
Google™
10

YouTube API Uploads: Tools, Tips, and Best Practices

Gareth McSorley, Jeffrey Posnick, and Kuan Yong

May 20, 2010

View live notes and ask questions about this session on Google Wave:

http://bit.ly/9fCla2

Google I/O 10

Session Overview

- Basic info for all uploaders
- Uploading from a web app (browser-based upload)
- Uploading from a client app (direct upload)
- Resumable uploads
- Handling metadata for uploaded videos
- Mobile uploads
- YouTube Direct

Session Overview

- **Basic info for all uploaders**
- Uploading from a web app (browser-based upload)
- Uploading from a client app (direct upload)
- Resumable uploads
- Handling metadata for uploaded videos
- Mobile uploads
- YouTube Direct

YouTube Needs You!

- Think of us for all your video hosting needs.
- We have APIs to let you upload videos to YouTube from just about anywhere.
- API uploads are nearly 15% of all YouTube uploads!

Always let users upload videos to their own YouTube account

- YouTube accounts have soft cap of **2000 videos**.
  - 50 additional videos/day after the first 2000.
  - Don't solicit user uploads into your account!

Basic Info for All Uploaders

- All upload API calls must contain a Developer Key.
- Videos up to 10 minutes long.
- Upload in original resolution and format.
- Upload requests are rate limited.
  - Don't upload 2000 videos all at once!
  - If you hit rate limits (too_many_recent_calls HTTP 40x response), wait 10 minutes.

Session Overview

- Basic info for all uploaders
- **Uploading from a web app (browser-based upload)**
-
- Uploading from a client app (direct upload)
- Resumable uploads
- Handling metadata for uploaded videos
- Mobile uploads
- YouTube Direct

Browser-Based Uploads

- Submit metadata, get back ***URL*** and ***TOKEN***:

```
POST http://gdata.youtube.com/action/GetUploadToken
```

- Create HTML form:

```
<form action="URL?nexturl=http://www.example.com">
  <input id="file" type="file" name="file"/>
  <input type="hidden" name="token" value="TOKEN"/>
  <input type="submit" value="Upload" />
</form>
```

- Browser submits directly to YouTube.

- No excuse not to use AuthSub or OAuth.

Browser-Based Uploads

ActionScript 3 Example

Source: http://gdata-samples.googlecode.com/svn/trunk/gdata/YouTubeApi/YouTubeApi.mxml

Live Demo: http://gdata-samples.googlecode.com/svn/trunk/gdata/YouTubeApi/YouTubeApi.swf

Session Overview

- Basic info for all uploaders
- Uploading from a web app (browser-based upload)
- **Uploading from a client app (direct upload)**
- Resumable uploads
- Handling metadata for uploaded videos
- Mobile uploads
- YouTube Direct

Direct Uploads

- For installed apps, when you have direct access to video file.
  - But... check out resumable uploads.
- HTTP POST of multipart/related MIME message, containing metadata and binary video data.
- If your code is used by third parties, make sure you're uploading into the *user's* account.
  - Temptation is to use ClientLogin, but OAuth or AuthSub are preferable.

Session Overview

- Basic info for all uploaders
- Uploading from a web app (browser-based upload)
- Uploading from a client app (direct upload)
- **Resumable uploads**
- Handling metadata for uploaded videos
- Mobile uploads
- YouTube Direct

Resumable Uploads

- Similar in use case and best practices to direct uploads.
- Two (or more) part process:
  - HTTP POST of video metadata. Response contains unique upload URL.
  - One of more HTTP PUTs to upload video data or query for status of interrupted upload.
- Should be used instead of direct uploads anywhere client library support is available.
- Perfect for low-bandwidth devices or large videos.

Resumable Uploads

Python Example

Source: http://gdata-samples.googlecode.com/svn/trunk/gdata/resumable-uploads/yt_resumable_upload.py

Resumable Upload Example (1)

POST /resumable/feeds/api/users/menoexist/uploads
Authorization: GoogleLogin auth=<auth token>
X-GData-Key key=<dev key>
Content-Type: application/atom+xml
Content-length: 276
Host: uploads.gdata.youtube.com
Slug: my_file.mpg

<?xml version="1.0"?>
<entry ...>
  Atom video entry goes here.
</entry>

Resumable Upload Example (2)

