OAuth 2.0 in Depth

By Rohit Ghatol

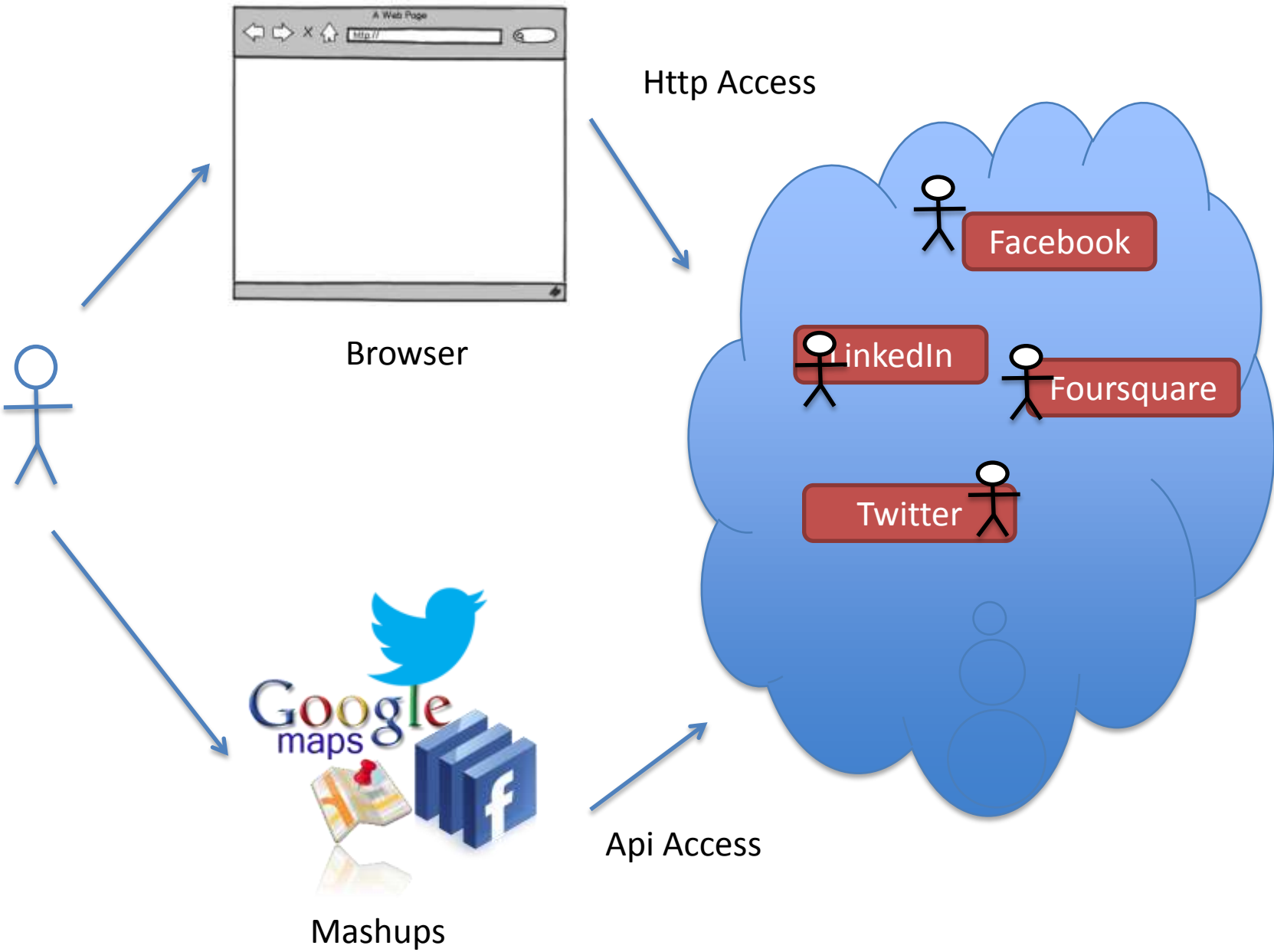
Director @ Synerzip

Passionate about TechNext

Why study about OAuth?

Do you care about these or Similar Sites?

Reference - http://rainbowseo.com/wp-content/uploads/2012/06/smm.png


A Web Page
http://
Http Access
Browser
Facebook
LinkedIn
Foursquare
Twitter
Google
maps
Api Access
Mashups

7155 APIs listed on
http://ProgrammableWeb.com

www.programmableweb.com

Register / Login

Find APIs, mashups, code and developers Search

programmableweb

Hot APIs » Twitter YouTube Facebook Google Maps Flickr LinkedIn More »

Latest news Intelligent Learning With CCKF's RealizeIt API

Home API News API Directory Mashups Community How-to Subscribe:

Keeping you up to date with APIs, mashups and the Web as platform. Learn more »

Find APIs, mashups, code and developers Search

Popular searches: photo google flash mapping enterprise sms

7155 APIs

6760 Mashups

New APIs

- LocalWiki
- WhyGo
- GoMobileIQ Headlight
- AppNowGo!
- FiftyOne
- Totango
- See more APIs

Mashup of the Day

See previous winners

New Mashups

- Twups
- Erfahrungen.com
- Zip Code Catcher
- Furkot - free online road trip planner
- PhoneDuty
- Youplaylist.com
- See more mashups

390 APIs on http://ProgrammableWeb.com
support OAuth

www.programmableweb.com/apis/directory/1?auth=OAuth

Home | API News | API Directory | Mashups | Community | How-to | Subscribe:

Web Services Directory

Subscribe to get the latest APIs

Hide Filters

Sort by: Name | Date | Popularity | Category

Keywords | Category | Company | Protocols / Styles

Data Format | Date: All | Managed By

Filter This List

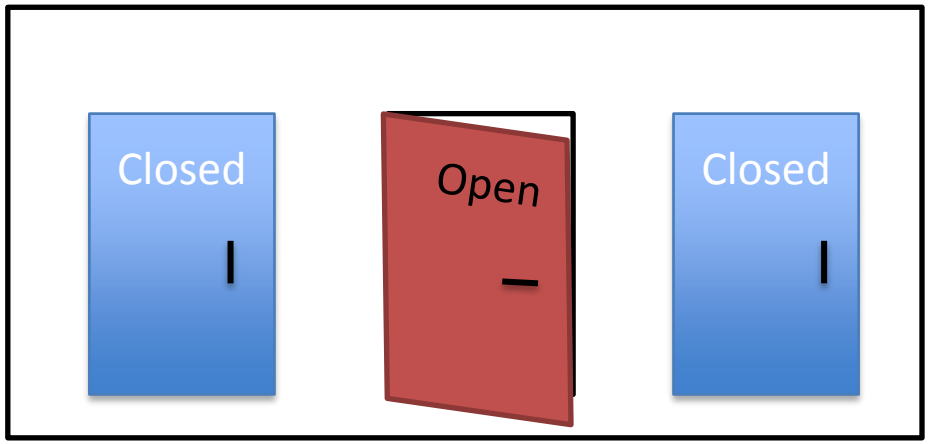
Viewing 1 to 390 of 390 APIs

Previous 1 Next

API	Description	Category	Updated
#blue	Text messaging storage service	Messaging	2011-04-23
11870	Spanish bookmarking and directory service	Search	2011-12-09
1DayLater	Business expense tracking tool	Enterprise	2010-02-07
500px	Online community for photographers	Photos	2011-10-18
7digital	Music downloads store	Music	2010-08-13
88 Miles	Project time tracking services	Project Management	2008-08-14

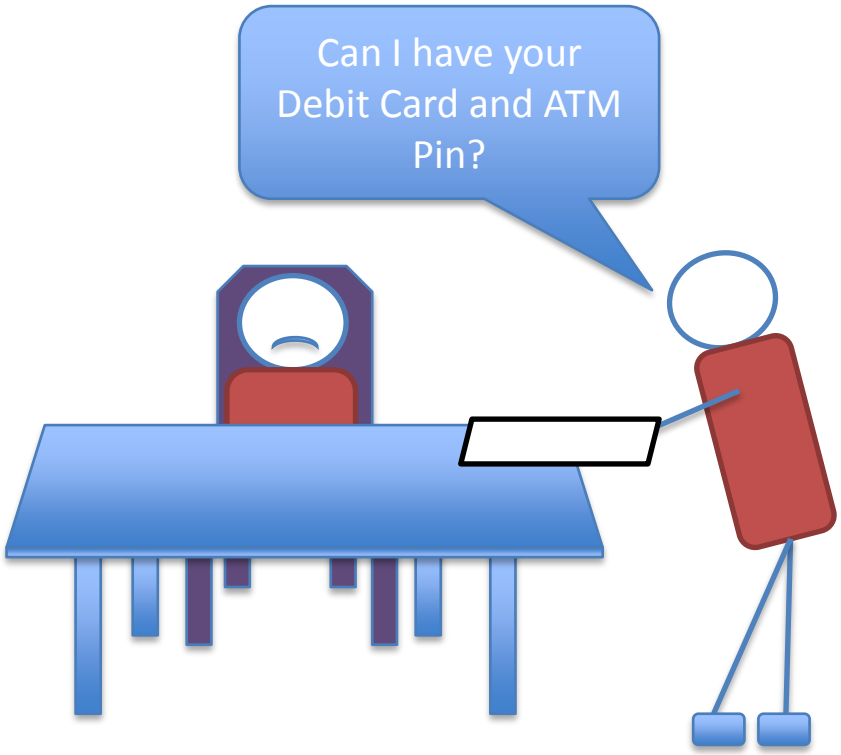
Security

Authentication

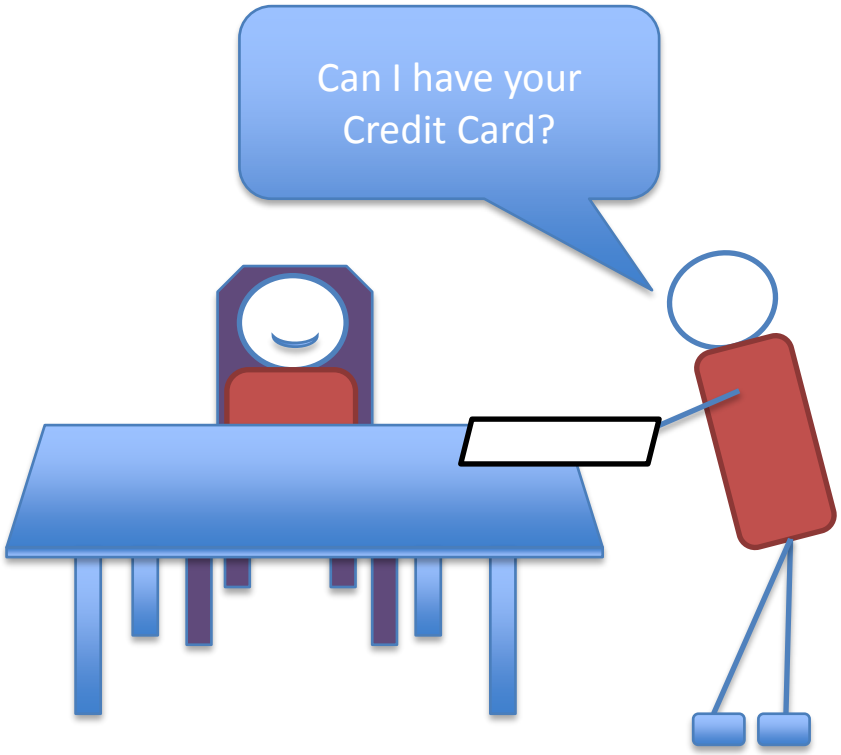


Authorization

OAuth In a Nut Shell



OAuth In a Nut Shell

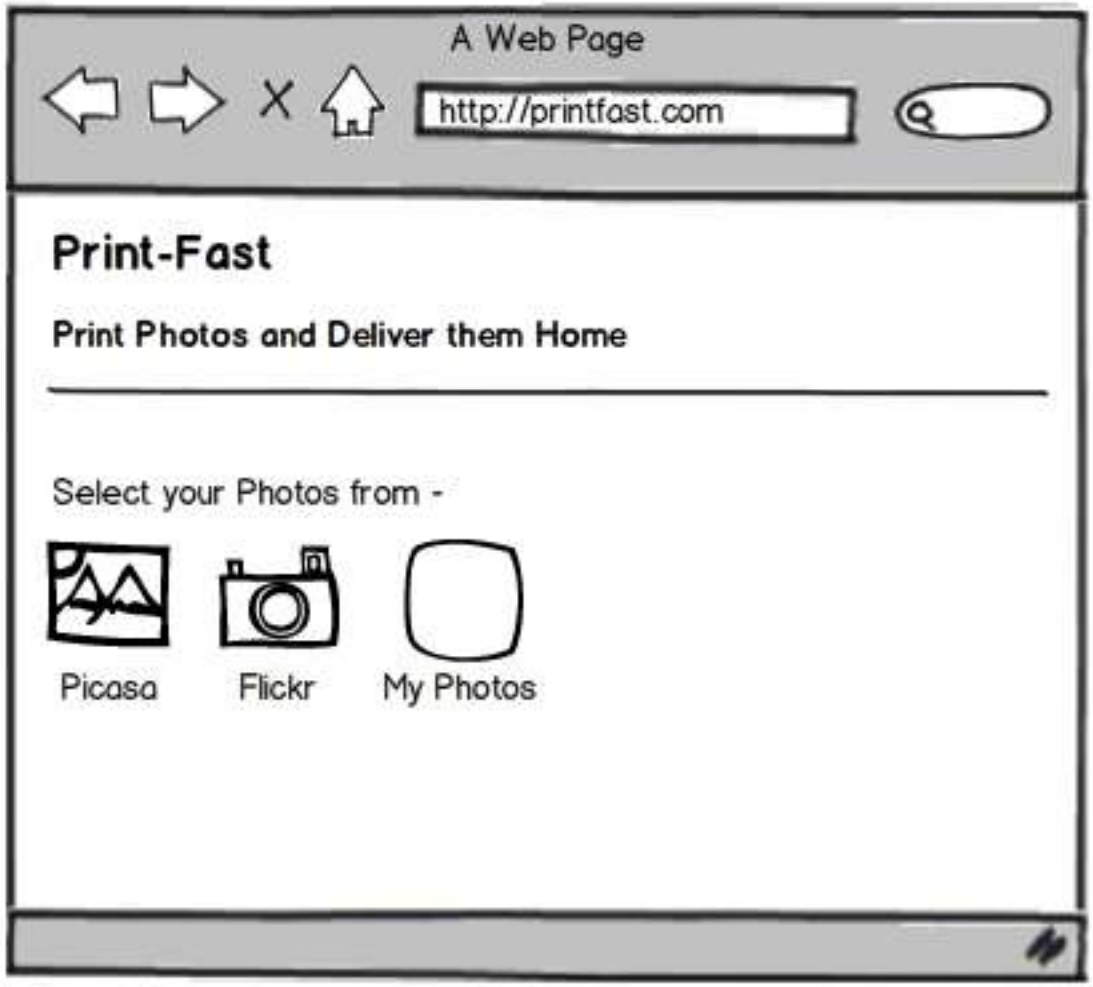


OAuth Practical Example

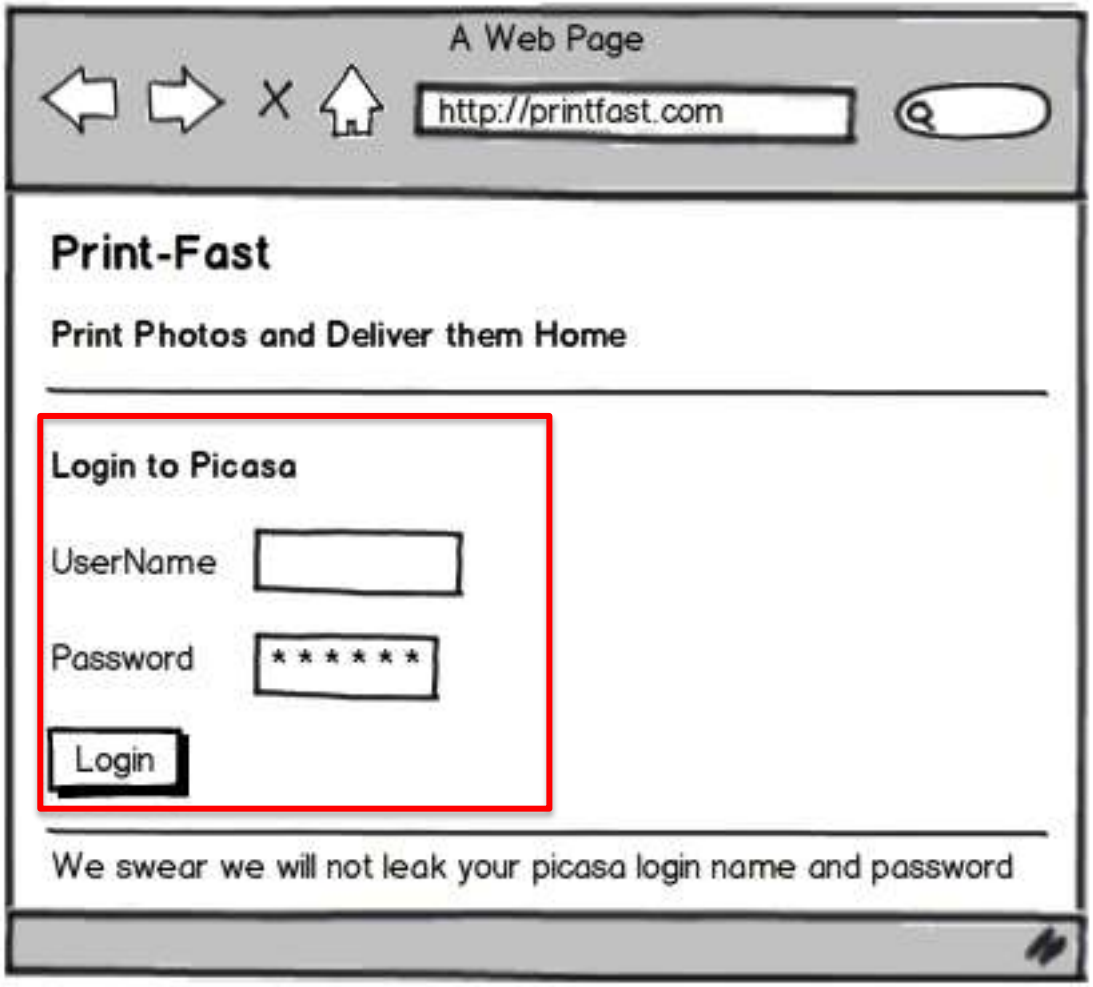
Disclaimer before you read ahead:

All product names and people names used in the following slides are not entirely accurate. They are only placeholders to explain the concept. None of that information should assumed to be correct or incorrect.

