

Users like to keep their data in one place on the web where it's easily accessible. Whether it's YouTube videos, Google Drive files, Google contacts or one of many other types of data, users need a way to securely grant applications access to their data. OAuth is the key web standard for delegated data access and OAuth 2.0 is the next-generation version with additional security features. This session will cover the latest advances in how OAuth can be used for data access, but will also dive into how you can lower the barrier to entry for your application by allowing users to login using their Google accounts. You will learn, through an example written in Python, how to use OAuth 2.0 to incorporate user identity into your web application. Best practices for desktop applications, mobile applications and server-to-server use cases will also be discussed



About Ryan Boyd

[View full profile](#)

Ryan is a Developer Advocate at Google, focused on cloud data services. He's been at Google for 5 years and previously helped build out the Google Apps ISV ecosystem. He published his first book "Getting Started with OAuth 2.0" with O'Reilly.



OAuth 2.0

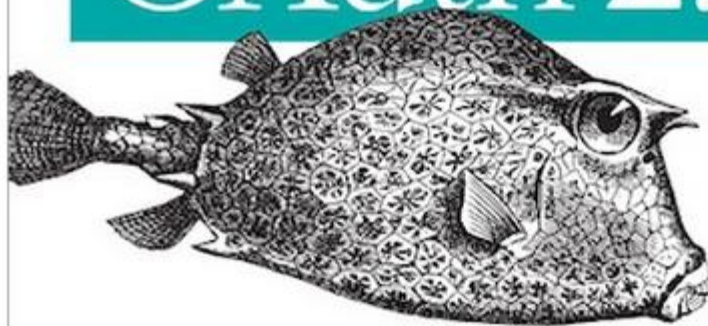
Identity and Data Access

Ryan Boyd
Developer Advocate, Google

*Programming Clients for Secure Web API
Authorization and Authentication*

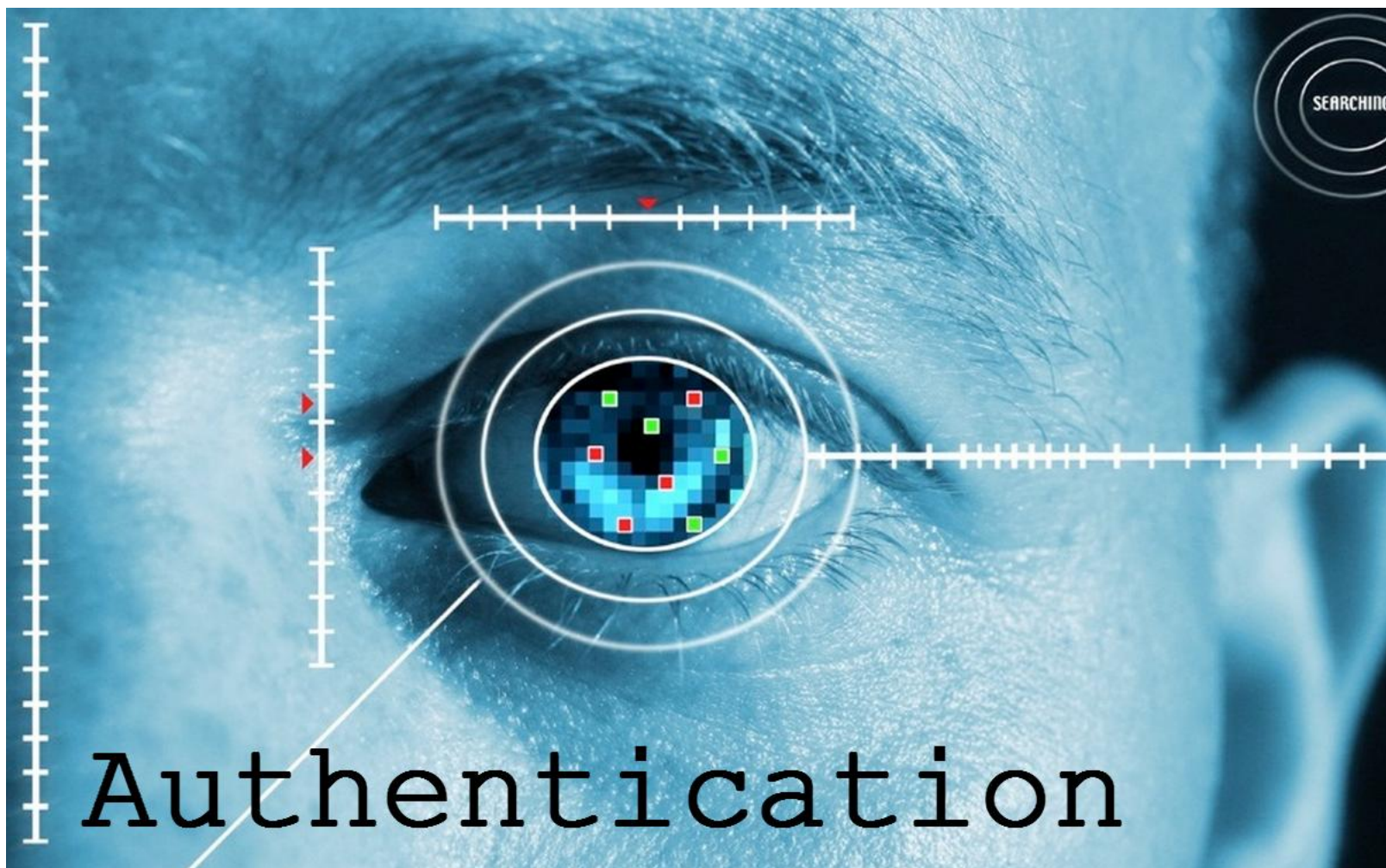
Getting Started with

OAuth 2.0



O'REILLY®

Ryan Boyd



Authentication

Authorization



Have you ever shared
your password with an
app so it could use your
data?



Yes :-)

Your opportunity

- 1) Eliminate the need for users to reveal their password to your app
- 2) Restrict the level of data available to your app
- 3) Allow users to revoke access to their data

OAuth 2.0 for Authorization

Do you use the same
username & password for
multiple web sites?



Yes :-)

How many keystrokes do
you type to sign up for a
new account?



50+!

Your opportunity

- 1) Minimize how many passwords users need
- 2) Discourage password re-use
- 3) Optimize sign-up flows to onboard users faster

OAuth 2.0 for Login (OpenID Connect)

Agenda

- Terminology
- Authorization
 - End-goal
 - Client-side in JavaScript
 - Server-side in Python
 - Mobile Apps
 - App-based in Python
- Authentication
- Resources



OAuth 2.0 for Authorization

35+ APIs!

Goal





Getting Started

APIs Console

developers.google.com/console



Demo Project ▼

Overview

Services

Team

API Access

API Access

To prevent abuse, Google places limits on API requests. Using a valid OAuth token or API key allows you to exceed anonymous limits by connecting requests back to your project.

Authorized API Access

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private. A single project may contain up to 7 client IDs. [Learn more](#)