HTTP/1.1 200 OK
Content-Length: 0
Expires: Fri, 01 Jan 1990 00:00:00 GMT
Server: Upload Server Built on ...
Location: http://uploads.gdata.youtube.com/resumableupload/AEnB2Vr...4ftyz69ZqK8Hg
Pragma: no-cache
Cache-Control: no-cache, no-store, must-revalidate
Date: Tue, 20 Apr 2010 17:29:55 GMT
Content-Type: text/html

Resumable Upload Example (3)

```
PUT /resumableupload/AEnB2Vr...4ftyz69ZqK8Hg
Host: uploads.gdata.youtube.com
Content-length: 2048
Content-Type: video/mpeg
X-User-Ip: 127.0.0.1

[Video File]
```

Resumable Upload Example (4)

201 Created
content-length: 3120
content-location: http://gdata.youtube.
com/feedsa/api/users/menoexist/uploads/fjHde15ksgU
content-type: application/atom+xml; charset=UTF-8
location: http://gdata.youtube.
com/feeds/api/users/menoexist/uploads/fjHde15ksgU

<?xml version="1.0"?>
<entry ...>
  Atom video entry will be here.
</entry>

Resumable Upload Example (5)

```
PUT /resumableupload/AEnB2Vr...4ftyz69ZqK8Hg
Host: uploads.gdata.youtube.com
Content-length: 0
Content-Range: bytes */*
```

Resumable Upload Example (6)

HTTP/1.1 308 Resume Incomplete
Host: uploads.gdata.youtube.com
Expires: Fri, 01 Jan 1990 00:00:00 GMT
Server: Upload Server Built on ...
Content-length: 0
Range: bytes=0-1023
Pragma: no-cache
Cache-Control: no-cache, no-store, must-revalidate
Date: Tue, 20 Apr 2010 17:29:55 GMT
Content-Type: text/html

Resumable Upload Example (7)

```
PUT /resumableupload/AEnB2Vr...4ftyz69ZqK8Hg
Host: uploads.gdata.youtube.com
Content-length: 1024
Content-Range: bytes 1024-2047/2048
Content-Type: video/mpeg
X-User-Ip: 127.0.0.1

[Video File from byte 1024-end]
```

Session Overview

- Basic info for all uploaders
- Uploading from a web app (browser-based upload)
- Uploading from a client app (direct upload)
- Resumable uploads
- **Handling metadata for uploaded videos**
- Mobile uploads
- YouTube Direct

Incomplete Video Metadata

- Until recently, it was necessary to submit a title, description, and category at upload time.
- New <yt:incomplete> tag can be used to defer metadata creation.
  - Video will remain in a "draft" state until metadata is supplied.
  - Perfect for mobile uploads!
- Resumable uploads even have a "no metadata" option

Incomplete Video Metadata Examples (1)

```
<?xml version="1.0"?>
  <entry xmlns='http://www.w3.org/2005/Atom' xmlns:yt='http:
//gdata.youtube.com/schemas/2007' xmlns:app='http://www.w3.
org/2007/app'>
  <app:control>
    <yt:incomplete/>
  </app:control>
</entry>
```

Incomplete Video Metadata Examples (2)

POST /resumable/feeds/api/users/menoexist/uploads
Authorization: GoogleLogin auth=<auth token>
X-GData-Key key=<dev key>
Content-length: 0
Host: uploads.gdata.youtube.com
Slug: something_meaningful.mpg

Developer Tags

- Used to group videos.
- Tied to a particular developer key.
- Way better than uploading to a single account.
- On Upload:

```
<media:category scheme=
"http://gdata.youtube.com/schemas/2007/
developertags.cat">
  TagName
</media:category>
```

- To Fetch:

```
http://gdata.youtube.com/feeds/api/videos/-/
    %7Bhttp%3A%2F%2Fgdata.youtube.com
    %2Fschemas%2F2007%2Fdevelopertags.cat
    %7DTagName
```

Session Overview

- Basic info for all uploaders
- Uploading from a web app (browser-based upload)
- Uploading from a client app (direct upload)
- Resumable uploads
- Handling metadata for uploaded videos
- **Mobile uploads**
- YouTube Direct