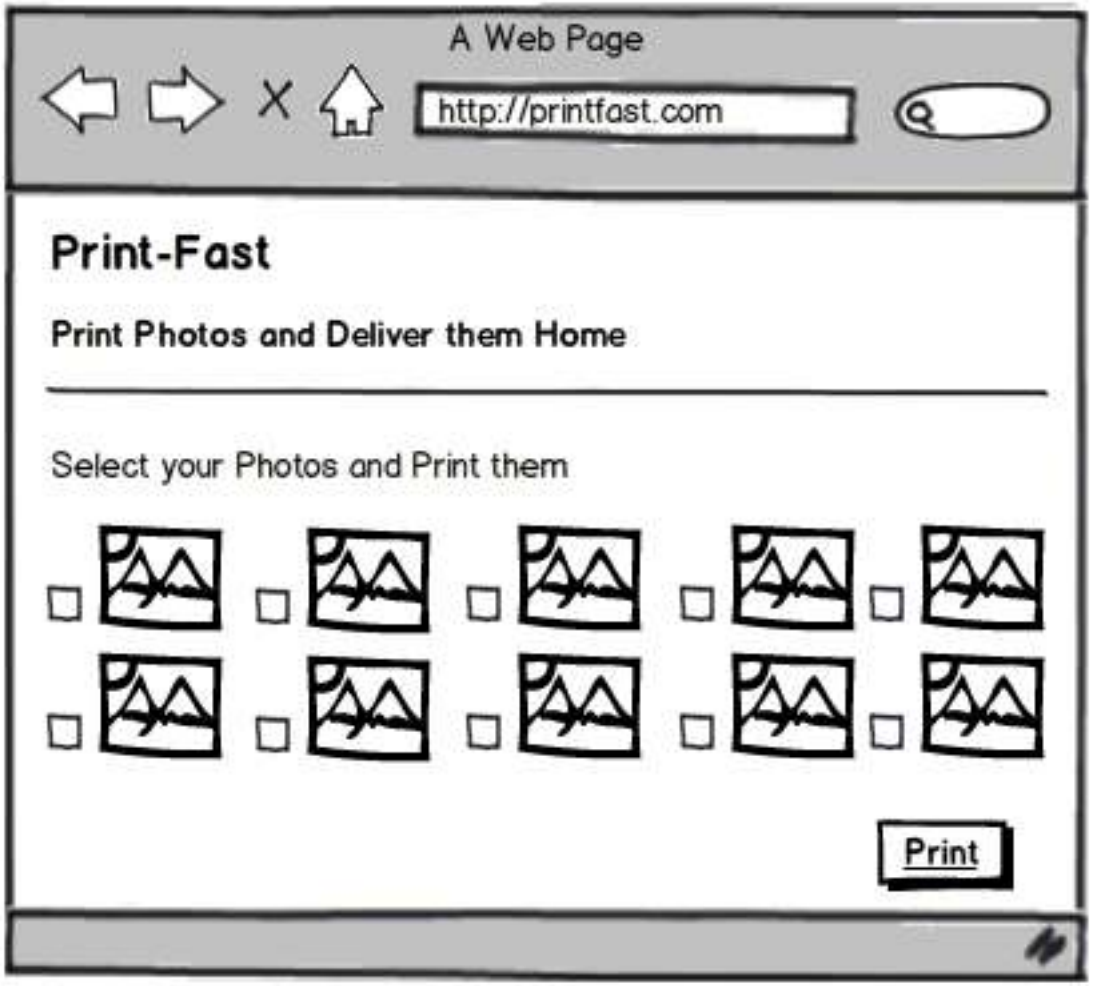
Without OAuth



Without OAuth

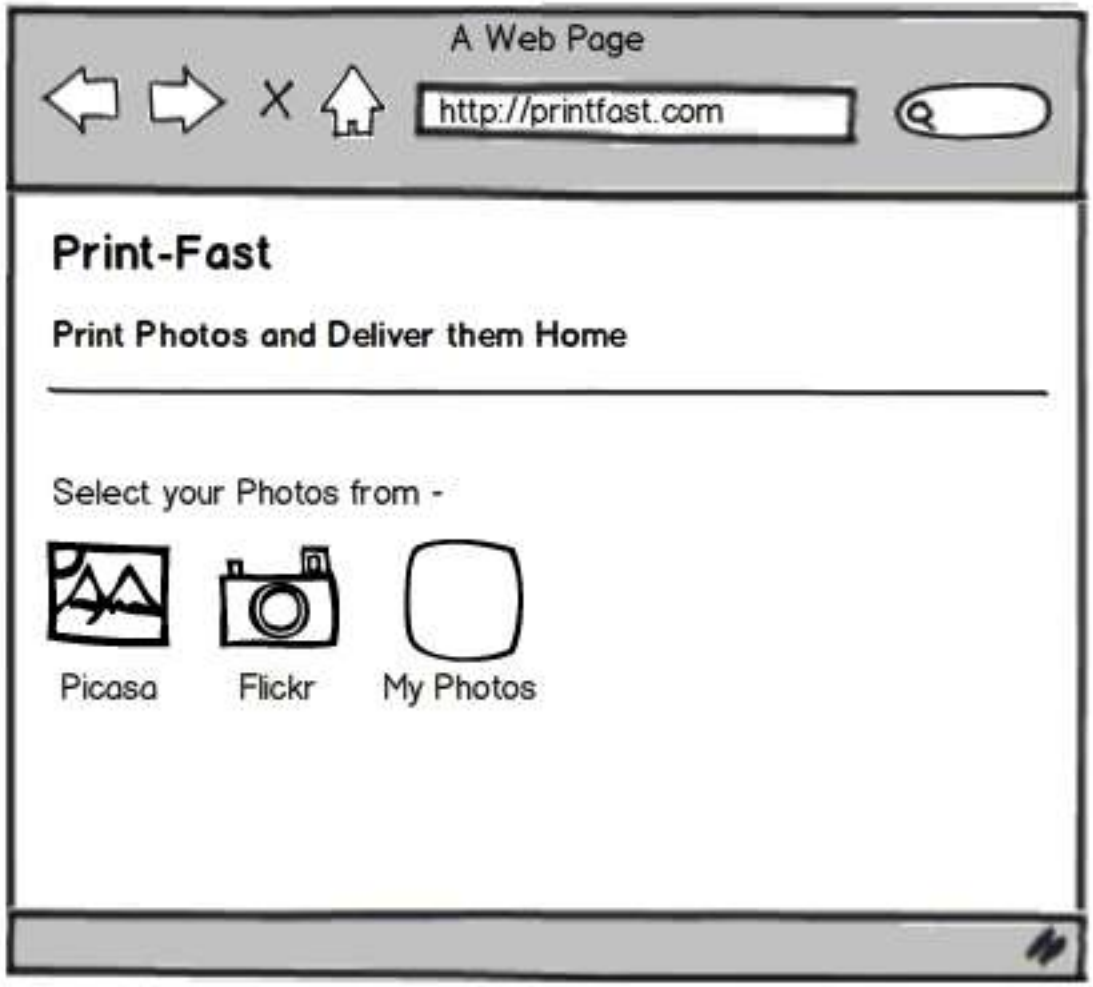


Without OAuth

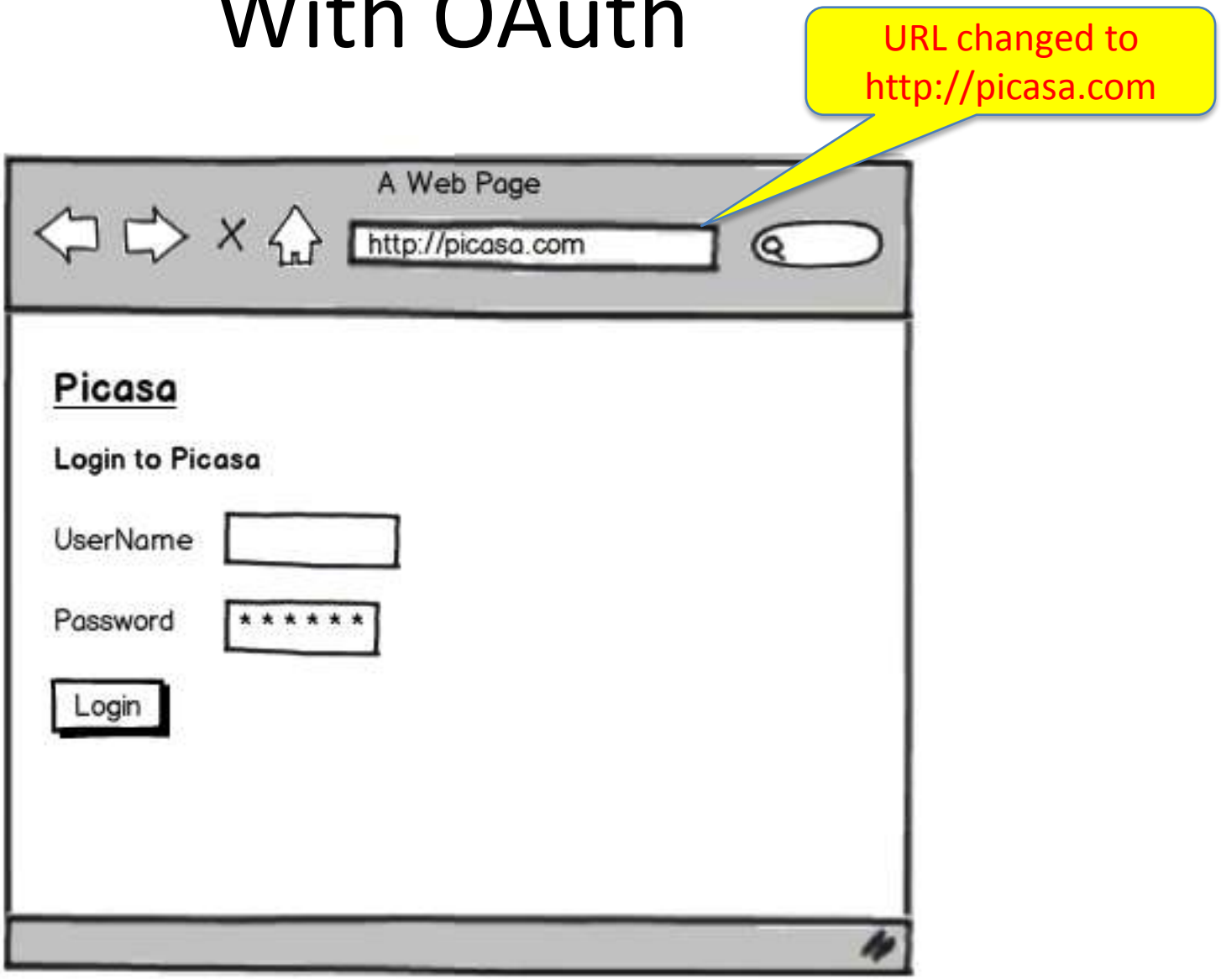


Lets Start Again

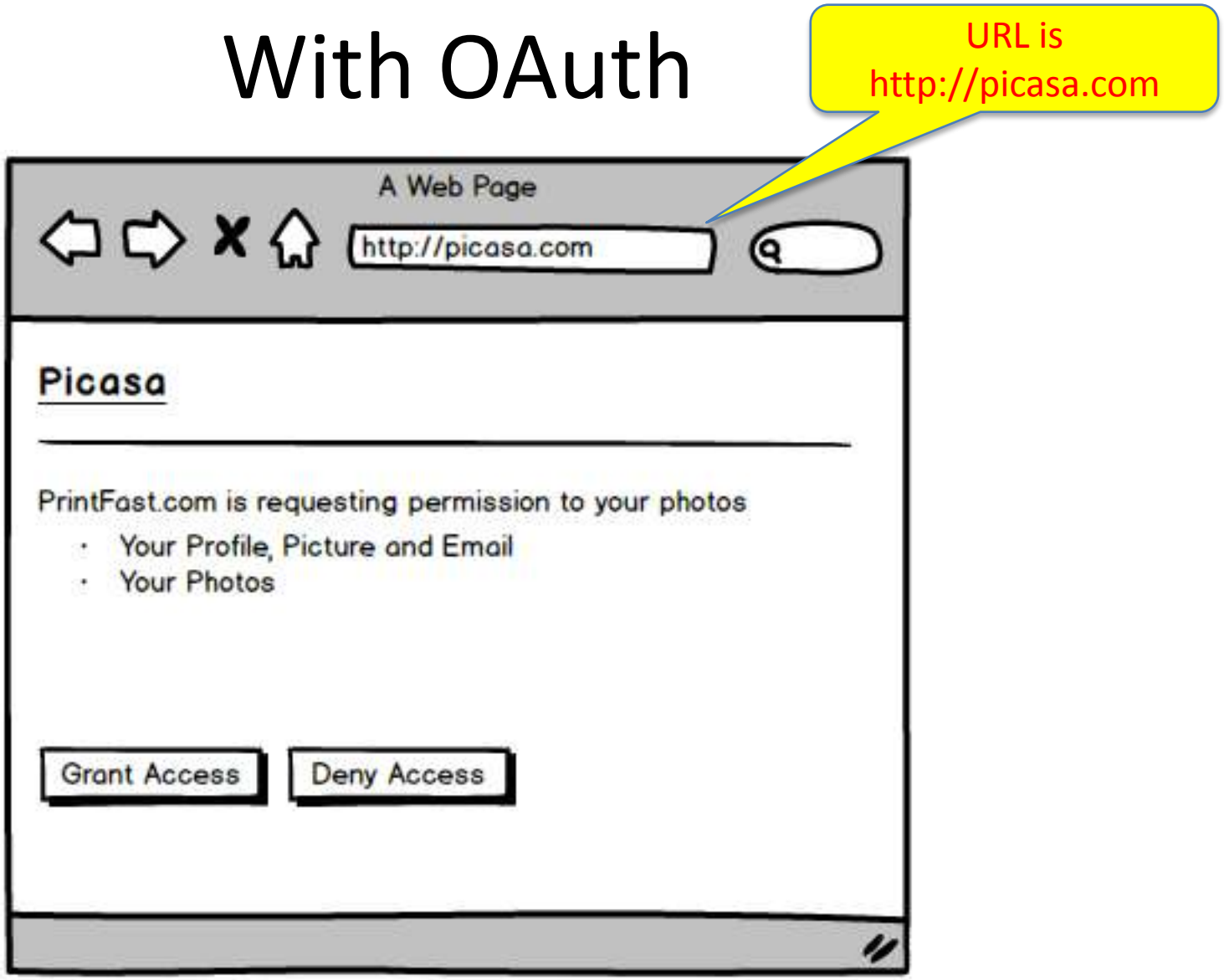
With OAuth


A Web Page
http://printfast.com
Print-Fast
Print Photos and Deliver them Home
Select your Photos from -
Picasa
Flickr
My Photos

