentication) guide to learn how to implement OAuth 2.0 authorization.
3. Select a [client library](https://developers.google.com/youtube/v3/libraries) to simplify your API implementation.
4. Familiarize yourself with the core concepts of the JSON (JavaScript Object Notation) data format. JSON is a common, language-independent data format that provides a simple text representation of arbitrary data structures. For more information, see [json.org](http://json.org/).

## Upload a video

This sample calls the API's videos.insert method to upload a video to the channel associated with the request.

#!/usr/bin/python

import httplib

import httplib2

import os

import random

import sys

import time

from apiclient.discovery import build

from apiclient.errors import HttpError

from apiclient.http import MediaFileUpload

from oauth2client.client import flow\_from\_clientsecrets

from oauth2client.file import Storage

from oauth2client.tools import argparser, run\_flow

# Explicitly tell the underlying HTTP transport library not to retry, since

# we are handling retry logic ourselves.

httplib2.RETRIES = 1

# Maximum number of times to retry before giving up.

MAX\_RETRIES = 10

# Always retry when these exceptions are raised.

RETRIABLE\_EXCEPTIONS = (httplib2.HttpLib2Error, IOError, httplib.NotConnected,

httplib.IncompleteRead, httplib.ImproperConnectionState,

httplib.CannotSendRequest, httplib.CannotSendHeader,

httplib.ResponseNotReady, httplib.BadStatusLine)

# Always retry when an apiclient.errors.HttpError with one of these status

# codes is raised.

RETRIABLE\_STATUS\_CODES = [500, 502, 503, 504]

# The CLIENT\_SECRETS\_FILE variable specifies the name of a file that contains

# the OAuth 2.0 information for this application, including its client\_id and

# client\_secret. You can acquire an OAuth 2.0 client ID and client secret from

# the {{ Google Cloud Console }} at

# {{ https://cloud.google.com/console }}.

# Please ensure that you have enabled the YouTube Data API for your project.

# For more information about using OAuth2 to access the YouTube Data API, see:

# https://developers.google.com/youtube/v3/guides/authentication

# For more information about the client\_secrets.json file format, see:

# https://developers.google.com/api-client-library/python/guide/aaa\_client\_secrets

CLIENT\_SECRETS\_FILE = "client\_secrets.json"

# This OAuth 2.0 access scope allows an application to upload files to the

# authenticated user's YouTube channel, but doesn't allow other types of access.

YOUTUBE\_UPLOAD\_SCOPE = "https://www.googleapis.com/auth/youtube.upload"

YOUTUBE\_API\_SERVICE\_NAME = "youtube"

YOUTUBE\_API\_VERSION = "v3"

# This variable defines a message to display if the CLIENT\_SECRETS\_FILE is

# missing.

MISSING\_CLIENT\_SECRETS\_MESSAGE = """

WARNING: Please configure OAuth 2.0

To make this sample run you will need to populate the client\_secrets.json file

found at:

%s

with information from the {{ Cloud Console }}

{{ https://cloud.google.com/console }}

For more information about the client\_secrets.json file format, please visit:

https://developers.google.com/api-client-library/python/guide/aaa\_client\_secrets

""" % os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_),

CLIENT\_SECRETS\_FILE))

VALID\_PRIVACY\_STATUSES = ("public", "private", "unlisted")

def get\_authenticated\_service(args):

flow = flow\_from\_clientsecrets(CLIENT\_SECRETS\_FILE,

scope=YOUTUBE\_UPLOAD\_SCOPE,

message=MISSING\_CLIENT\_SECRETS\_MESSAGE)

storage = Storage("%s-oauth2.json" % sys.argv[0])

credentials = storage.get()

if credentials is None or credentials.invalid:

credentials = run\_flow(flow, storage, args)

return build(YOUTUBE\_API\_SERVICE\_NAME, YOUTUBE\_API\_VERSION,

http=credentials.authorize(httplib2.Http()))

def initialize\_upload(youtube, options):

tags = None

if options.keywords:

tags = options.keywords.split(",")

body=dict(

snippet=dict(

title=options.title,

description=options.description,

tags=tags,

categoryId=options.category

),

status=dict(

privacyStatus=options.privacyStatus

)

)

# Call the API's videos.insert method to create and upload the video.

insert\_request = youtube.videos().insert(

part=",".join(body.keys()),

body=body,

# The chunksize parameter specifies the size of each chunk of data, in

# bytes, that will be uploaded at a time. Set a higher value for

# reliable connections as fewer chunks lead to faster uploads. Set a lower

# value for better recovery on less reliable connections.

#

# Setting "chunksize" equal to -1 in the code below means that the entire

# file will be uploaded in a single HTTP request. (If the upload fails,

# it will still be retried where it left off.) This is usually a best

# practice, but if you're using Python older than 2.6 or if you're

# running on App Engine, you should set the chunksize to something like

# 1024 \* 1024 (1 megabyte).

media\_body=MediaFileUpload(options.file, chunksize=-1, resumable=True)

)

resumable\_upload(insert\_request)

# This method implements an exponential backoff strategy to resume a

# failed upload.

def resumable\_upload(insert\_request):

response = None

error = None

retry = 0

while response is None:

try:

print "Uploading file..."

status, response = insert\_request.next\_chunk()

if response is not None:

if 'id' in response:

print "Video id '%s' was successfully uploaded." % response['id']

else:

exit("The upload failed with an unexpected response: %s" % response)

except HttpError, e:

if e.resp.status in RETRIABLE\_STATUS\_CODES:

error = "A retriable HTTP error %d occurred:\n%s" % (e.resp.status,

e.content)

else:

raise

except RETRIABLE\_EXCEPTIONS, e:

error = "A retriable error occurred: %s" % e

if error is not None:

print error

retry += 1

if retry > MAX\_RETRIES:

exit("No longer attempting to retry.")

max\_sleep = 2 \*\* retry

sleep\_seconds = random.random() \* max\_sleep

print "Sleeping %f seconds and then retrying..." % sleep\_seconds

time.sleep(sleep\_seconds)

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

argparser.add\_argument("--file", required=True, help="Video file to upload")

argparser.add\_argument("--title", help="Video title", default="Test Title")

argparser.add\_argument("--description", help="Video description",

default="Test Description")

argparser.add\_argument("--category", default="22",

help="Numeric video category. " +

"See https://developers.google.com/youtube/v3/docs/videoCategories/list")

argparser.add\_argument("--keywords", help="Video keywords, comma separated",

default="")

argparser.add\_argument("--privacyStatus", choices=VALID\_PRIVACY\_STATUSES,

default=VALID\_PRIVACY\_STATUSES[0], help="Video privacy status.")

args = argparser.parse\_args()

if not os.path.exists(args.file):

exit("Please specify a valid file using the --file= parameter.")

youtube = get\_authenticated\_service(args)

try:

initialize\_upload(youtube, args)

except HttpError, e:

print "An HTTP error %d occurred:\n%s" % (e.resp.status, e.content)