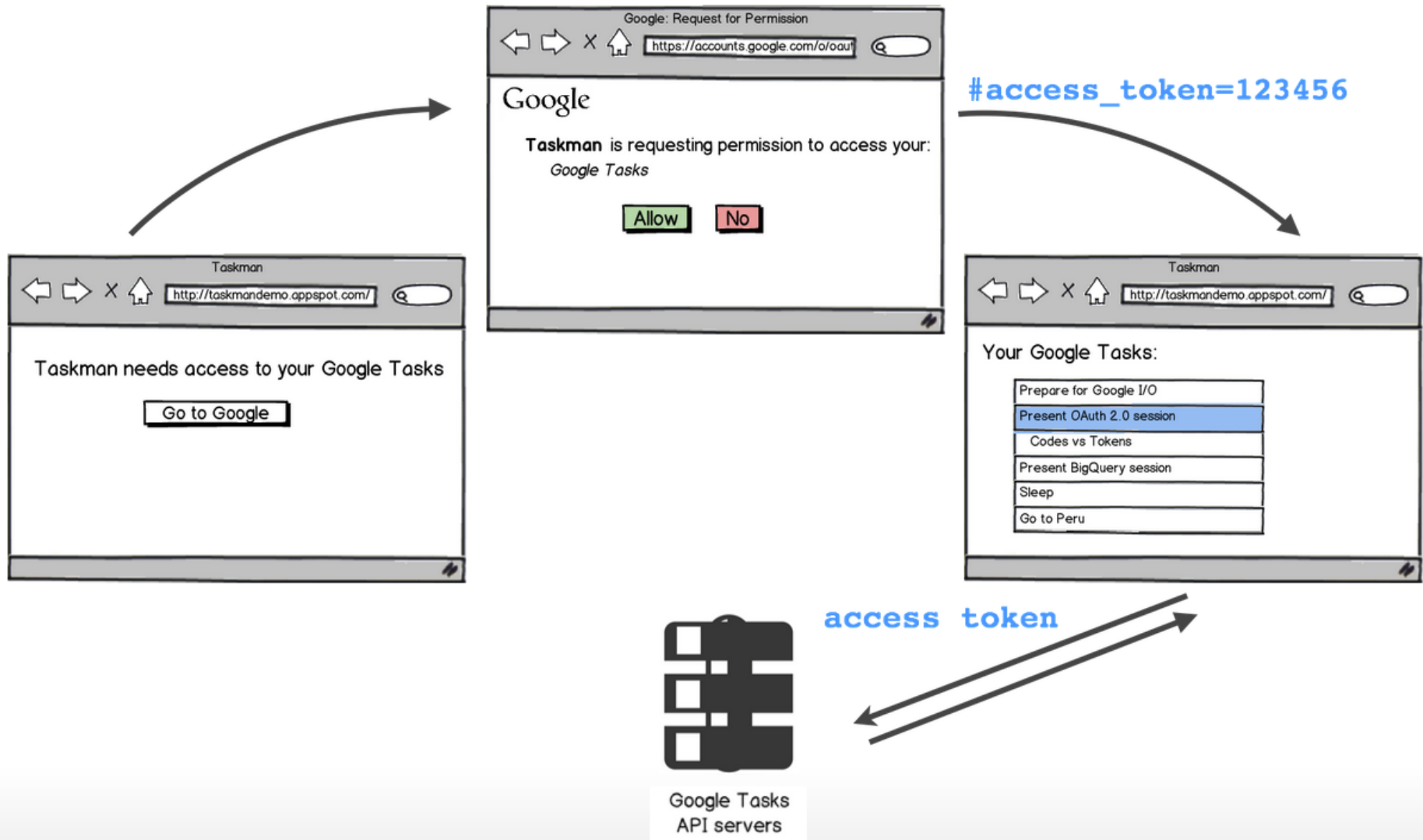


Create an OAuth 2.0 client ID...



Pure JavaScript Flow

Developing Client-side Applications



Client-side Flow in JavaScript

Google APIs Client Library for JavaScript

Let's see it in action!

Client-side Flow in JavaScript

Google APIs Client Library for JavaScript

JAVASCRIPT

```
gapi.auth.authorize({  
  client_id: '387636757294.apps.googleusercontent.com',  
  scope: 'https://www.googleapis.com/auth/tasks'},  
  handleAuthResultPopup);  
  
function handleAuthResultPopup() {  
  alert(gapi.auth.getToken());  
}
```

Client-side Flow in JavaScript

Step 1 - App directs the user to Google for Authorization

JAVASCRIPT

```
<script type="text/javascript">
  var clientId = '387636757294.apps.googleusercontent.com';
  var authorizationUrlBase = 'https://accounts.google.com/o/oauth2/auth';
  var redirectUri = 'https://taskmandemo.appspot.com/oauth2callback.html';
  var scope = 'https://www.googleapis.com/auth/tasks';

  function startOauth() {
    var url = authorizationUrlBase;
    url += '?response_type=token'