With OAuth

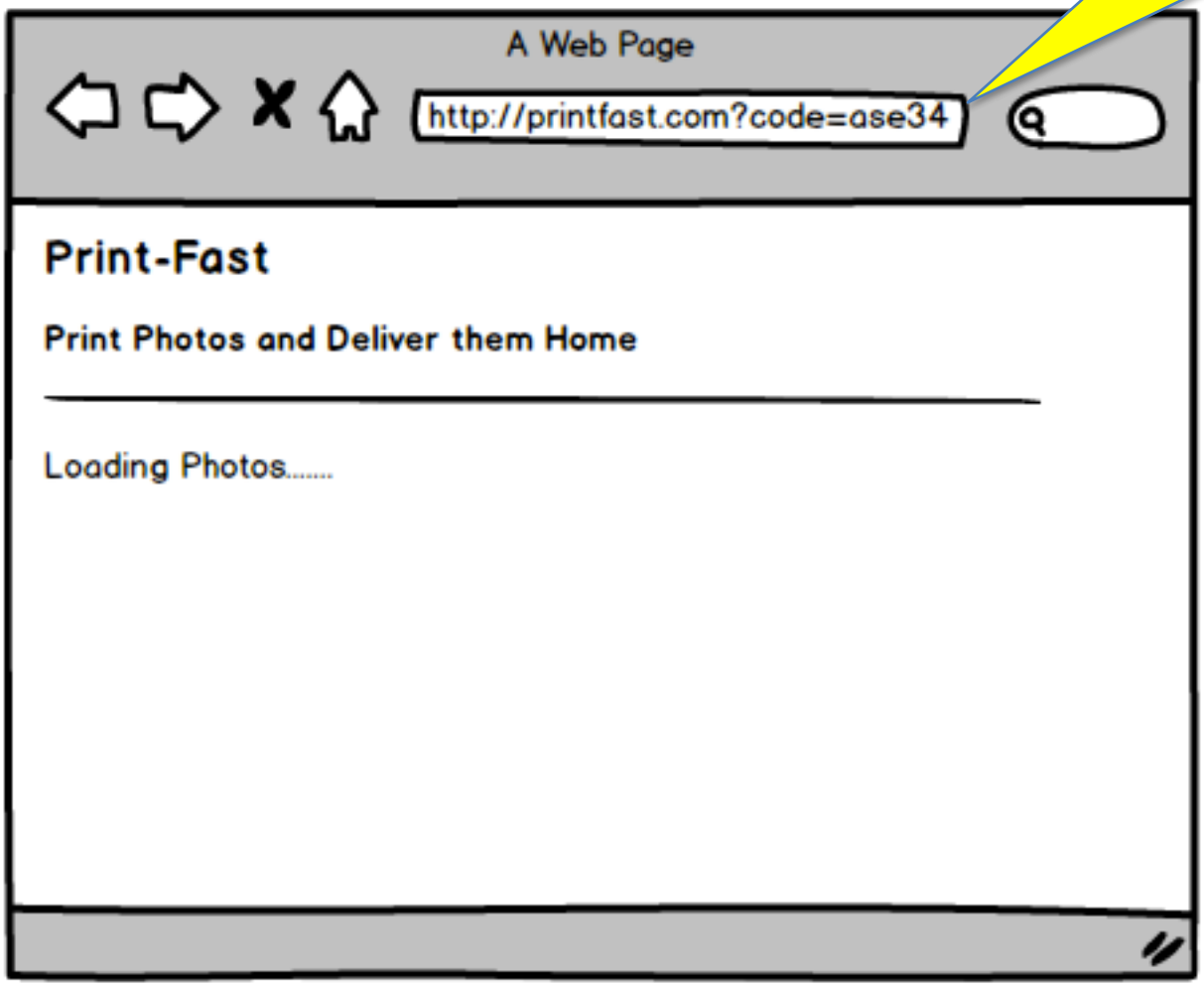

URL changed to
http://picasa.com
A Web Page
http://picasa.com
Picasa
Login to Picasa
UserName
Password
* * * * * *
Login

With OAuth

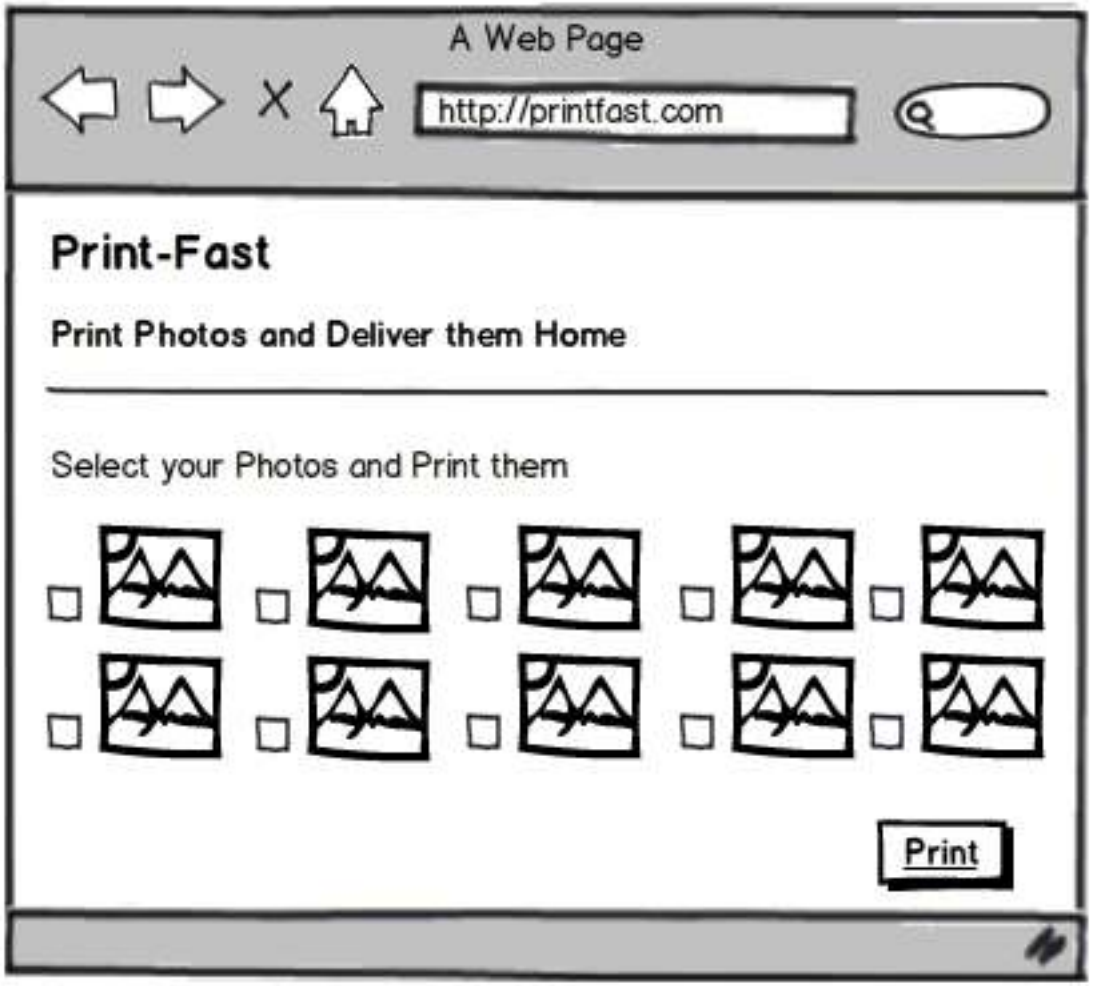

URL is
http://picasa.com
A Web Page
http://picasa.com
Picasa
PrintFast.com is requesting permission to your photos
Your Profile, Picture and Email
Your Photos
Grant Access
Deny Access

With OAuth

URL changed to http://picasa.com with code parameter

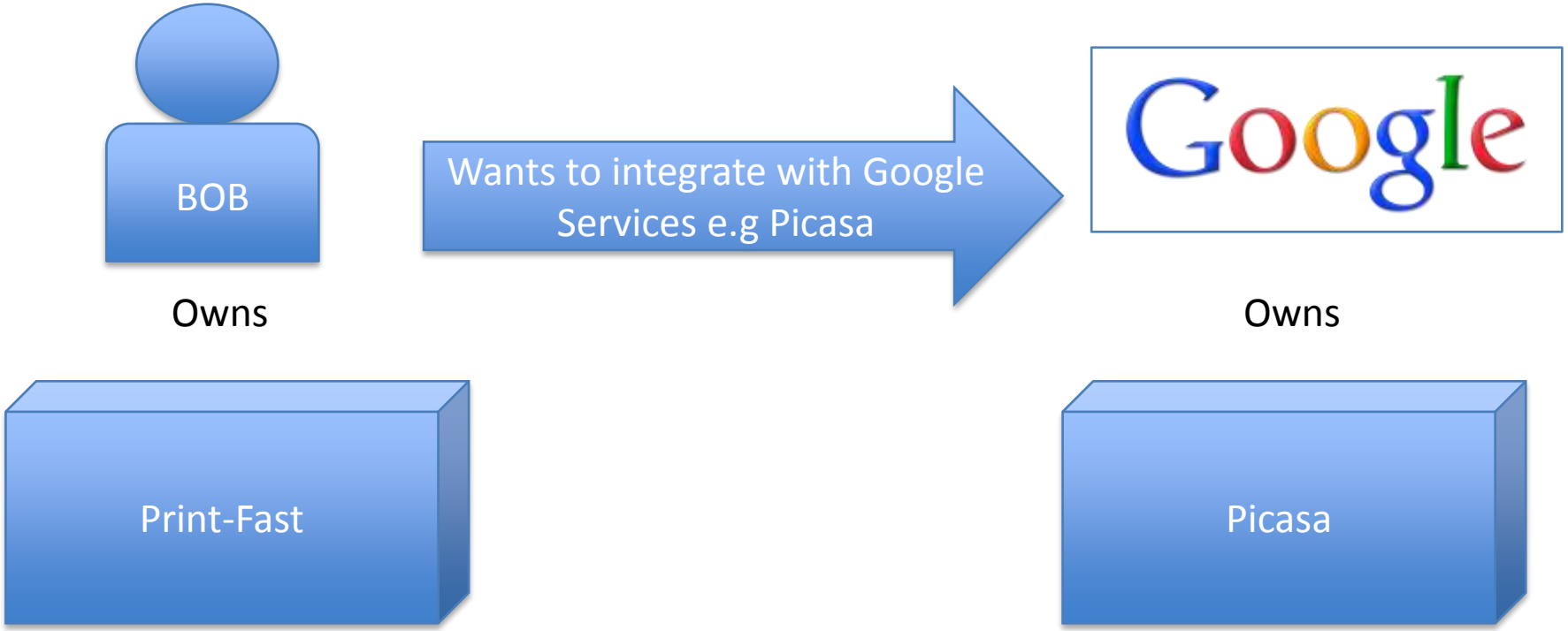


With OAuth

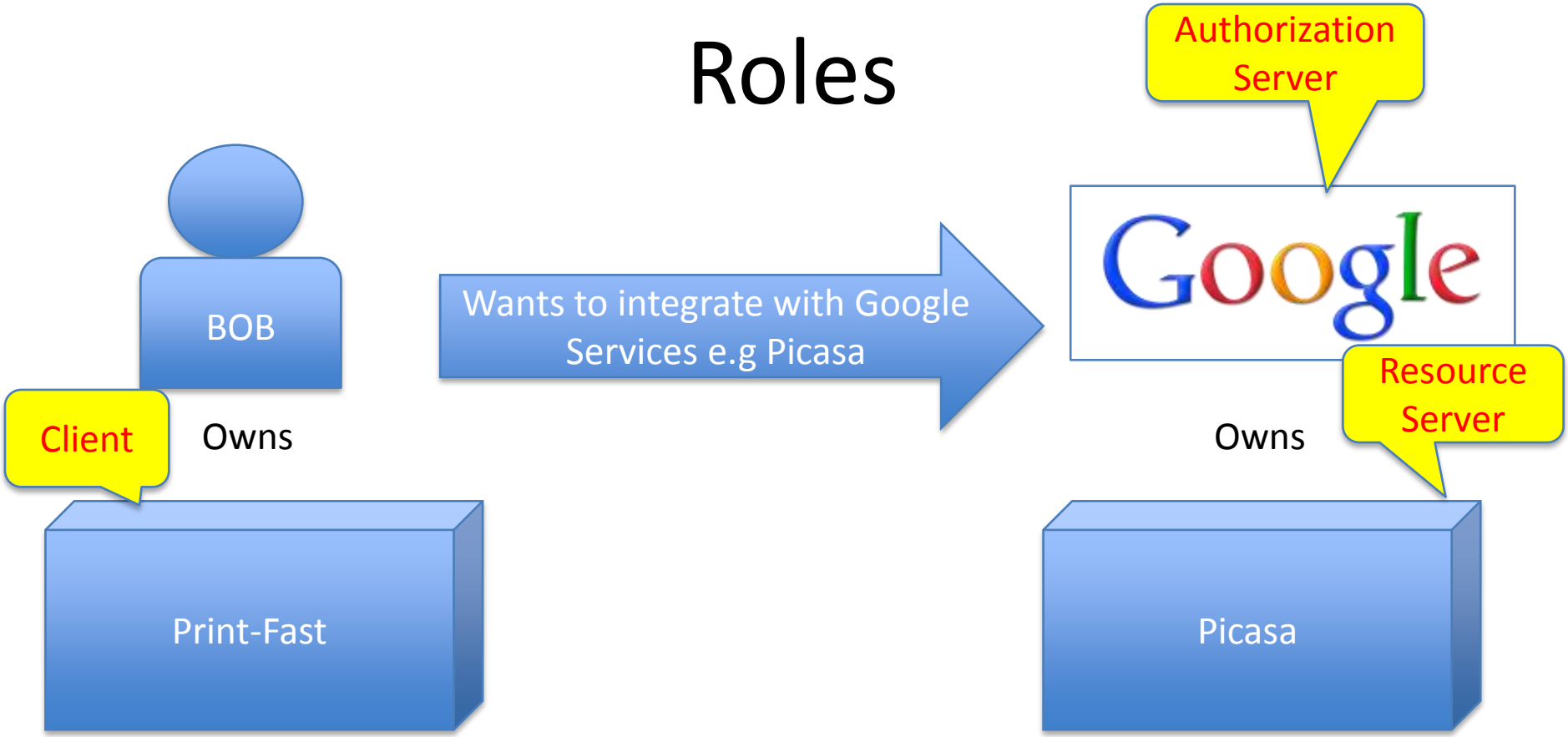

A Web Page
http://printfast.com
Print-Fast
Print Photos and Deliver them Home
Select your Photos and Print them
Print

OAuth 2.0 Flow in Depth

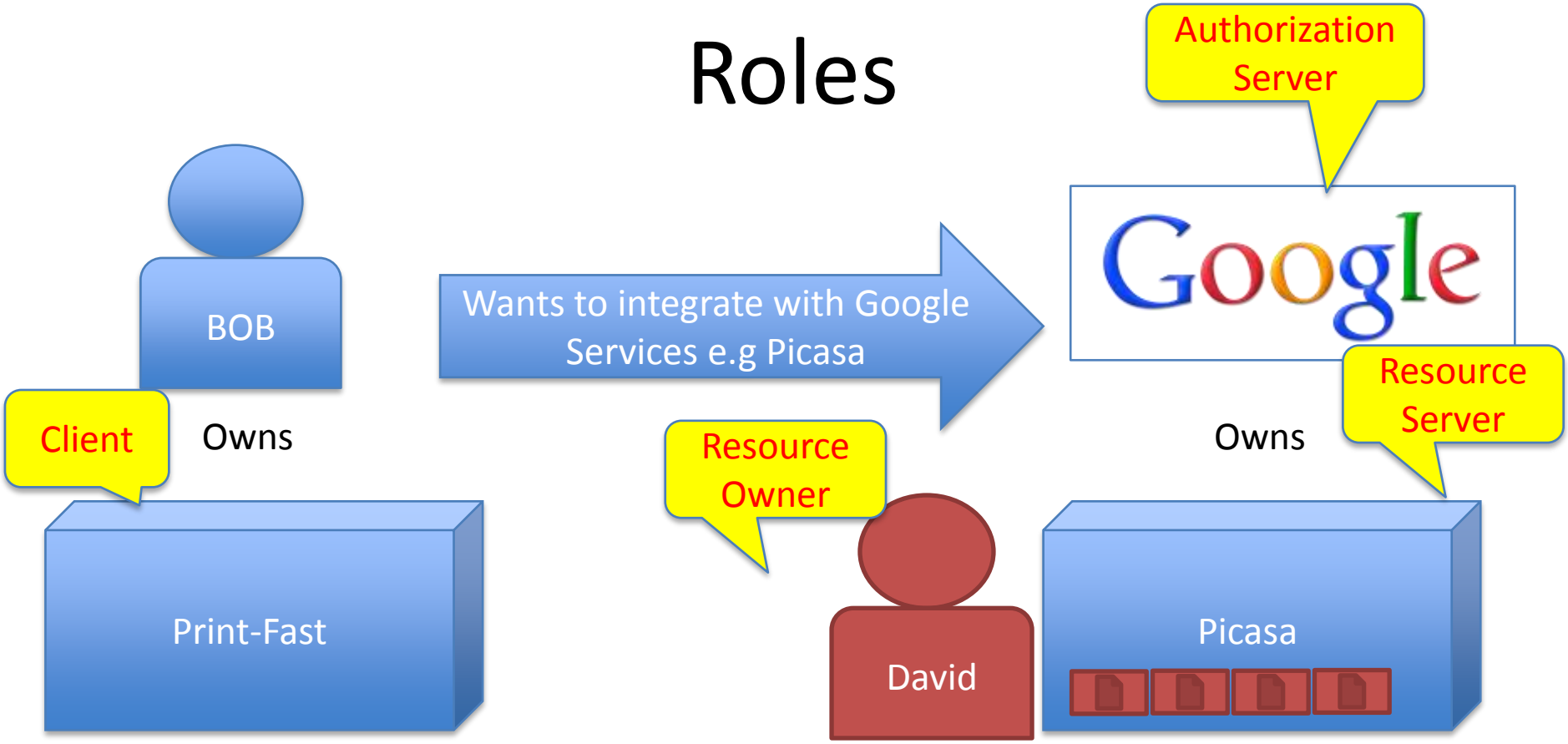
Scenario


BOB
Wants to integrate with Google Services e.g Picasa
Google
Owns
Owns
Print-Fast
Picasa