Mobile Uploads
Android OS

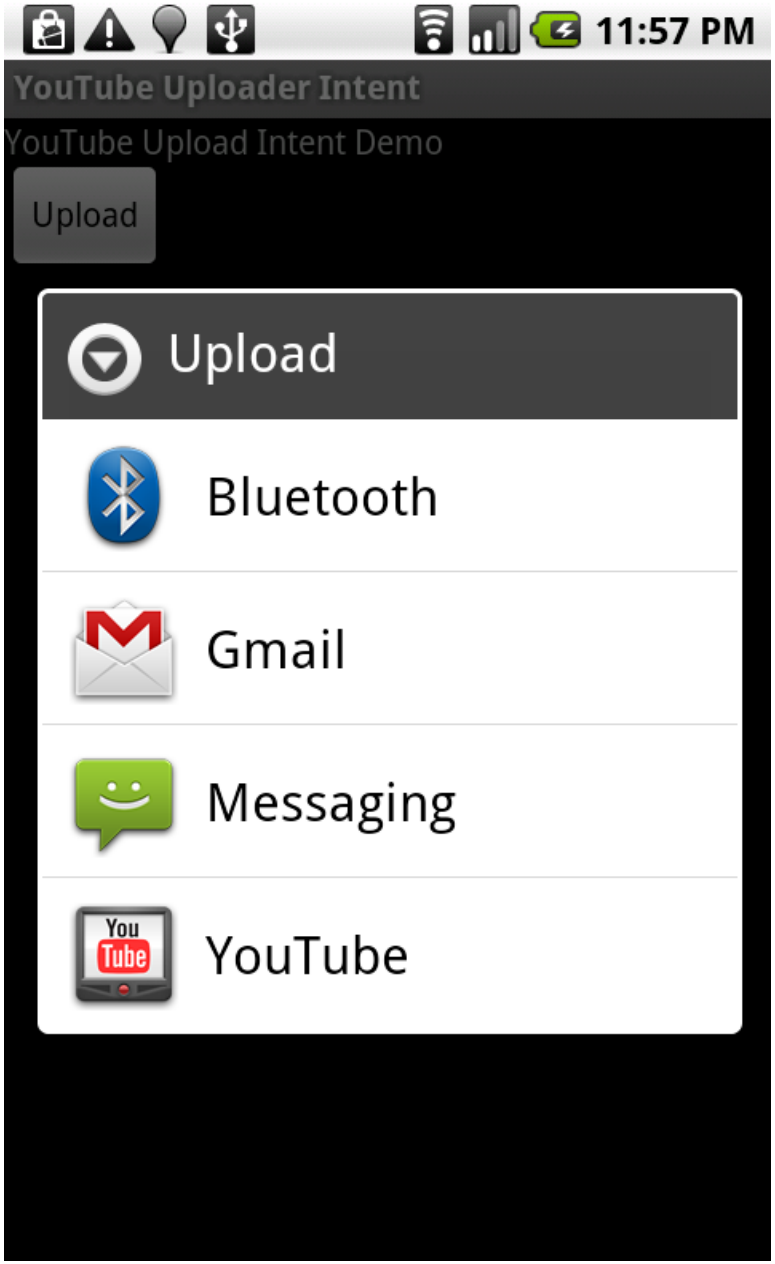
- For most cases, use the ACTION_SEND Intent.

```
public void onUploadClick(View v) {
 Intent uploadIntent = new Intent(Intent.ACTION_SEND);
 // In a real app, video would be captured from camera.
 File f = new File("/sdcard/test.mov");
 uploadIntent.setDataAndType(Uri.fromFile(f), "video/quicktime");
 startActivity(Intent.createChooser(uploadIntent, "Upload"));
}
```

- Video will be uploaded into the YouTube account associated with Android device.
- If you need more control, use the newest Google Data Java client library or raw HTTP communication.

Mobile Uploads

Android OS



Mobile Uploads

iPhone OS

- Use UIImagePickerController (iPhone 3GS only).
- Copy video file to app's Documents directory and delete after upload is successful.
  - Won't lose video file if app crashes/quits during upload.
- Invoking video capture controller will most likely result in viewDidUnload and didReceiveMemoryWarning calls.
  - Make sure your app handles low memory situations properly!

Mobile Uploads

iPhone OS

```
UIImagePickerController *imagePicker = [[[UIImagePickerController alloc] init] autorelease];
imagePicker.delegate = self;
imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;
imagePicker.mediaTypes = [NSArray arrayWithObject:(NSString *)kUTTypeMovie];
[self presentModalViewController:imagePicker animated:YES];

...

- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {
  NSURL *mediaUrl = [info valueForKey:UIImagePickerControllerMediaURL];
  NSString *fromPath = [mediaUrl path];
  NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
  NSString *documentsDirectory = [paths objectAtIndex:0];
  NSString *toPath = [documentsDirectory stringByAppendingPathComponent:kTempVideoFile];
  NSFileManager *fileManager = [NSFileManager defaultManager];
  [fileManager copyItemAtPath:fromPath toPath:toPath error:NULL];
...
}
```

Mobile Uploads

iPhone OS

- Check out ASIHtttpRequest for making HTTP requests
  - http://allseeing-i.com/ASIHTTPRequest/
  - Easy to do progress bars, form POSTs, etc.