      + '&redirect_uri=' + encodeURIComponent(redirectUri)
      + '&client_id=' + encodeURIComponent(clientId)
      + '&scope=' + encodeURIComponent(scope);
    var w = window.open(url, 'oauth', 'width=500,height=400');
  }
</script>
```

Client-side Flow in JavaScript

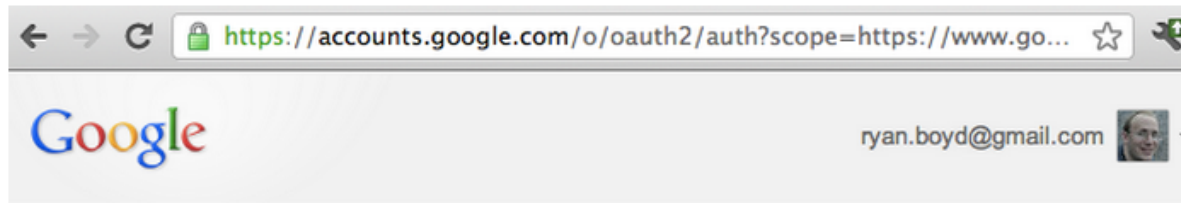
Step 1 - App directs the user to Google for Authorization

```
https://accounts.google.com/o/oauth2/auth?  
  client_id=387636757294.apps.googleusercontent.com&  
  scope=https://www.googleapis.com/auth/tasks&  
  redirect_uri=https://taskmandemo.appspot.com/oauth2callback.html&  
  response_type=token
```

URL

Client-side Flow in JavaScript

Step 2a - User authorizes access



Taskman is requesting permission to:

▸ Manage your tasks

Allow access

No thanks

Taskman

[Learn more](#)

Client-side Flow in JavaScript

Step 2b - User is redirected back to the app

```
https://taskmandemo.appspot.com/oauth2callback.html#  
  access_token=ya29.AHES6ZT8XLGWjlYfP1KvKLQVvYj81C6uA_bUsaZBKWB4ZE&  
  token_type=Bearer&  
  expires_in=3600
```

URL

GOAL!

Client-side Flow in JavaScript

Questions!