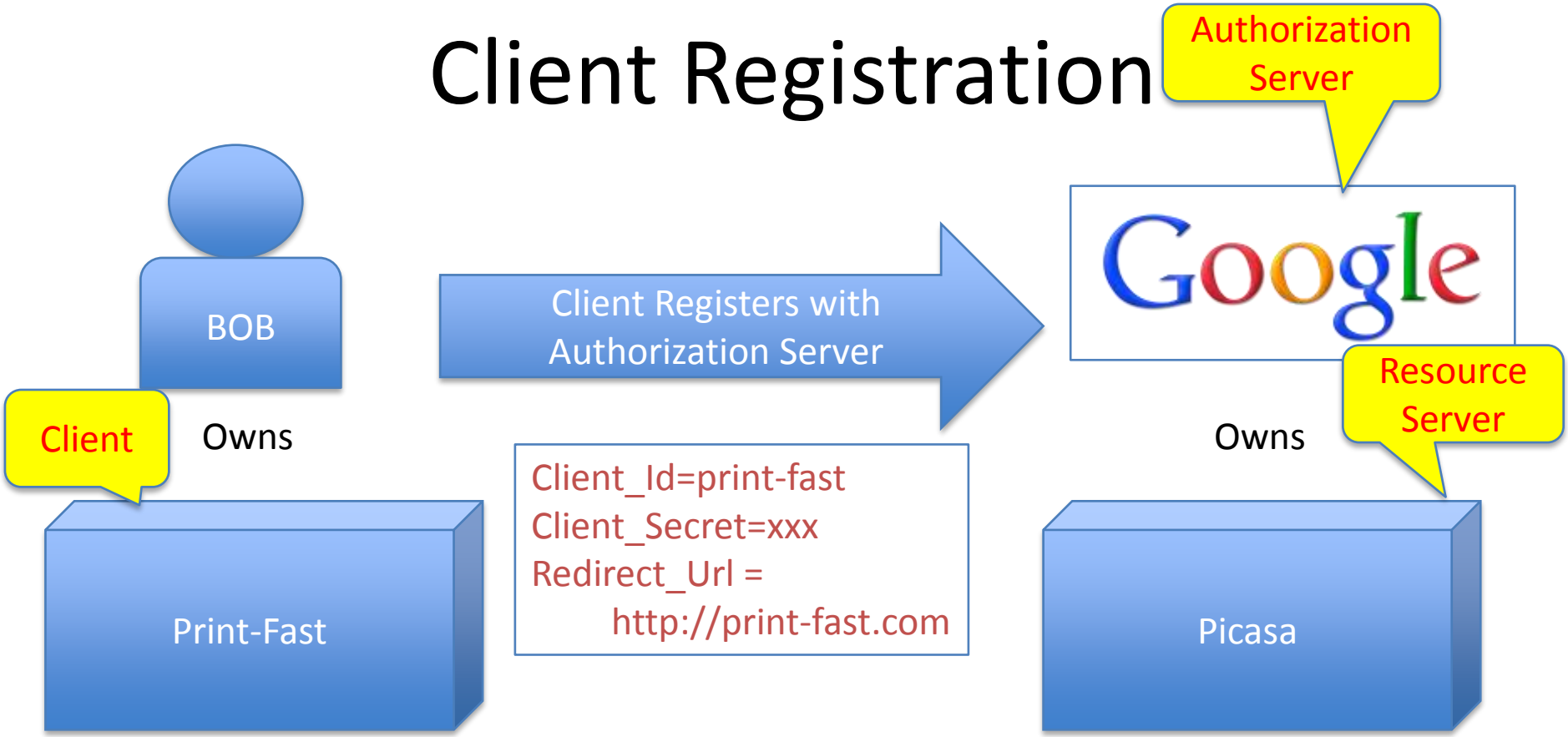
Roles


Authorization Server
BOB
Wants to integrate with Google Services e.g Picasa
Google
Client
Owns
Resource Server
Owns
Print-Fast
Picasa

Roles


Authorization Server
Google
BOB
Wants to integrate with Google Services e.g Picasa
Client
Owns
Resource Owner
Owns
Resource Server
Print-Fast
David
Picasa

Client Registration

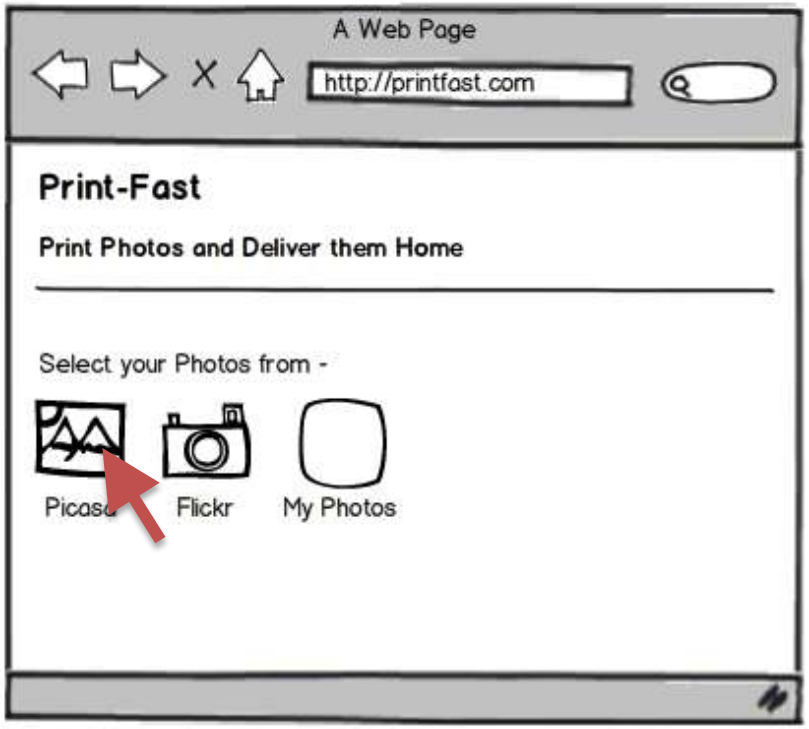


OAuth Flows/Grant Types

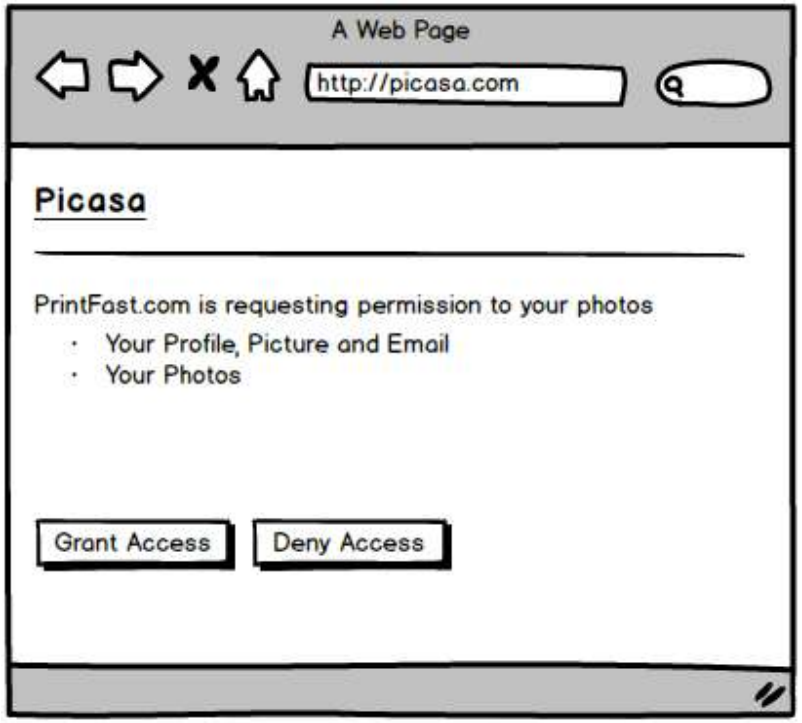
- Authorization Code Grant
- Implicit Grant
- Resource Owner Password Credentials Grant
- Client Credentials Grant

Step 1 – Get Authorization Grant

Authorization Request

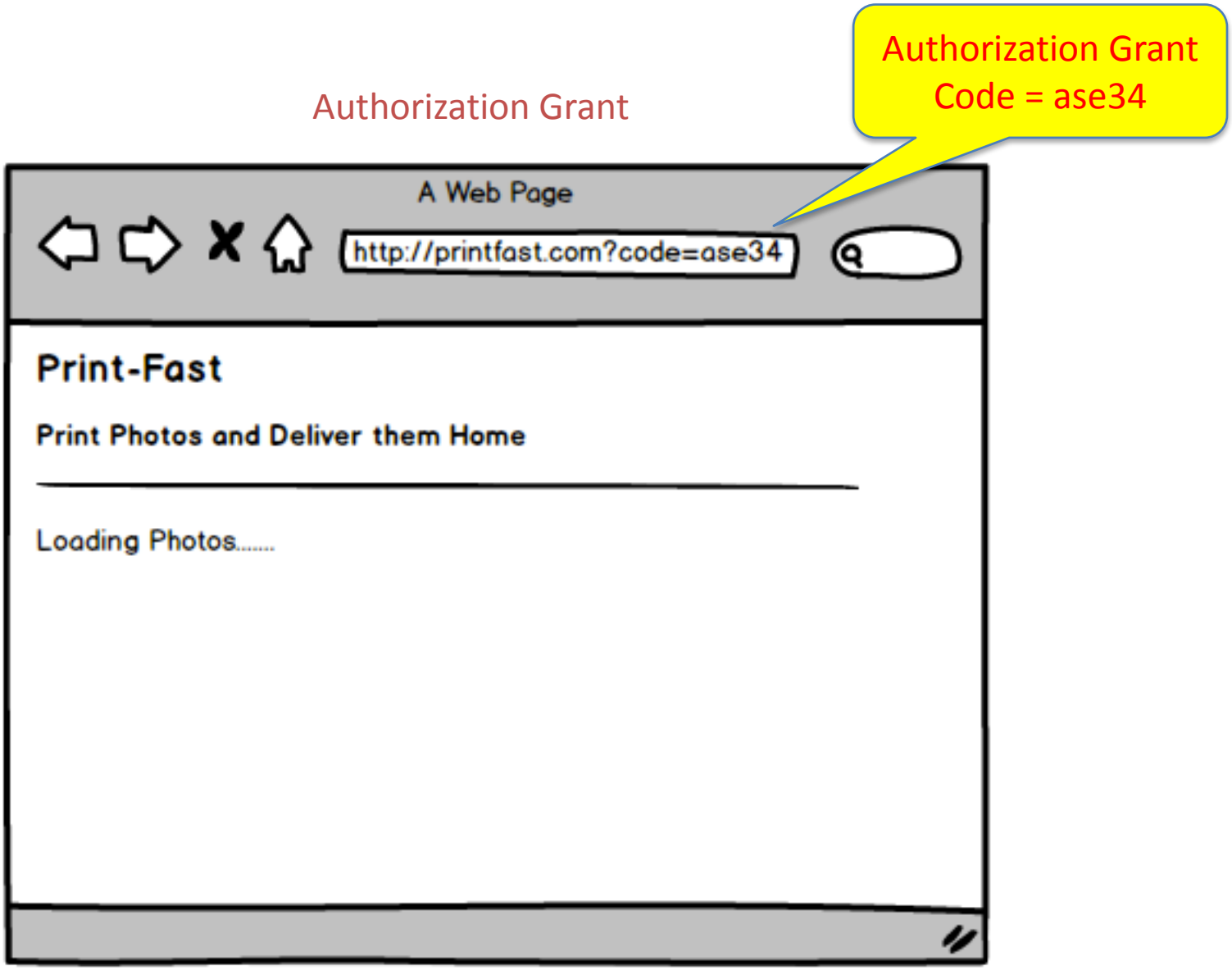


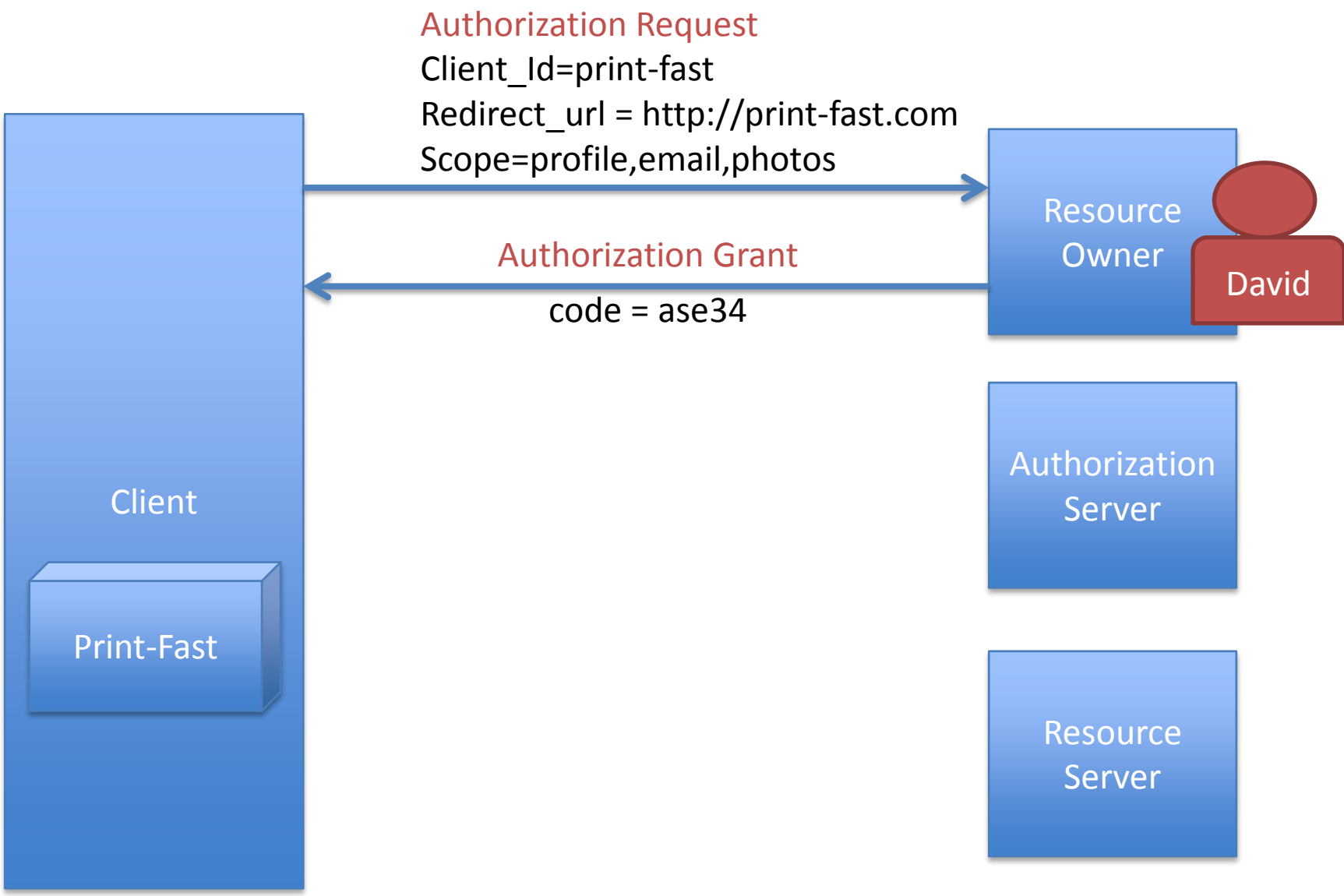
Authorization Grant



URL used is
http://picasa.com/?client_id=photo-fast &scope=profile,email,photos
&redirect_uri=http://print-fast.com&response_type=code