- YouTube Direct mobile app source code:
  - http://code.google.com/p/ytd-iphone/
  - Reader exercise: Update code to use resumable upload API.

- **Side tip**: Check out our blog post on how to play YouTube videos in iPhone/iPad apps.□□
  - <embed> in UIWebView
  - http://apiblog.youtube.com/2009/02/youtube-apis-iphone-cool-mobile-apps.html

Session Overview

- Basic info for all uploaders
- Uploading from a web app (browser-based upload)
- Uploading from a client app (direct upload)
- Resumable uploads
- Handling metadata for uploaded videos
- Mobile uploads
- **YouTube Direct**

YouTube Direct

Overview

YouTube Direct (YTD) is an open source video submission platform that is built on top of the YouTube API and Google App Engine.

YTD has two components:

- Embeddable video uploader <iframe>.
- Admin-only moderation control panel.

Google Code Project:
http://code.google.com/p/youtube-direct/

YouTube Direct

Overview

Download the code and deploy to your own App Engine instance.

Demo at:
http://ytd-demo.appspot.com/test.html

YouTube Direct

Upload Interface

la in fringilla. Cum
sent vulputate nibh ut
eu diam. Aliquam vel
litora torquent per
c ipsum metus, sit
as consequat lectus
urus. Maecenas
sodales. Sed porta
ue vel, vehicula eget
ollicitudin neque nec

powered by YouTube

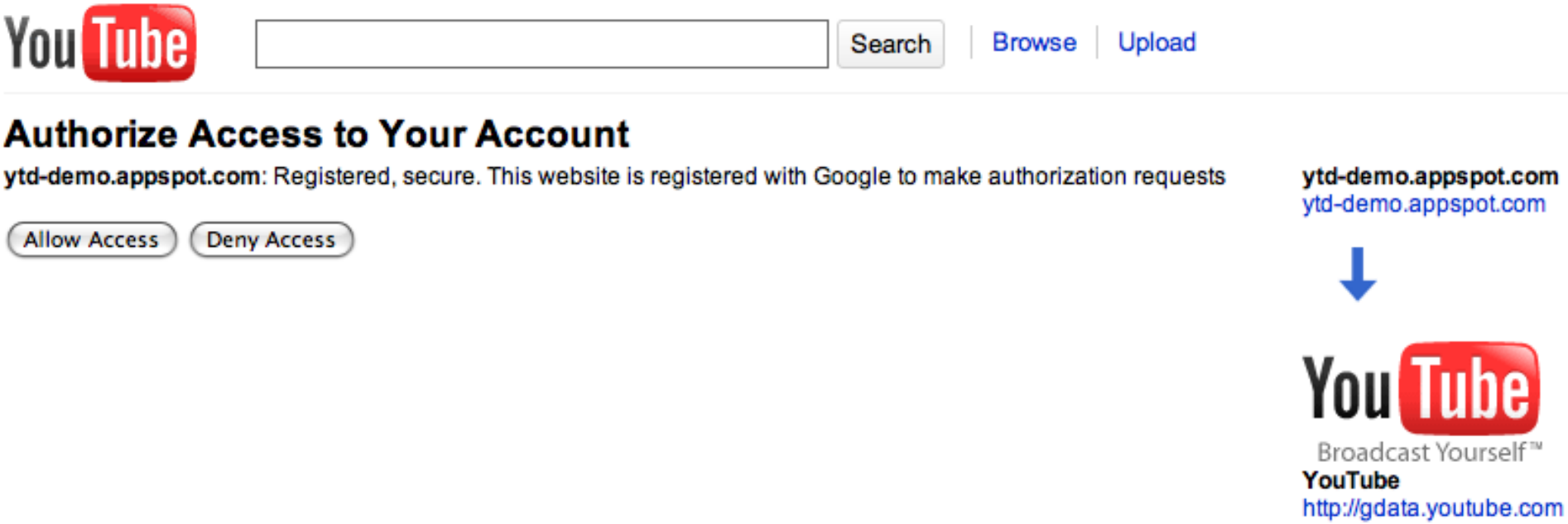
Welcome to the YouTube Direct demo!