- How do I get the token back to the app from the popup window?
- How do I get another access token?
- What happens if the user isn't logged in?

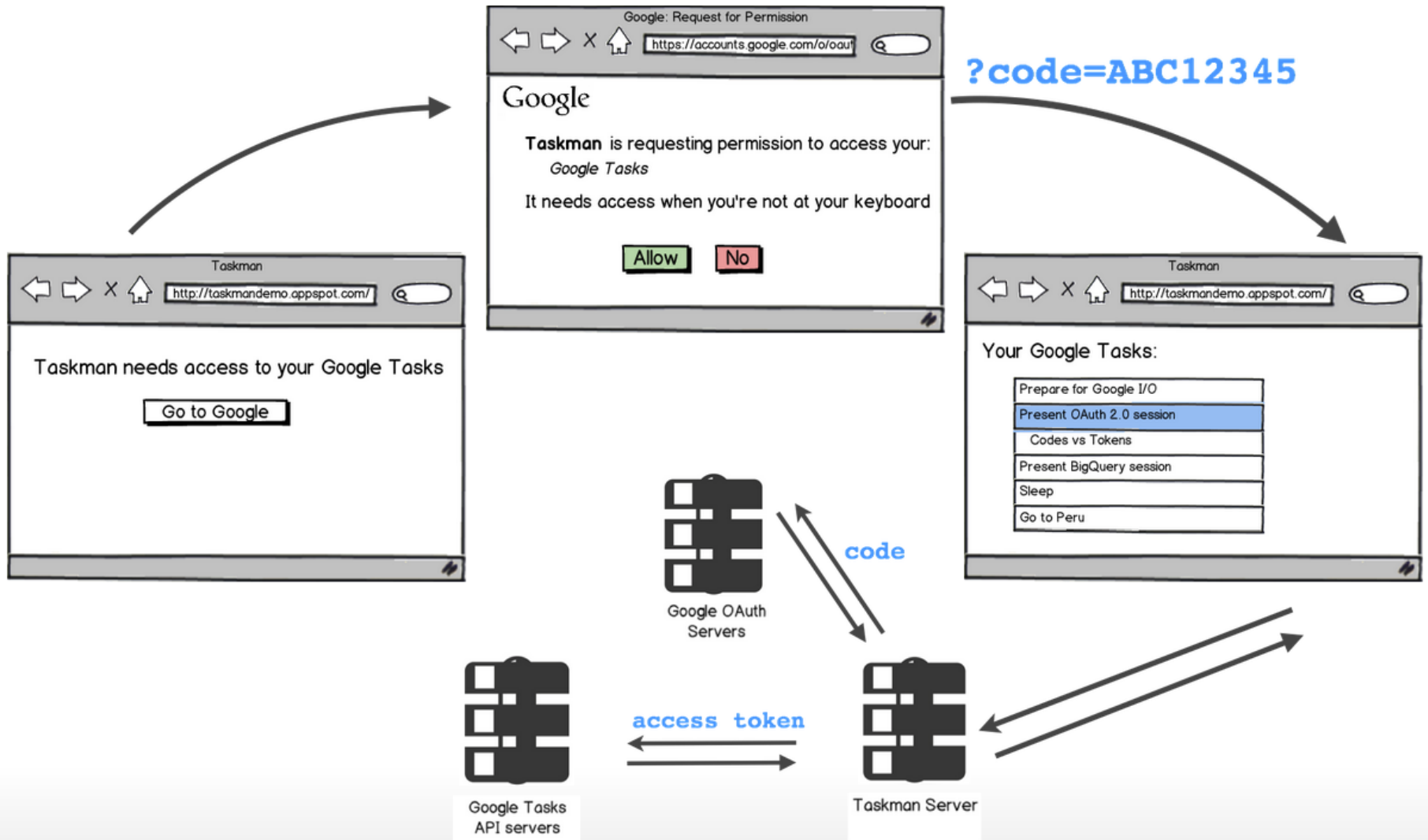
Client-side Flow in JavaScript

Summary

- You're looking for simplicity
- You like coding in JavaScript
- You only need access when the user is logged into Google
- Learn more about the JavaScript library
 - Catch on YouTube: "Building Web Applications that use Google APIs and the JavaScript Client for Google APIs"



Server-side Flow in Python



Server-side Flow in Python

Step 1 - App re-directs the user

PYTHON

```
flow = OAuth2WebServerFlow(  
    # Visit https://code.google.com/apis/console to  
    # generate your client_id, client_secret and to  
    # register your redirect_uri.  
    client_id='387636757294.apps.googleusercontent.com',  
    client_secret='-8IwOyyundwNY6W0B',  
    scope='https://www.googleapis.com/auth/tasks')  
  
callback = self.request.relative_url('/oauth2callback')  
authorize_url = flow.step1_get_authorize_url(callback)  
self.redirect(authorize_url)
```