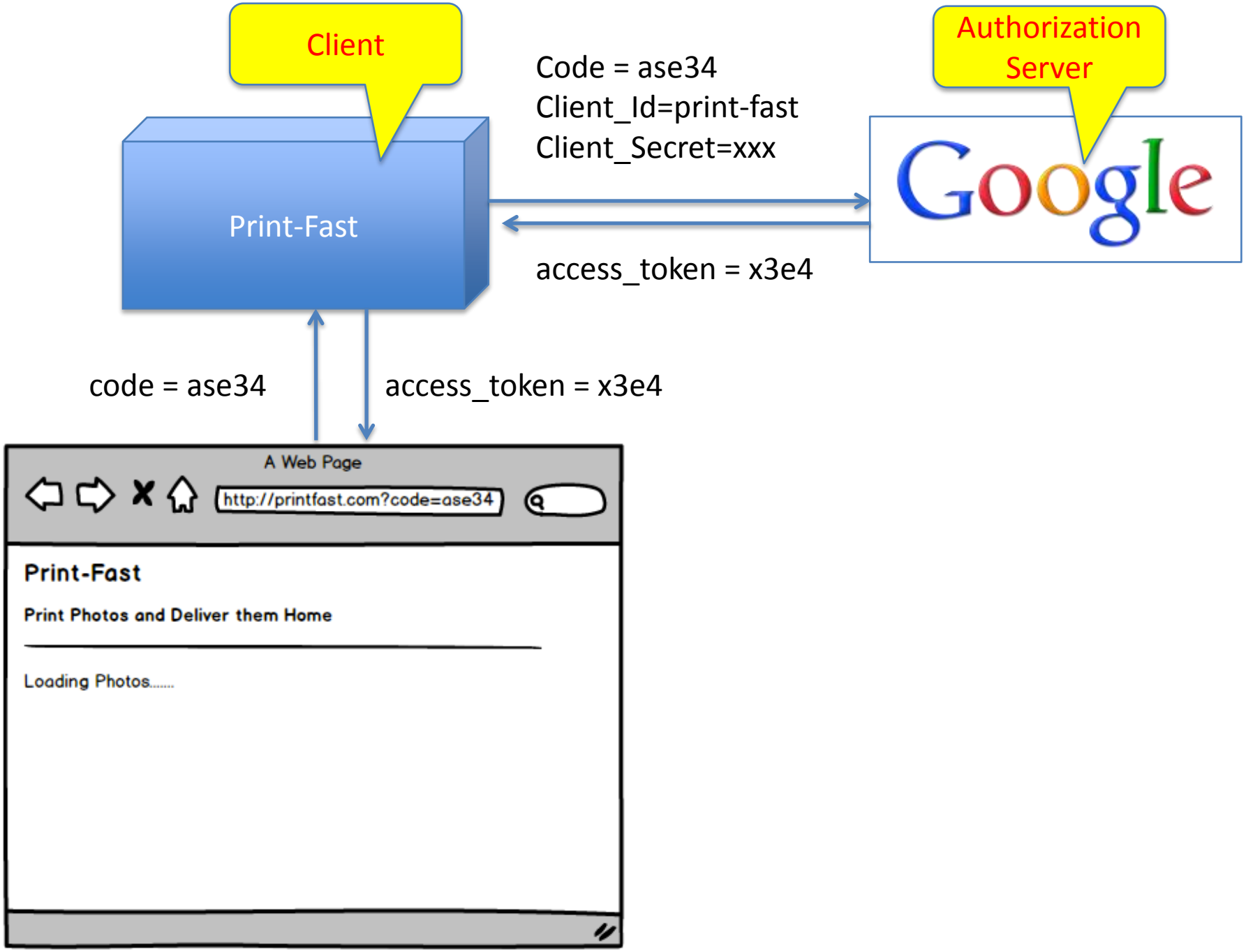
Authorization Grant

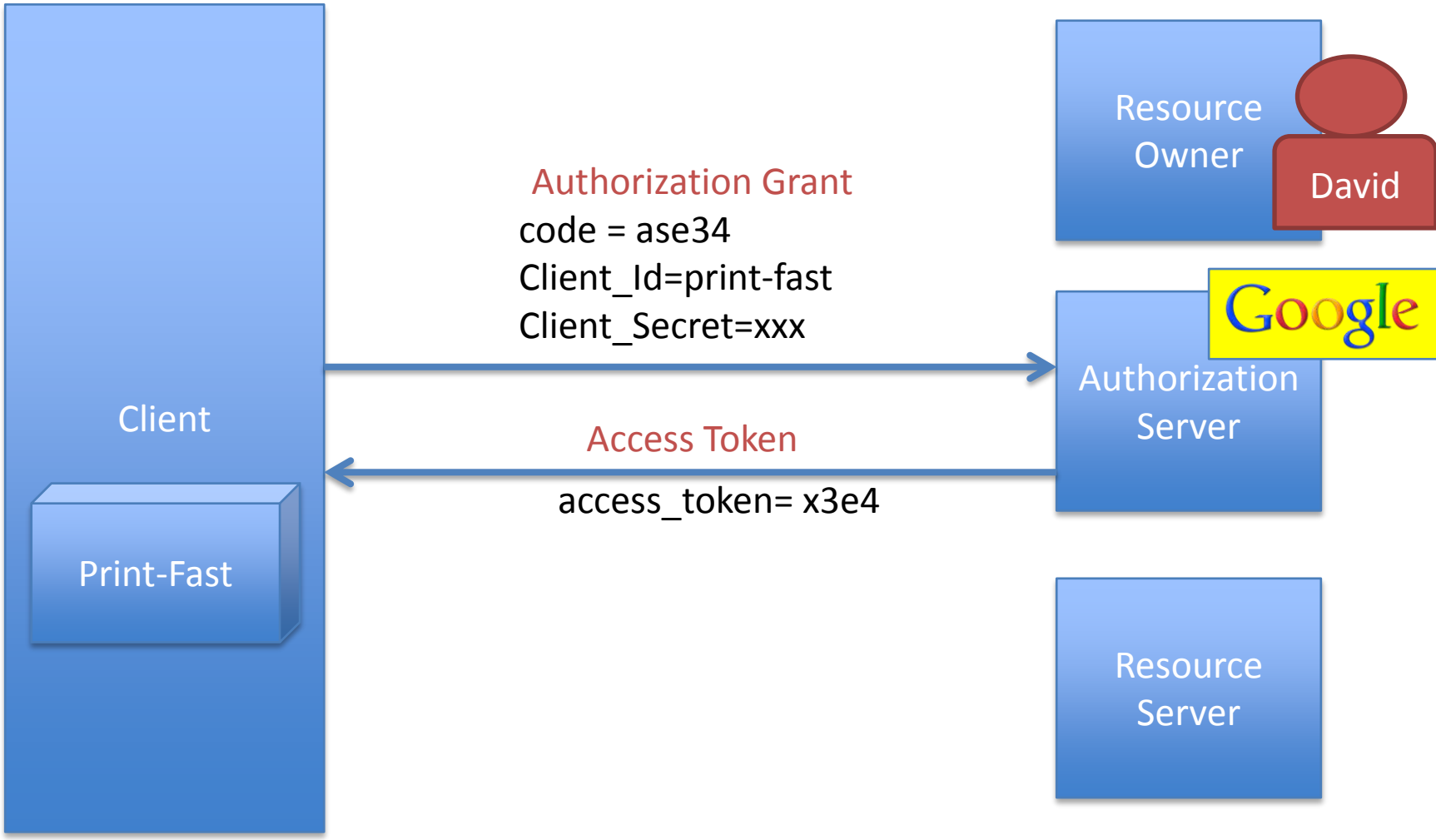




Protocol Flow

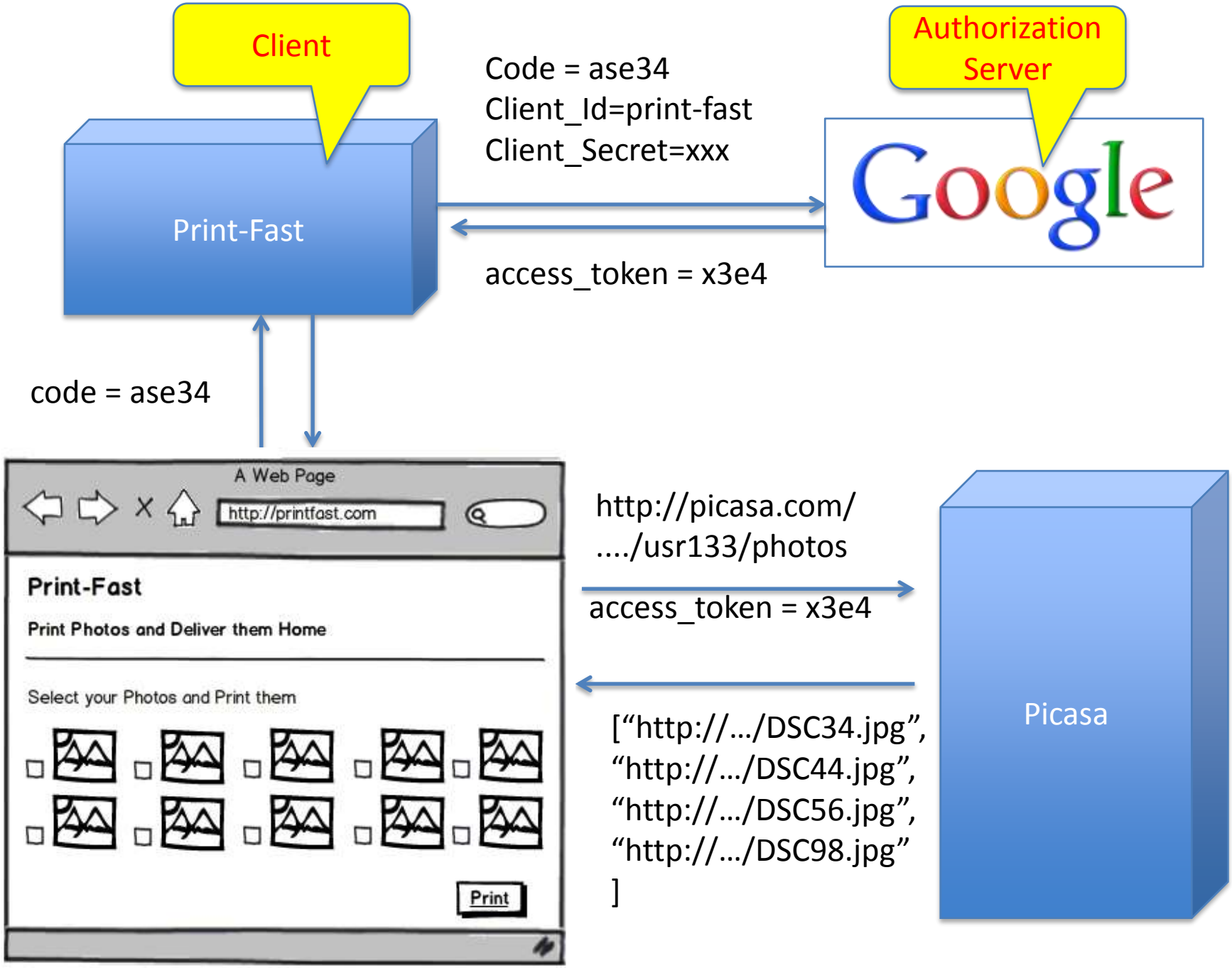
Step 2 – Exchange for Access Token


Client
Print-Fast
Code = ase34
Client_Id=print-fast
Client_Secret=xxx
Authorization Server
Google
access_token = x3e4
code = ase34
access_token = x3e4
A Web Page
http://printfast.com?code=ase34
Print-Fast
Print Photos and Deliver them Home
Loading Photos.....



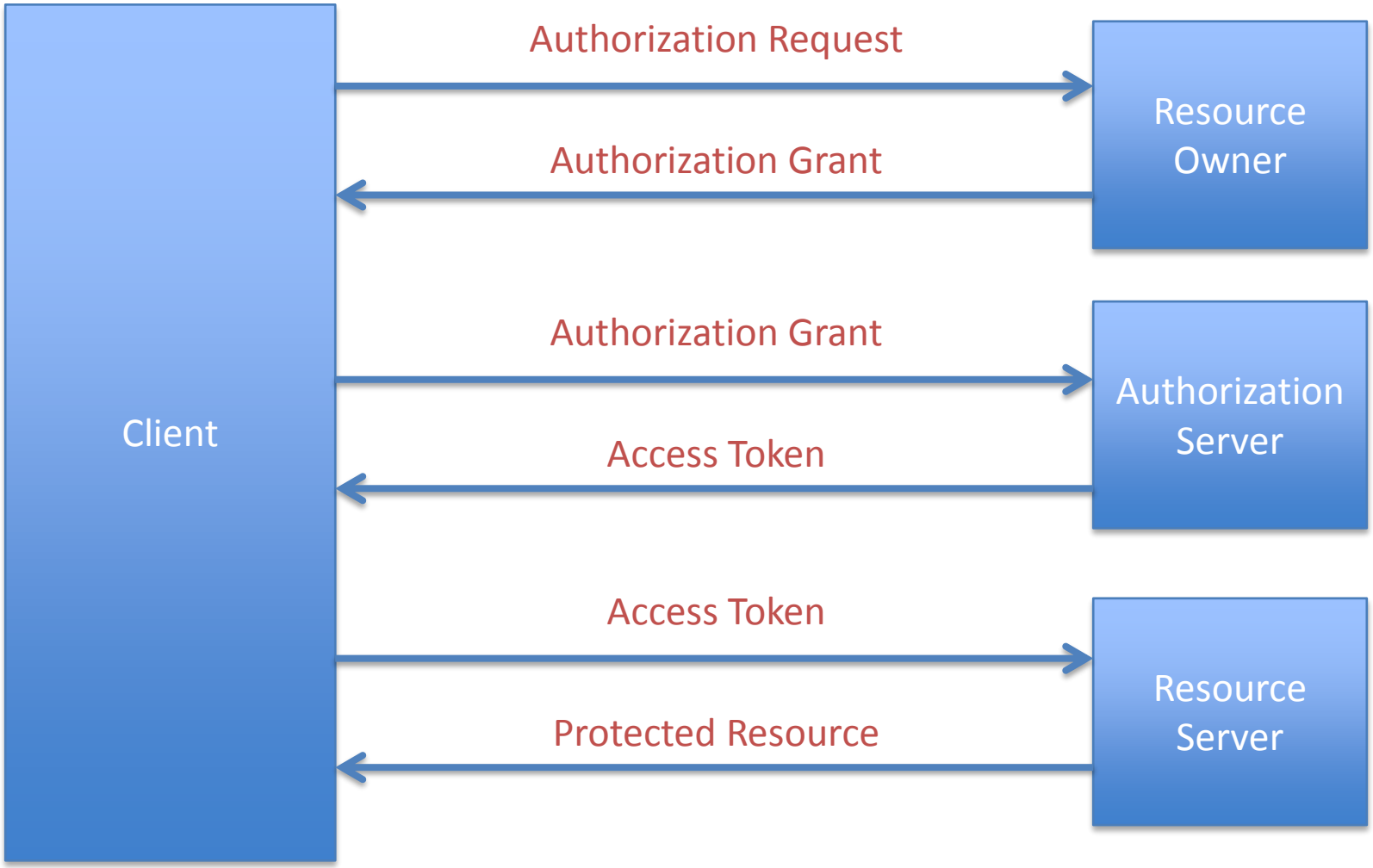
Protocol Flow

Step 3 – Access Protected Resources


Client
Print-Fast
Code = ase34
Client_Id=print-fast
Client_Secret=xxx
Authorization Server
Google
access_token = x3e4
code = ase34
A Web Page
http://printfast.com
Print-Fast
Print Photos and Deliver them Home
Select your Photos and Print them
Print
http://picasa.com/
..../usr133/photos
access_token = x3e4
["http://.../DSC34.jpg",
"http://.../DSC44.jpg",
"http://.../DSC56.jpg",
"http://.../DSC98.jpg"
]
Picasa