Login to YouTube

Google I/O 10

YouTube Direct

Upload Interface



Google I/O 10

YouTube Direct

Upload Interface

la in fringilla. Cum
sent vulputate nibh ut
eu diam. Aliquam vel
litora torquent per
c ipsum metus, sit
ras consequat lectus
urus. Maecenas
sodales. Sed porta
ue vel, vehicula eget
ollicitudin neque nec

jeffposnicktest [logout]

powered by YouTube

Video Title: (required)

Video Description: (required)

Tags: (required) (use "," to separate)

Date:

Location:

Phone Number:

Email me on approval:

Select file: (required) Choose File No file chosen

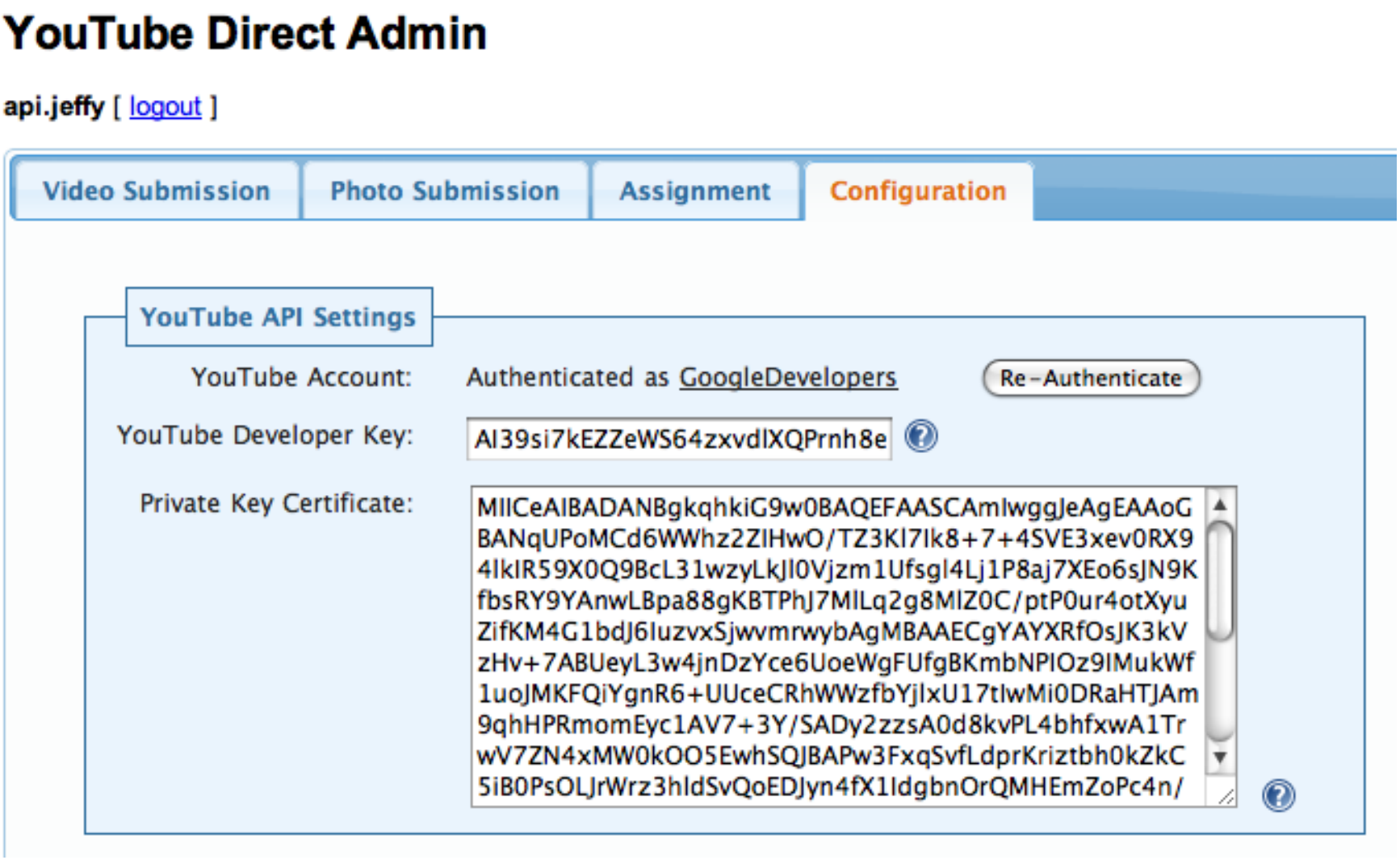
Upload Cancel

By clicking 'Upload,' you certify that you own all rights to the content or that you are authorized by the owner to make the content publicly available on YouTube, and that it otherwise complies with the YouTube Terms of Service located at http://www.youtube.com/t/terms.