Server-side Flow in Python

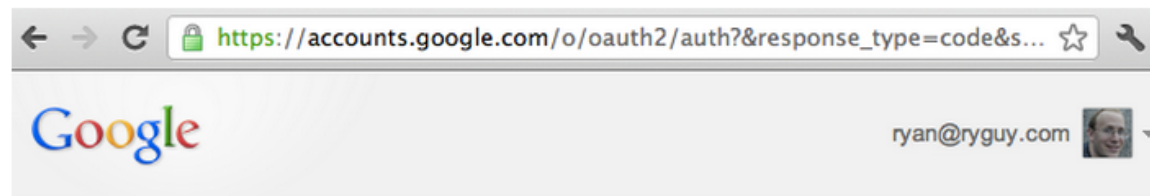
Step 1 - App re-directs the user

```
https://accounts.google.com/o/oauth2/auth?  
  client_id=387636757294.apps.googleusercontent.com&  
  scope=https://www.googleapis.com/auth/tasks&  
  redirect_uri=https://taskmandemo.appspot.com/oauth2callback&  
  response_type=code&  
  access_type=offline
```

URL

Server-side Flow in Python

Step 2a - User authorizes access



Taskman is requesting permission to:

- Manage your tasks
- ◊ Perform these operations when I'm not using the application

Allow access

No thanks

Taskman

[Learn more](#)

Server-side Flow in Python

Step 2b - User is redirected back to the app

```
https://taskmandemo.appspot.com/oauth2callback?  
code=4/jHpY6Rs1b32PWBIR9bU6wJ4GrMmF.Ej0UBPzd
```

URL

Server-side Flow in Python

Step 3 - App exchanges authorization code for access token

```
credentials = flow.step2_exchange(self.request.params)
print 'Access token: %s' % credentials.access_token
```

PYTHON

GOAL!

```
print 'Refresh token: %s' % credentials.refresh_token
```

PYTHON

Server-side Flow in Python

Step 3 - App exchanges authorization code for access token

POST /o/oauth2/token HTTP/1.1

Host: accounts.google.com

client_id=387636757294.apps.googleusercontent.com&
redirect_uri=https://taskmandemo.appspot.com/oauth2callback&
grant_type=authorization_code&
code=4/jHpY6Rslb32PWBir9bU6wJ4GrMmF.EjOUBPzd&
client_secret=-8IwOyyundwNY6W0B

HTTP REQUEST

```
{  
  "access_token": "1/ffAGRNJru1FTz70BzhT3Zg",  
  "refresh_token": "1/xEoDL4iW3cx1I7yDbSRFYNG01kVKM2C-259H0F2aQbI"  
  "expires_in": 3600,  
  "token_type": "Bearer",  
}
```

HTTP RESPONSE

Server-side Flow in Python

Step 4 - App exchanges refresh token for a new access token

```
# nothing to see here!
```

PYTHON

Server-side Flow in Python

Step 4 - App exchanges refresh token for a new access token

POST /o/oauth2/token HTTP/1.1

Host: accounts.google.com

Content-Type: application/x-www-form-urlencoded

client_id=387636757294.apps.googleusercontent.com&

client_secret=-8IwOyyundwNY6W0B&

refresh_token=1/xEoDL4iW3cx1I7yDbSRFYNG01kVKM2C-259H0F2aQbI&

grant_type=refresh_token

HTTP REQUEST

```
{  
  "access_token": "1/hGAFZKUio3N0c90BxjU48l",  
  "expires_in": 3600,  
  "token_type": "Bearer"  
}
```

HTTP RESPONSE

Server-side Flow in Python

Questions!

- When do refresh tokens expire?
- How do I store the refresh tokens?

Server-side Flow in Python

Making it even easier: the decorator pattern

PYTHON

```
decorator = OAuth2Decorator(  
    client_id='387636757294.apps.googleusercontent.com'  
    client_secret='-8Iw0yyundwNY6W0B'  
    scope='https://www.googleapis.com/auth/tasks')  
  
http = httpplib2.Http(memcache)  
service = build("tasks", "v1", http=http)  
  
class MainHandler(webapp.RequestHandler):  
    @decorator.oauth_required  
    def get(self):  
        http = decorator.http()  
        tasks = service.tasks.list(tasklist='@default').execute(http)
```

Server-side Flow in Python

Summary

- You're looking for long-lived access to user data
- You need access when the user isn't at the keyboard
- You need to call the API from server-side code



I have the token. Now what?

Calling the API

But... how do I call the API?