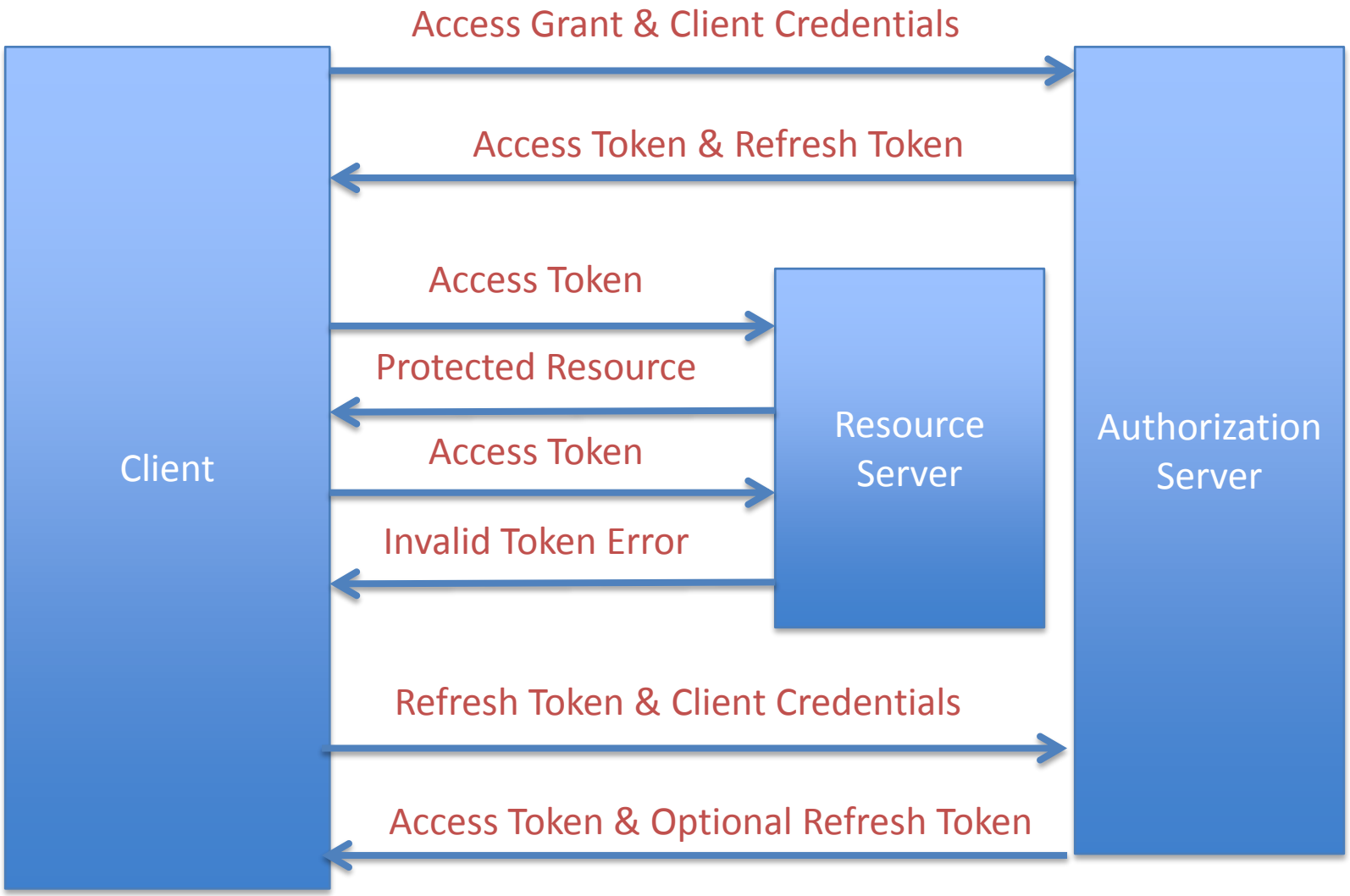
Protocol Flow

Complete Flow at Once



Protocol Flow

With Refresh Token



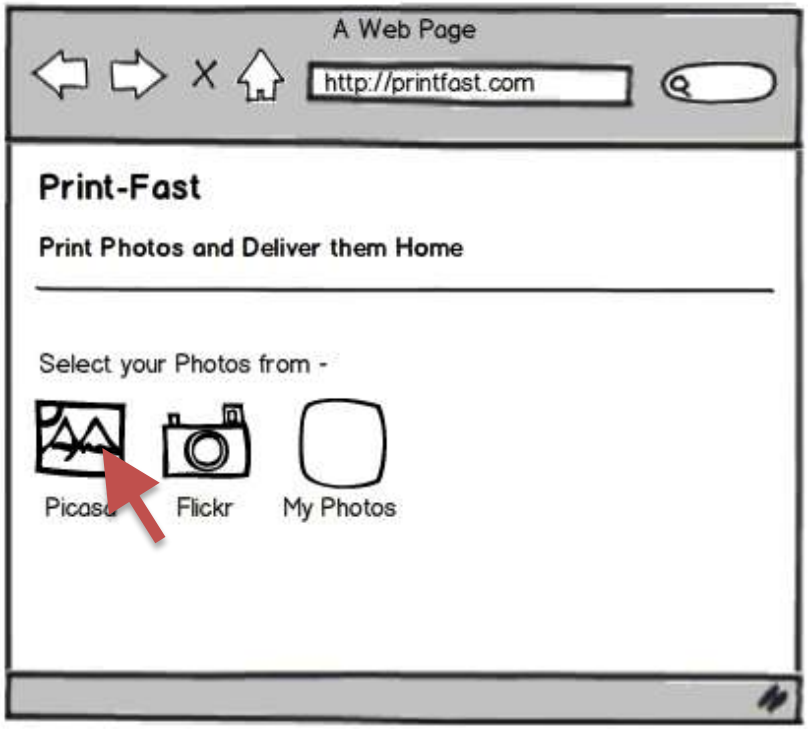
Protocol Flow

OAuth Flows/Grant Types

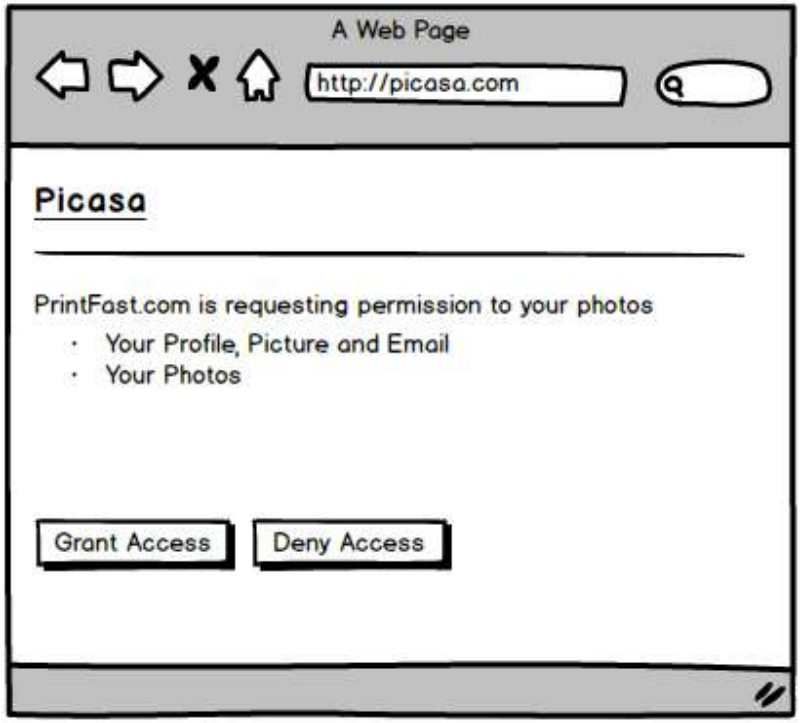
- Authorization Code Grant
- Implicit Grant
- Resource Owner Password Credentials Grant
- Client Credentials Grant

Step 1 – Get Access Token

Implicit Grant Request

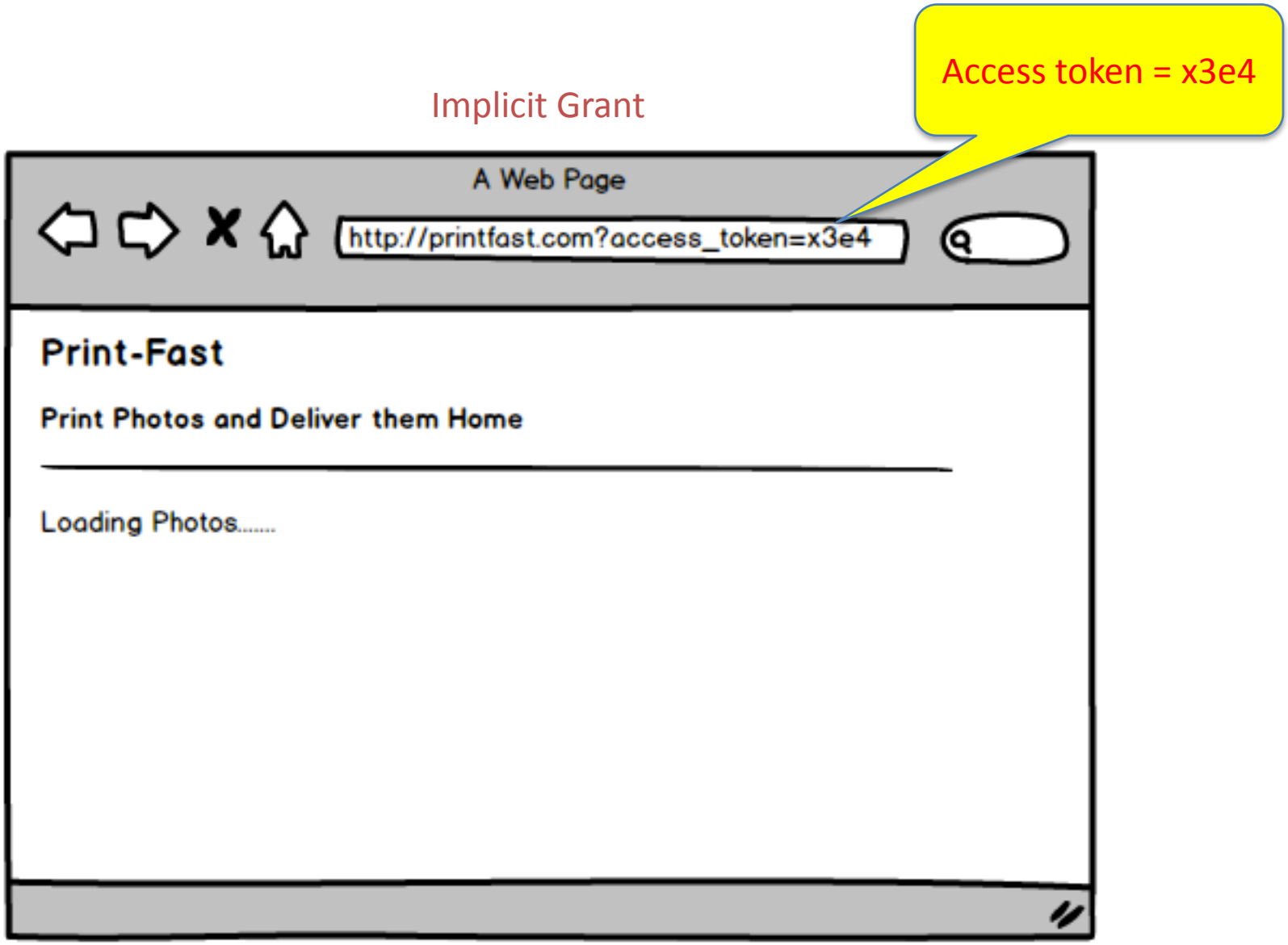


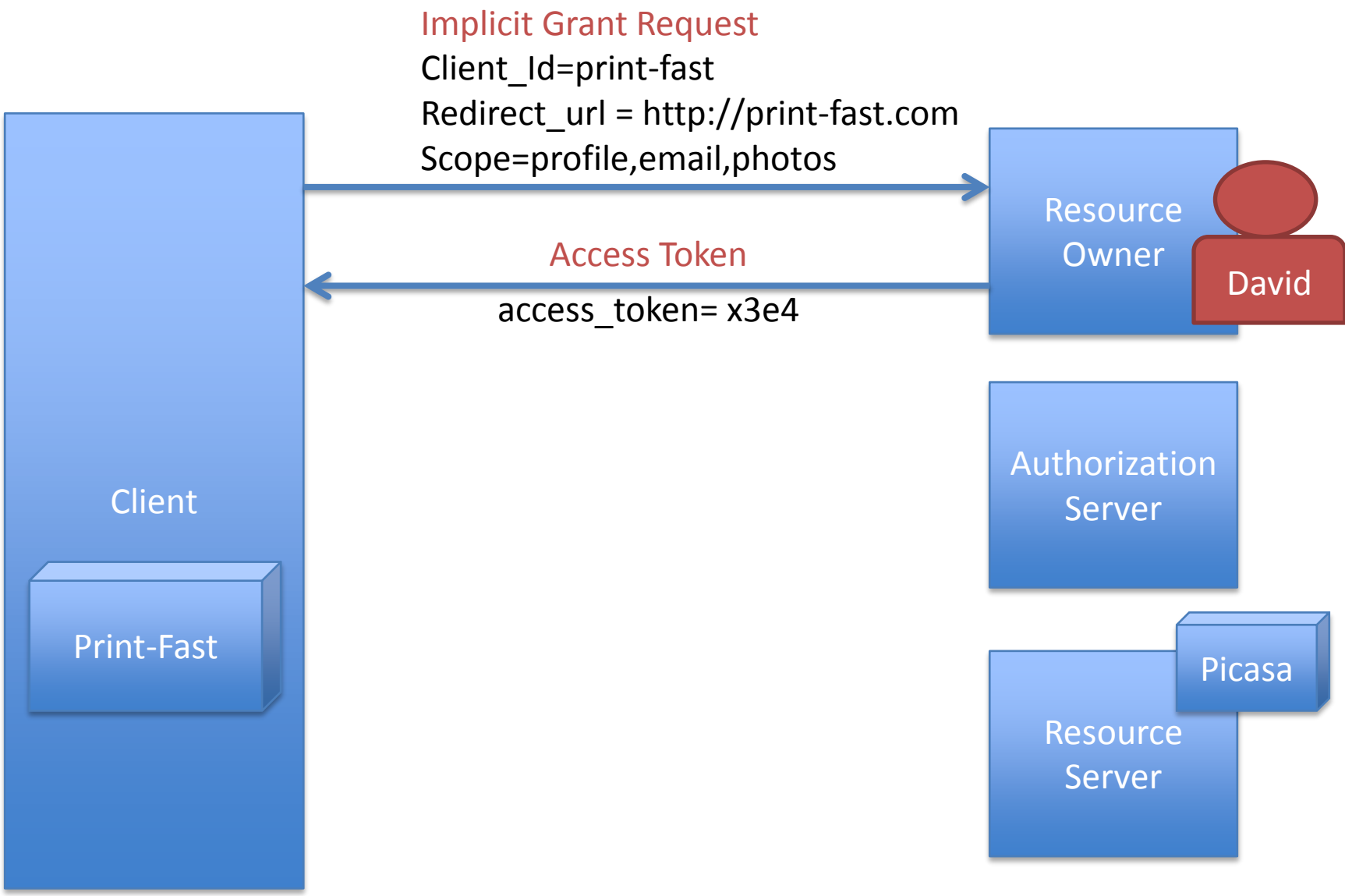
Implicit Grant



URL used is
http://picasa.com/?client_id=photo-fast &scope=profile,email,photos
&redirect_uri=http://print-fast.com&response_type=token