Google 10 I/O

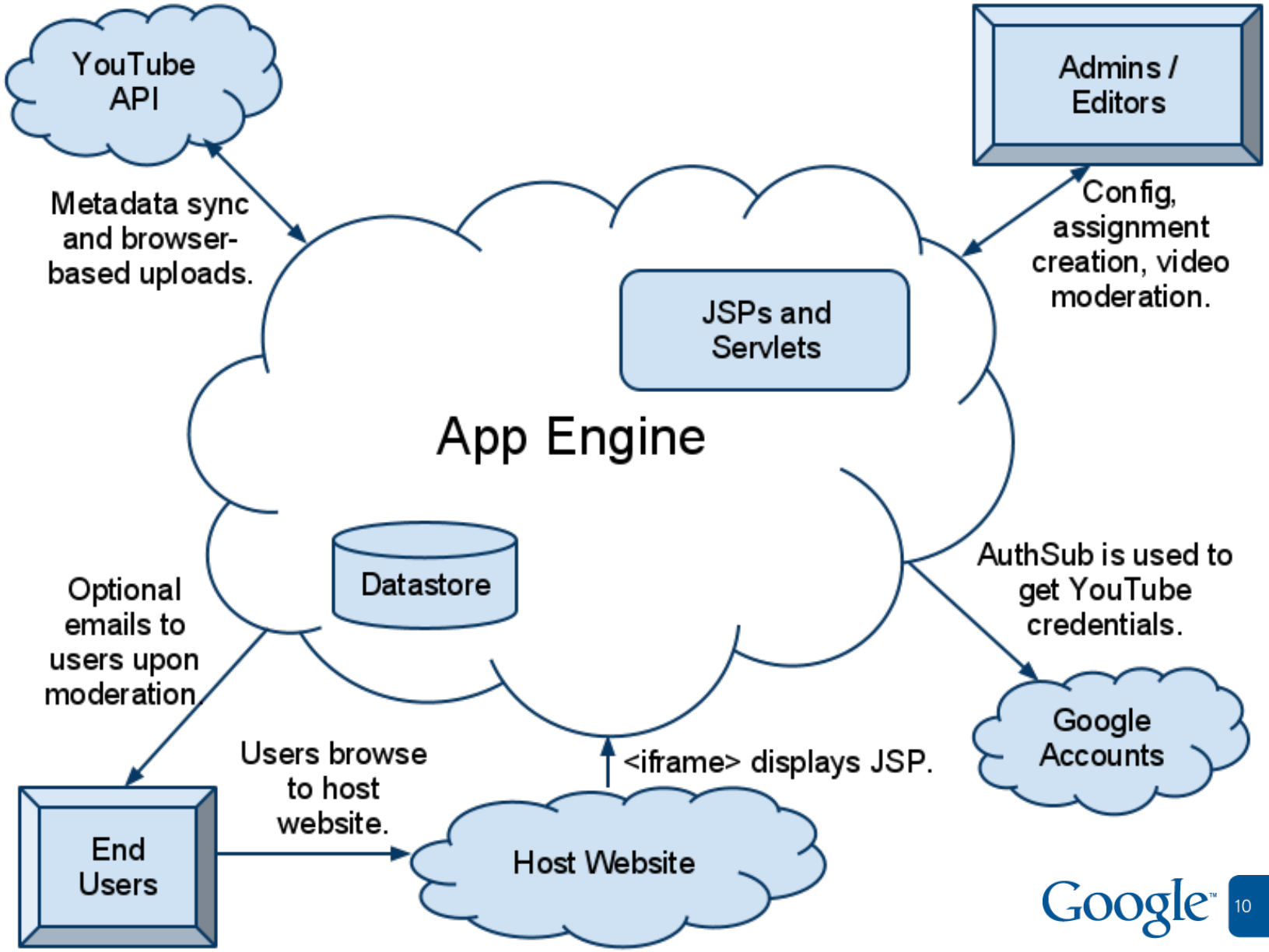
YouTube Direct

Admin Interface



YouTube Direct

Architecture



Questions? Answers!

View live notes and ask questions about this session on Google Wave:

http://bit.ly/9fCla2

Google I/O 10


Google™
10