Using access tokens

Using a HTTP Header:

```
GET /tasks/v1/lists/@default/tasks HTTP/1.1  
Host: www.googleapis.com  
Authorization: Bearer 1/hGAFZKUio3N0c90BxjU48l
```

HTTP

Using a Query Parameter:

```
GET /tasks/v1/lists/@default/tasks?access_token=1/hGAFZKUio3N0c90BxjU48l HTTP/1.1  
Host: www.googleapis.com
```

HTTP

Tell me more about scopes

Getting authorization for multiple APIs at once

Space-delimited list!

<https://www.googleapis.com/auth/tasks> <https://www.googleapis.com/auth/plus.me>



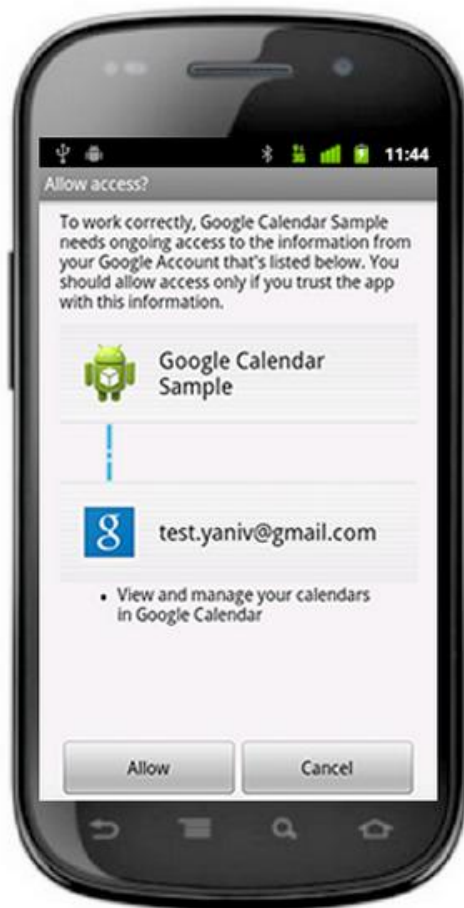
Mobile Authorization

Three Techniques

- Cross-platform - Embedded WebViews
- Cross-platform - System web browser
- Android-specific - GoogleAuthUtil



GoogleAuthUtil



GoogleAuthUtil

App Registration

Create Client ID

Client ID Settings

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).

Installed application type

☒ Android [Learn more](#)
Package name:

Signing certificate fingerprint (SHA1):

☐ Chrome Application [Learn more](#)

☐ iOS [Learn more](#)

☐ Other

Create client ID

Back

Cancel

[Learn more](#)

GoogleAuthUtil

Acquiring an access token

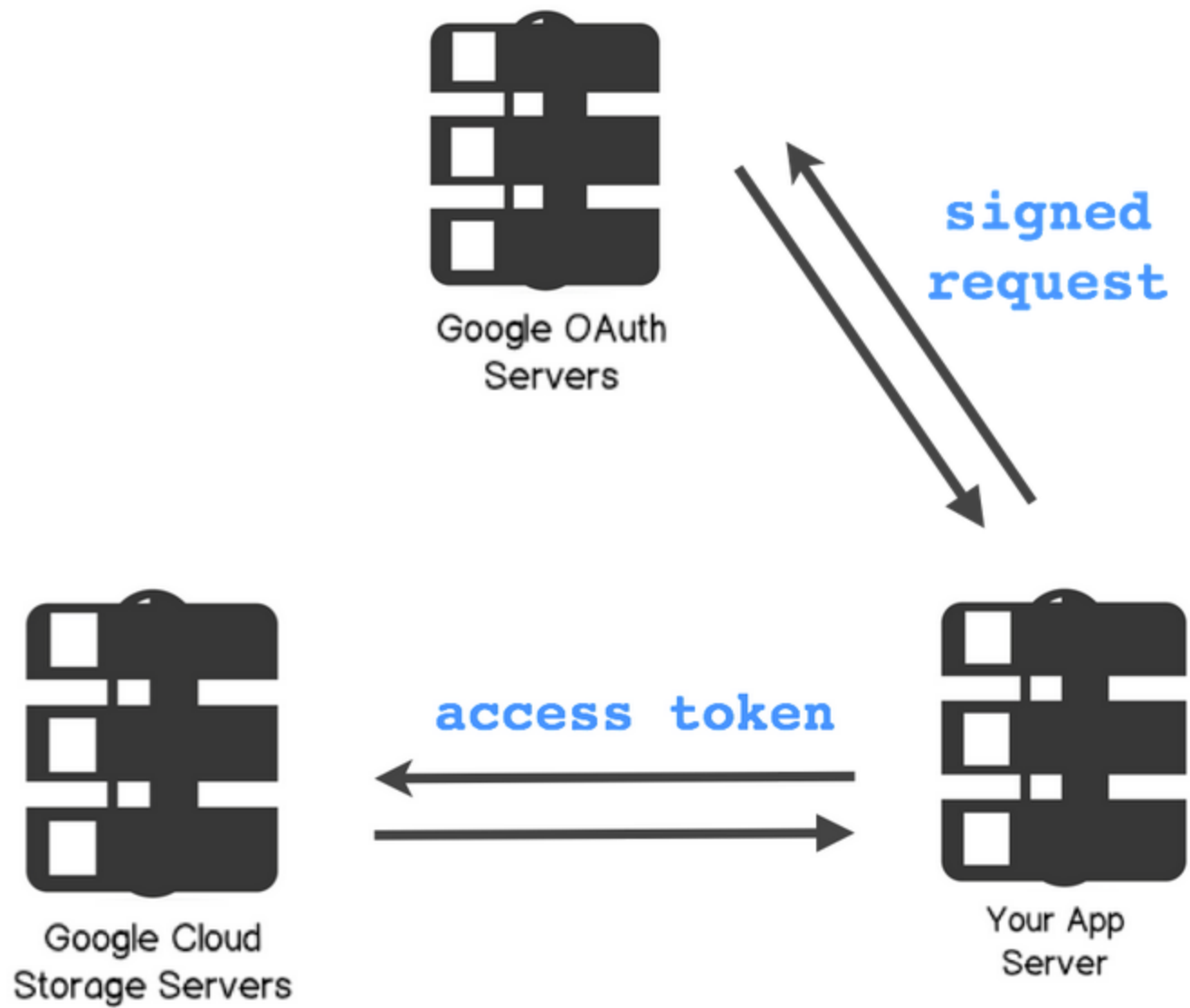
ANDROID

```
// Allow user to select an account  
// AccountManager.getAccountsByType("com.google");  
  
String scope = "https://www.googleapis.com/auth/tasks";  
  
// Get an access token for 'email' account and requested scope  
String token = GoogleAuthUtil.getToken(context, email, scope);
```