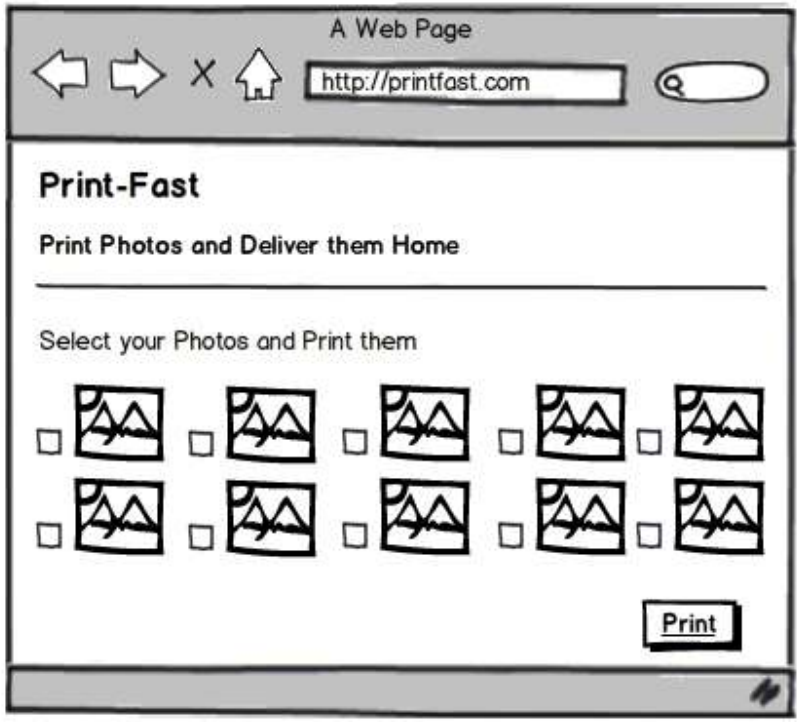
Implicit Grant





Protocol Flow

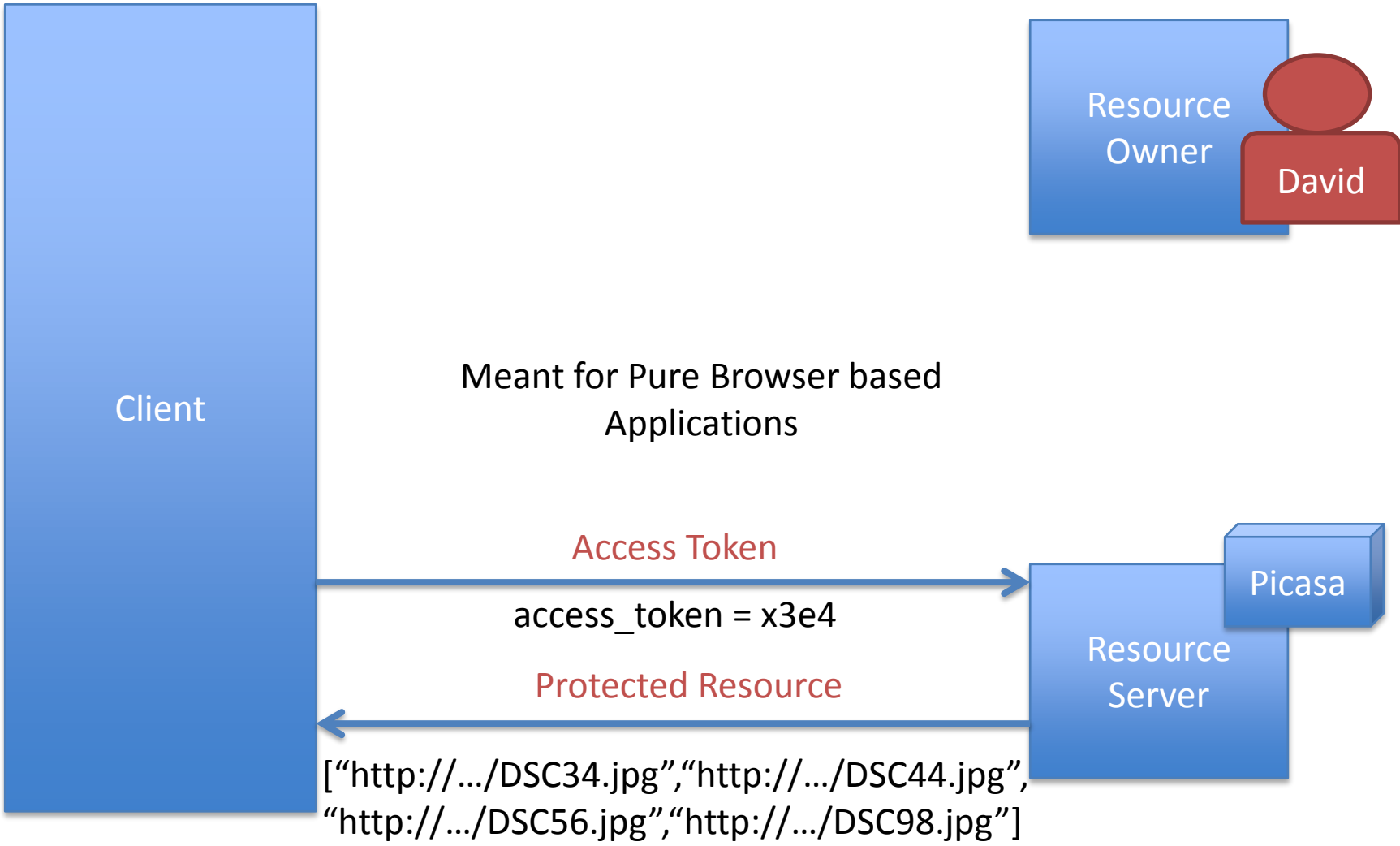
Step 2 – Access Protected Resources



http://picasa.com/
..../usr133/photos

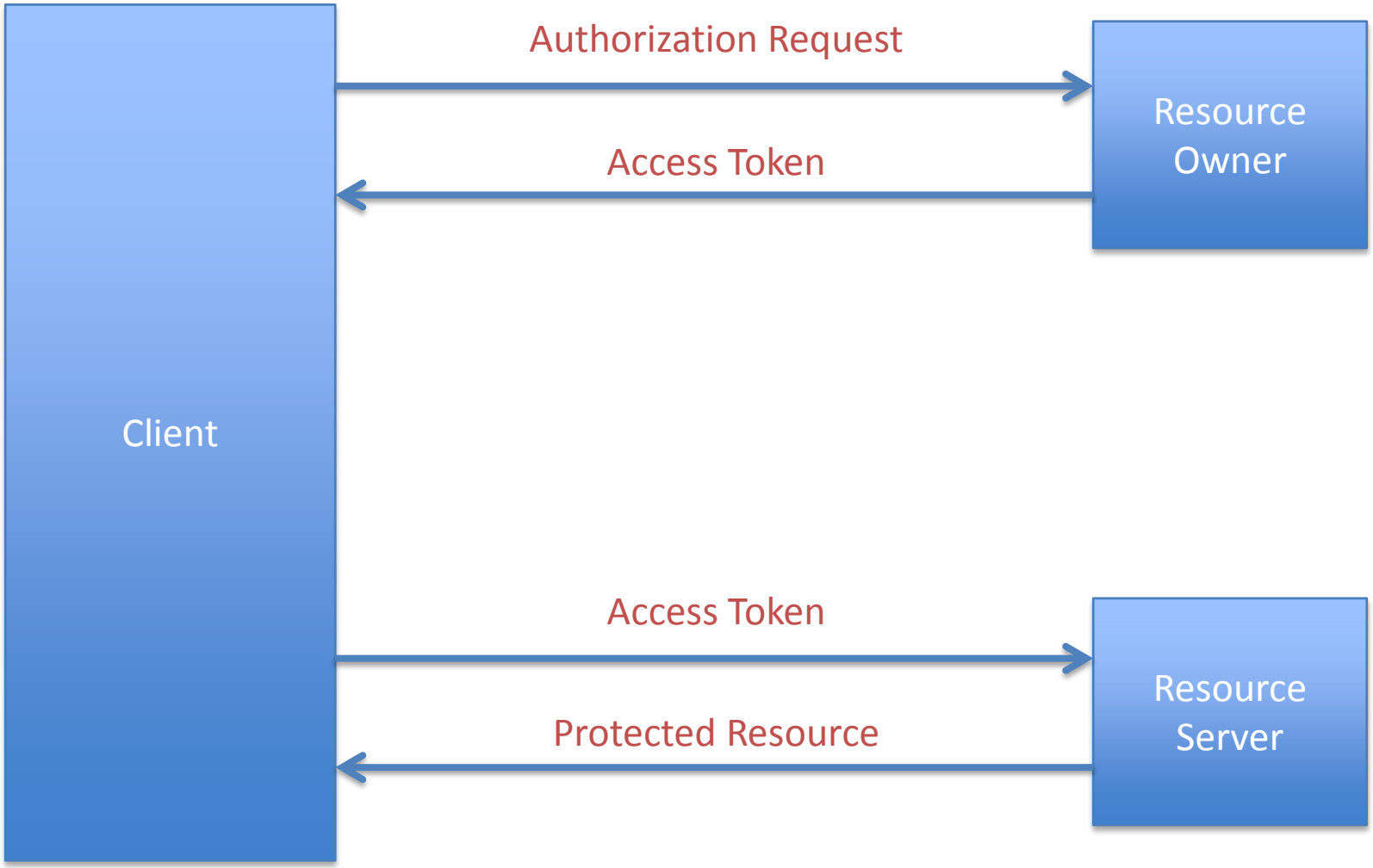
access_token = x3e4

["http://.../DSC34.jpg",
"http://.../DSC44.jpg",
"http://.../DSC56.jpg",
"http://.../DSC98.jpg"
]



Protocol Flow

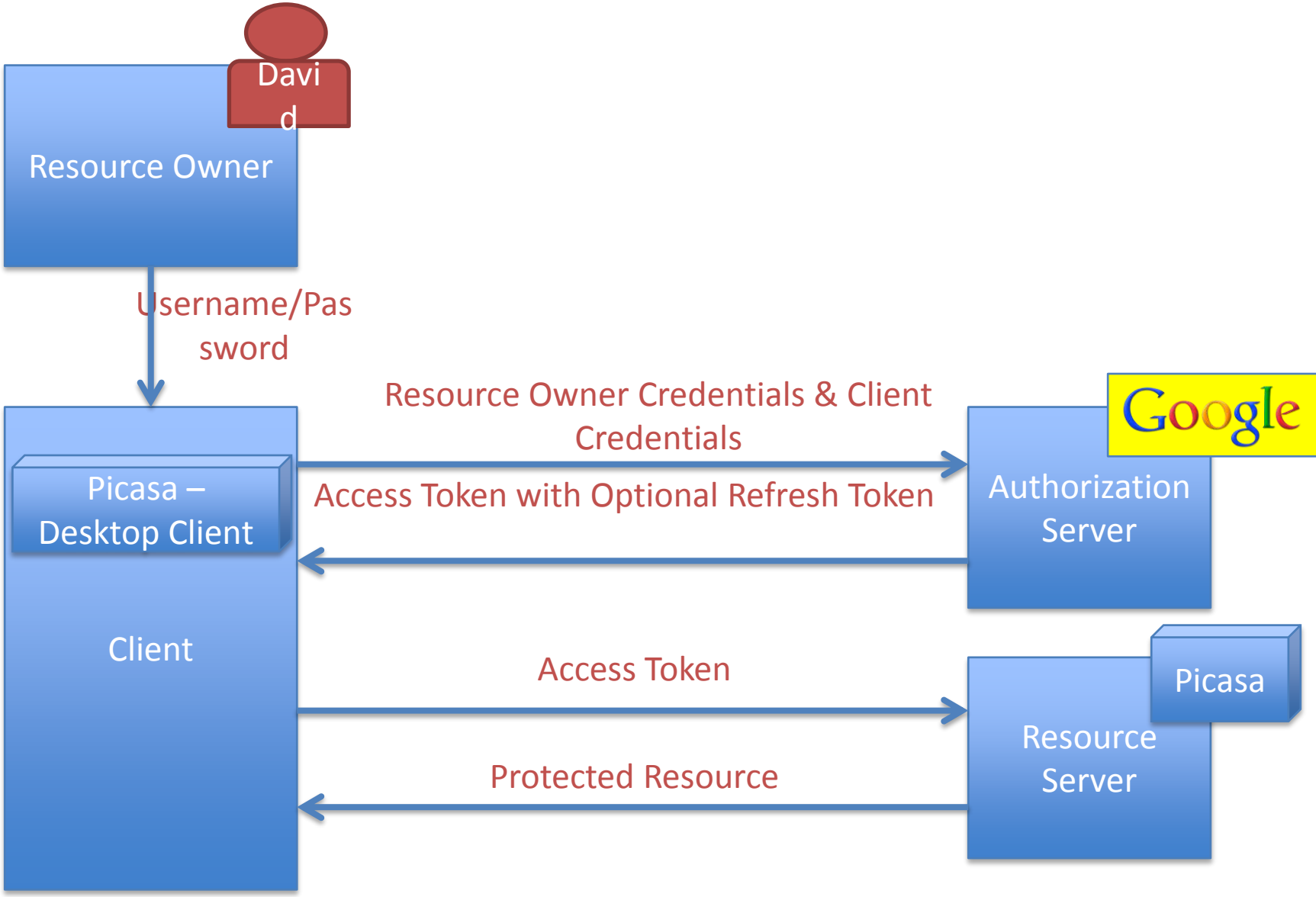
Complete Flow at Once



Protocol Flow

OAuth Flows/Grant Types

- Authorization Code Grant
- Implicit Grant
- Resource Owner Password Credentials Grant
- Client Credentials Grant



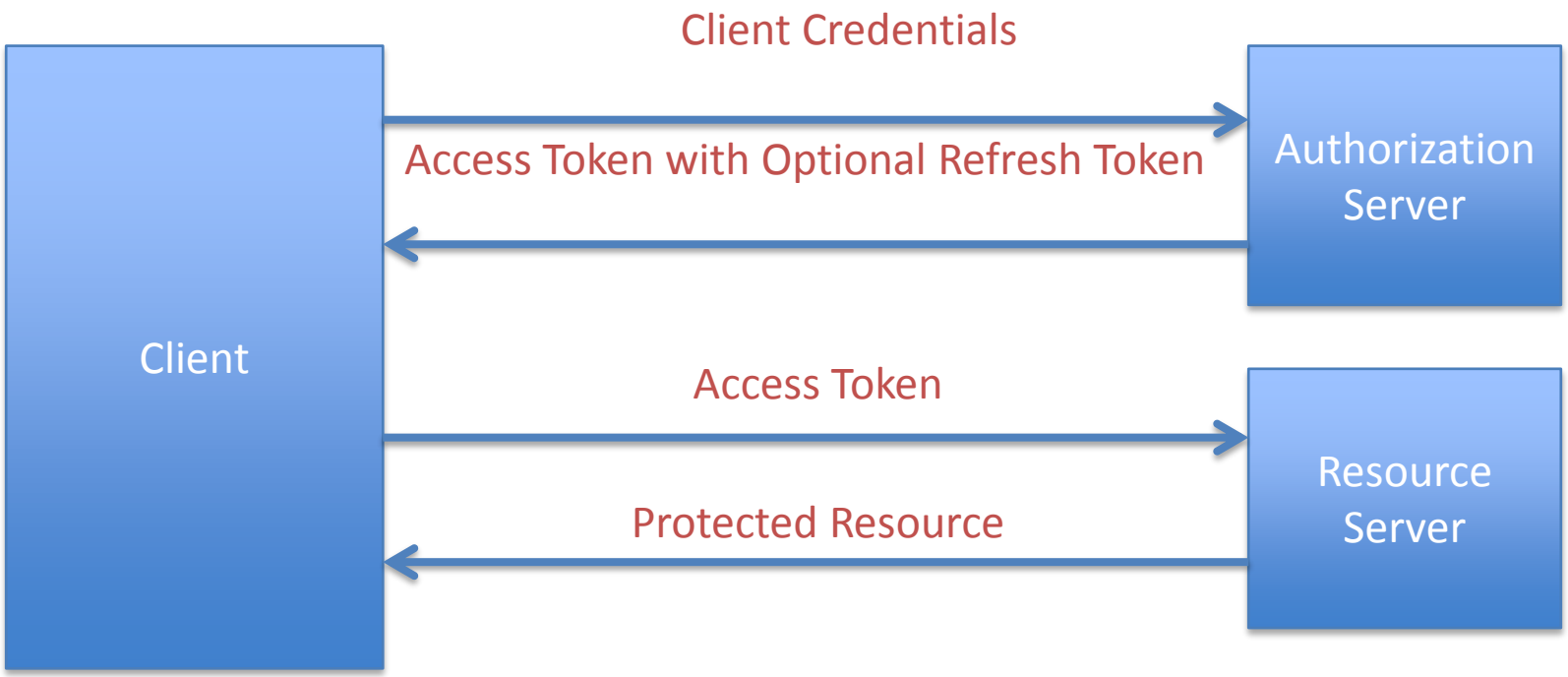
Protocol Flow

Use Cases

- Strong Trust between Resource Owner and Client e.g Operating System or Privileged App
- Client is not supposed to store the Credentials but only the Access token and Refresh Token if provided
- Example – Salesforce OAuth has provision for this

Grant Types

- Authorization Code Grant
- Implicit Grant
- Resource Owner Password Credentials Grant
- Client Credentials Grant



Protocol Flow

Use case

- The Data accessed is not owned by Resource Owner, but by the Client
- Say Skype showing statistics of uptime of its services

Use case

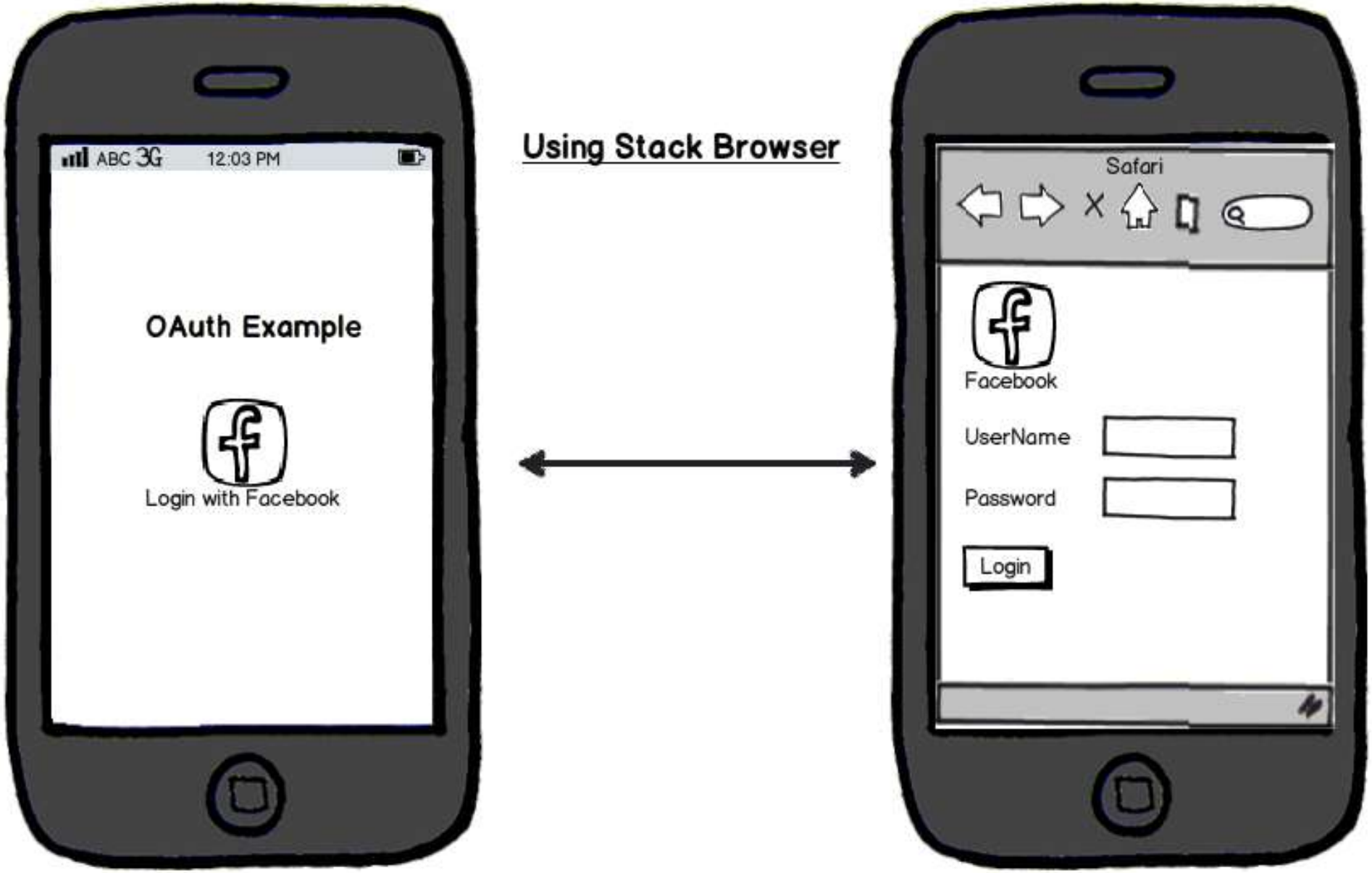
- There is contract already set between the Client and the Authorization Server
- E.g Google Apps Marketspace

- An App installed on Google Apps requires permission to everyone’s calendar in that domain. This permission is provided by the admin and not the end user.