GOAL!



App-based Authorization



App-based Authorization

Using service accounts

PYTHON

Load the key in PKCS 12 format that you downloaded from the Google APIs Console

```
f = file('key.p12', 'rb')
```

```
key = f.read()
```

```
f.close()
```

Create an httplib2.Http object to handle our HTTP requests and authorize it

```
credentials = SignedJwtAssertionCredentials( '141491975384@developer.gserviceaccount.com',  
      key,
```

```
      scope='https://www.googleapis.com/auth/devstorage.read_write')
```

```
http = httplib2.Http()
```

```
http = credentials.authorize(http)
```

```
print credentials.access_token
```

GOAL!



OAuth 2.0 for Login

Authentication Goals

Traditional signup form

Sign up today

Email Address:

First Name:

Last Name:

Profile picture:

Choose File

No file chosen

Username:

what you'll use to log in

Password:

at least 8 characters

Confirm Password:

Sign Up!

Authentication Goals

New signup form



Sign up with Google

First name:

Last name:

Email:

Profile:

Authentication Goals

- Make it faster and easier to onboard users
- Securely get a unique, stable user identifier
- Personalize your site

Authentication

Step 1 - Use OAuth to get an access token

Scope	Description
<code>https://www.googleapis.com/auth/userinfo.profile</code>	Unique id, name, profile photo, profile URL, country, language, timezone, birthdate, etc.
<code>https://www.googleapis.com/auth/userinfo.email</code>	E-mail address

Request access:

```
https://accounts.google.com/o/oauth2/auth?  
  client_id=387636757294.apps.googleusercontent.com&  
  scope=https://www.googleapis.com/auth/userinfo.profile%20https://www.googleapis.com/auth/userinfo.email&  
  redirect_uri=https://taskmandemo.appspot.com/oauth2callback-authn.html&  
  response_type=token
```

URL

Authentication

Step 1 - Use Google-provided button and library



```
<script src="http://plus.google.com/js/plusone.js"></script>
<g:plus action="connect"
  clientid="387636757294.apps.googleusercontent.com"
  callback="handleAuthResult">
</g:plus>
```

JAVASCRIPT

```
function handleAuthResult() {
  alert(gapi.auth.getToken());
}
```

JAVASCRIPT

Authentication

Step 2 - Use TokenInfo API to get a secure, unique user identifier

My id:

```
https://www.googleapis.com/oauth2/v1/tokeninfo?  
access_token=
```

URL

```
{  
  "issued_to": "387636757294.apps.googleusercontent.com",  
  "audience": "387636757294.apps.googleusercontent.com",  
  "user_id": "113487456102835830811",  
  "scope": "https://www.googleapis.com/auth/userinfo.profile https://www.googleapis.com/auth/userinfo.email",  
  "expires_in": 3574,  
  "email": "ryan@ryguy.com",  
  "verified_email": true,  
  "access_type": "online"  
}
```

JSON

Personalization

Use UserInfo API to get user profile data

```
https://www.googleapis.com/oauth2/v1/userinfo?  
access_token=
```

URL

```
{  
  "id": "113487456102835830811",  
  "email": "ryan@ryguy.com",  
  "verified_email": true,  
  "name": "Ryan Boyd",  
  "given_name": "Ryan",  
  "family_name": "Boyd",  
  "link": "https://plus.google.com/113487456102835830811",  
  "picture": "https://lh4.googleusercontent.com/-BET-bMzn99g/AAAAAAAAAAI/AAAAAAAAQ8/wV0kQ3VUjOE/photo.jpg",  
  "gender": "male",  
  "birthday": "0000-10-05",  
  "locale": "en"  
}
```

JSON



Tools

OAuth 2.0 Playground

The screenshot shows the OAuth 2.0 Playground interface in a web browser. The browser's address bar displays the URL: `https://oauth2playground.appspot.com/?code=4/V2fsnya7yglhiCOap1ZPQqPalVN5#step2&`. The page features the Google logo at the top. Below it, a header bar contains the text "OAuth 2.0 Playground" and a close button (X). The main content area is divided into two columns. The left column, titled "Step 2 Exchange authorization code for tokens", contains a paragraph explaining the process: "The Authorization code below is provided by the OAuth authorization endpoint if you go through step 1. Click the 'Exchange authorization code for tokens' button, this will use the OAuth token endpoint to exchange the code for a refresh and an access token which is required to access OAuth protected resources." Below this text is a text input field labeled "Authorization code:" containing the value `4/V2fsnya7yglhiCOap1ZPQqPalVN5`. A blue button labeled "Exchange authorization code for tokens" is positioned below the input field. Further down, there are two more input fields: "Refresh token:" and "Access token:", each with a corresponding "Refresh" button. A note at the bottom of this section states: "Note: The refresh tokens never expires. Users have to go to their Google Account [Authorized Access](#) page if they would like to manually revoke them." The right column, titled "Request / Response", displays the HTTP response details. It shows "HTTP/1.1 302 Found", the "Location" header pointing to `https://accounts.google.com/o/oauth2/auth?scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Ftasks&redirect_uri=https%3A%2F%2Foauth2playground.appspot.com&response_type=code&client_id=407408718192.apps.googleusercontent.com`, and the "GET" request line: `GET /?code=4/V2fsnya7yglhiCOap1ZPQqPalVN5 HTTP/1.1` with the host `Host: oauth2playground.appspot.com`.

OAuth 2.0 Login Demo

OpenID Connect Demo w/Google [Connect Me!](#)

- Step 1: Show login UI
- Step 2: Grab token
- Step 3: Validate It
- Step 4: Get UserInfo

The user is presented with an element enticing them to login. In the simplest case, you can redirect the browser to a resource on the OpenID Connect provider. Popups are probably a tad better.

The request should include parameters to approximate the permissions the application is requesting. These parameters are shown below.

Parameter Name	Value
scope	https://www.googleapis.com/auth/userinfo.email https://www.googleapis.com/auth/userinfo.profile (list can be extended)
state	/ (the RP can add it's own value)
redirect_uri	https://oauthssodemo.appspot.com/oauthcallback (registered in the dev console)
response_type	token
client_id	8819981768.apps.googleusercontent.com (registered in the dev console)

Scope is a concatenated list of requested permissions. The two shown in the table request basic profile information and email address (with verification)

State is used to correlate the response. It is reflected back to the site.

redirect_uri is registered with the Authorization server a priori, and the Authorization Server uses it as a target for the response.

response_type indicates whether or not the site would like an access token or a code+refresh token. In this case, only an access token is needed.

client_id is used along with the redirect_uri to signal to the Authorization Server the application making the request.

The full URL of the target (with parameters) is:

```
https://accounts.google.com/o/oauth2/auth?scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.profile&state=%2F&redirect_uri=https%3A%2F%2Foauthssodemo.appspot.com%2Foauthcallback&response_type=token&client_id=8819981768.apps.googleusercontent.com
```



Summary

Resources

- These slides - oauth2-pres0.appspot.com
- APIs Console - developers.google.com/console
- JavaScript library - code.google.com/p/google-api-javascript-client
- Python library - code.google.com/p/google-api-python-client
- OAuth 2.0 Playground - code.google.com/oauthplayground
- OAuth 2.0 Login Demo - oauthssodemo.appspot.com
- Google Play Services - developers.google.com/android/google-play-services

<Thank You!>



g+ profiles.google.com/ryan.boyd
twitter [@ryguyrg](https://twitter.com/ryguyrg)
www www.ryguy.com