OAuth from Mobile Device

Popular Approaches

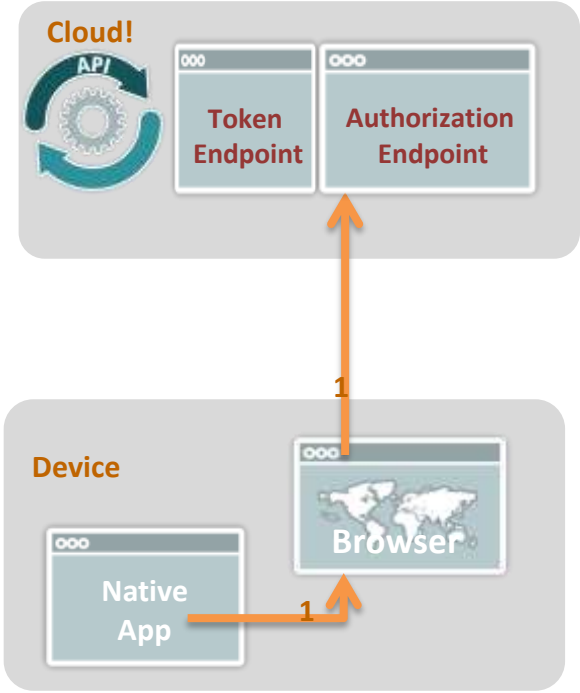
- Using User Agent (Stock Browser)
- Using Embedded WebView


Using Stack Browser
ABC 3G
12:03 PM
OAuth Example
Login with Facebook
Safari
Facebook
UserName
Password
Login

Disclaimer

- Following slides are extracted from http://www.slideshare.net/briandavidcampbell/is-that-a-token-in-your-phone-in-your-pocket-or-are-you-just-glad-to-see-me-oauth-20-and-mobile-devices
- I have no claim on the following slides with reference stated in them
- Thank you Brian Campbell for the excellent presentation

Request Authorization



- When user first needs to access some protected resource, client opens a browser and sends user to the authorization endpoint

```
https://as.example.com/as/authorization.oauth2?client_id=myapp&response_type
=code&scope=update_status
```

http://

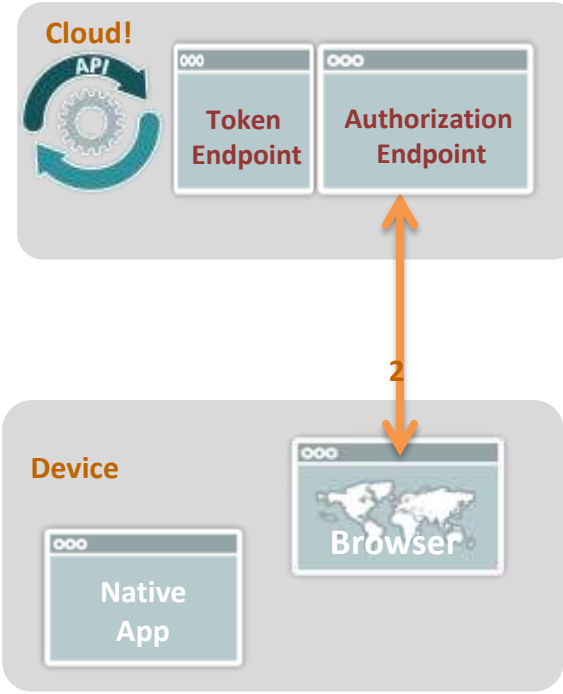
```
Uri authzUrl =
Uri.parse("https://as.example.com/as/authorization.oauth2?client_id=myapp&response_type=code&scope=update_
atus");

Intent launchBrowser = new Intent(Intent.ACTION_VIEW, authzUrl);

startActivity(launchBrowser);
```

```
NSString* launchUrl =
@"https://as.example.com/as/authorization.oauth2?client_id=myapp&response_type=code&scope=update_st

[[UIApplication sharedApplication] openURL:[NSURL URLWithString: launchUrl]];
```

Reference - http://www.slideshare.net/briandavidcampbell/is-that-a-token-in-your-phone-in-your-pocket-or-are-you-just-glad-to-see-me-oauth-20-and-mobile-devices

Authenticate and Approve

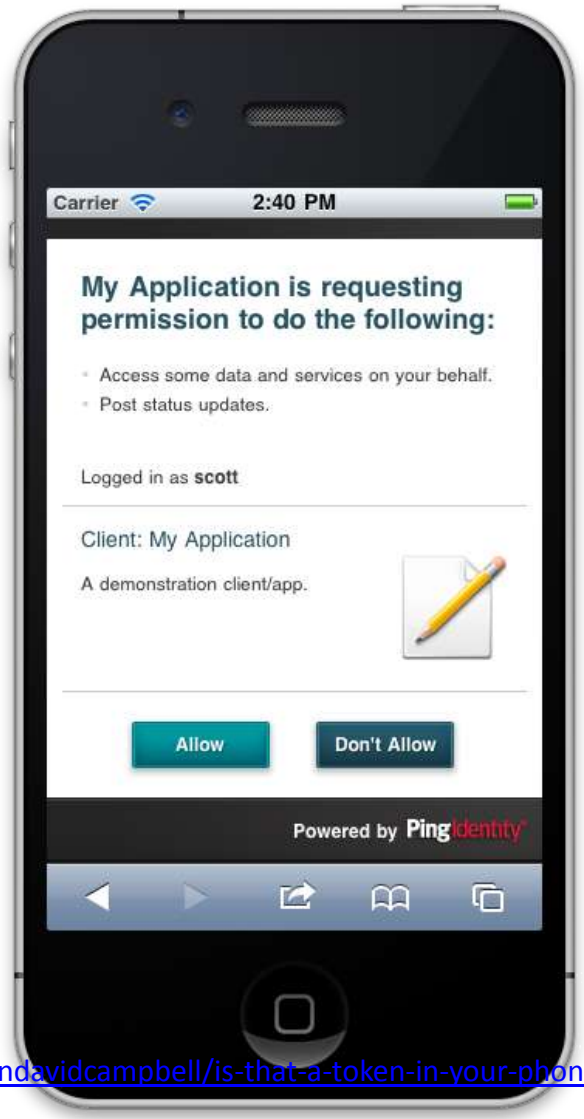
- The AS authenticates the user
  - Directly
  - Indirectly via Facebook, Twitter, Google, Yahoo, etc.

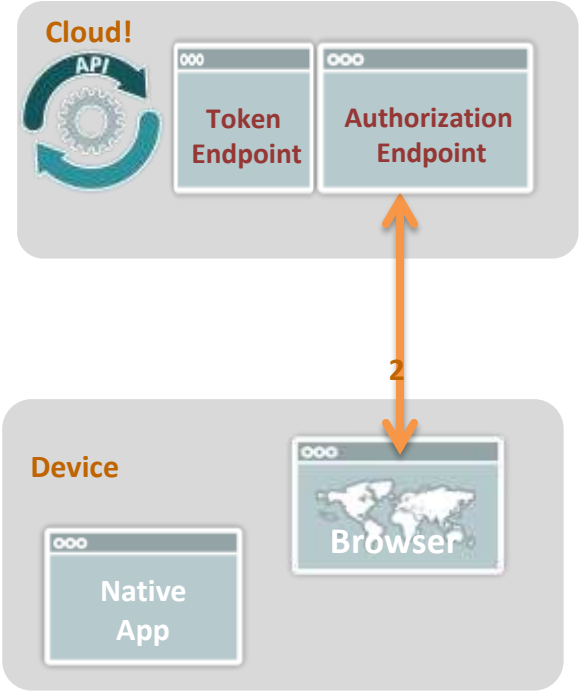


Reference - http://www.slideshare.net/briandavidcampbell/is-that-a-token-in-your-phone-in-your-pocket-or-are-you-just-glad-to-see-me-oauth-20-and-mobile-devices

Approve

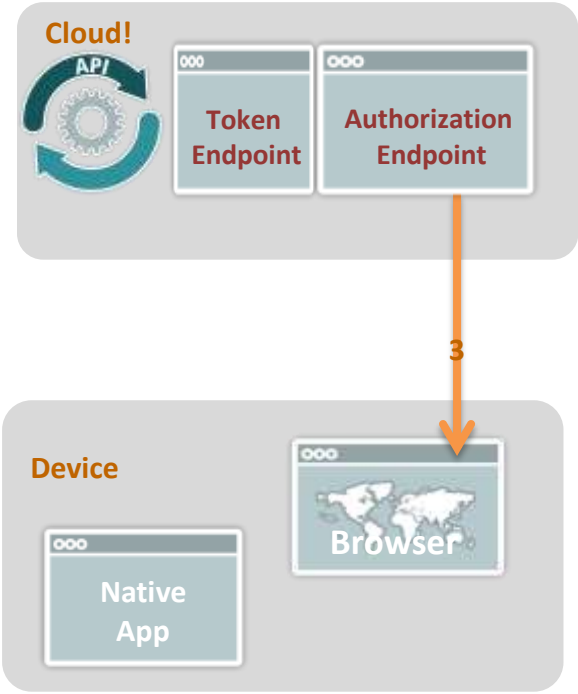
- User approves the requested access





Reference - http://www.slideshare.net/briandavidcampbell/is-that-a-token-in-your-phone-in-your-pocket-or-are-you-just-glad-to-see-me-oauth-20-and-mobile-devices

Handle Callback



Server returns control to the app via HTTP redirection and includes an authorization code

```
HTTP/1.1 302 Found
Location: x-com.mycorp.myapp://oauth.callback?code=SplxlOBeZQQYbYS6WxSbIA
```

http://

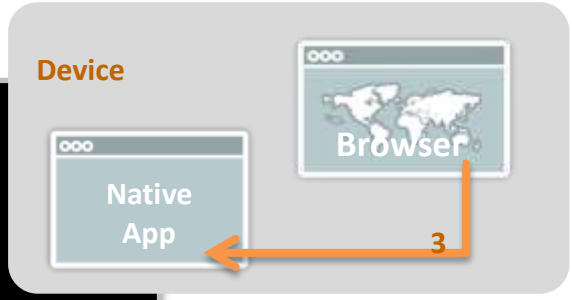
Reference - http://www.slideshare.net/briandavidcampbell/is-that-a-token-in-your-phone-in-your-pocket-or-are-you-just-glad-to-see-me-oauth-20-and-mobile-devices

Handle Callback (cont'd)

Registering a custom URI scheme

In AndroidManifest.xml file:

```
<activity android:name=".MyAppCallback" … >
 <intent-filter>
  <action android:name="android.intent.action.VIEW"/>
  <category android:name="android.intent.category.DEFAULT"/>
  <category android:name="android.intent.category.BROWSABLE"/>
  <data android:scheme="x-com.mycorp.myapp" />
 </intent-filter>
</activity>
```

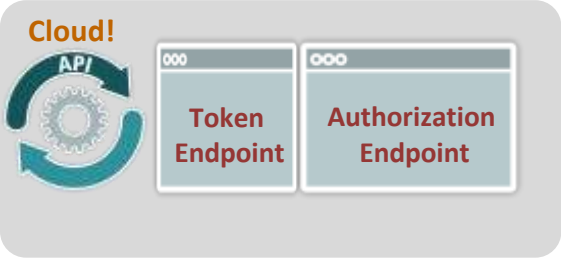



```
String authzCode = getIntent().getData().getQueryParameter("code");
```

Reference - http://www.slideshare.net/briandavidcampbell/is-that-a-token-in-your-phone-in-your-pocket-or-are-you-just-glad-to-see-me-oauth-20-and-mobile-devices

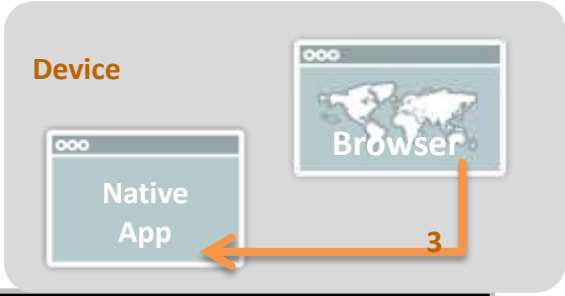
Handle Callback (cont'd)

Registering a custom URI scheme



In app info plist file:

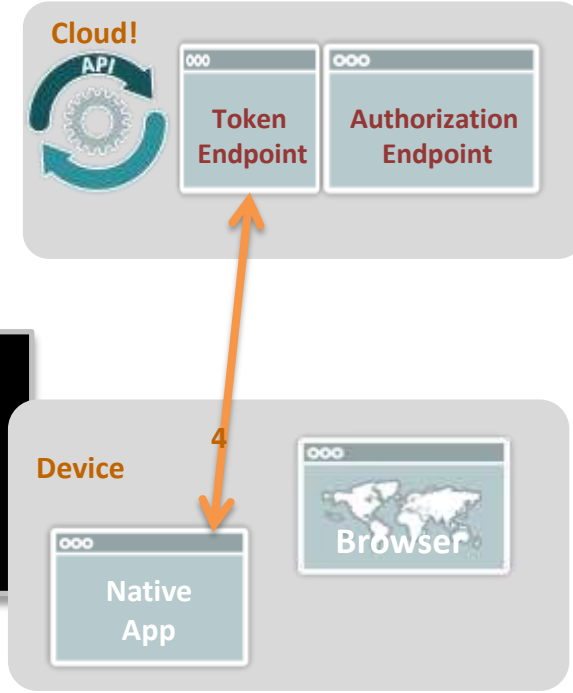
▼ URL types	(1 item)
▼ Item 0	(2 items)
URL identifier	www.mycorp.com
▼ URL Schemes	(1 item)
Item 0	x-com.mycorp.myapp




```
- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url
{
    NSString *queryString = [url query];
    NSMutableDictionary *qsParms = [[NSMutableDictionary alloc] init];
    for (NSString *param in [queryString componentsSeparatedByString:@"&"]) {
        NSArray *elts = [param componentsSeparatedByString:@"="];
        if([elts count] < 2) continue;
        [qsParms setObject:[elts objectAtIndex:1] forKey:[elts objectAtIndex:0]];
    };

    NSString *code = [qsParms objectForKey:@"code"];
...
```

Reference - http://www.slideshare.net/briandavidcampbell/is-that-a-token-in-your-phone-in-your-pocket-or-are-you-just-glad-to-see-me-oauth-20-and-mobile-devices

Trade Code for Token(s)



Token Endpoint Request

```
POST /as/token.oauth2 HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8

client_id=myapp&grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
```

http://

Token Endpoint Response

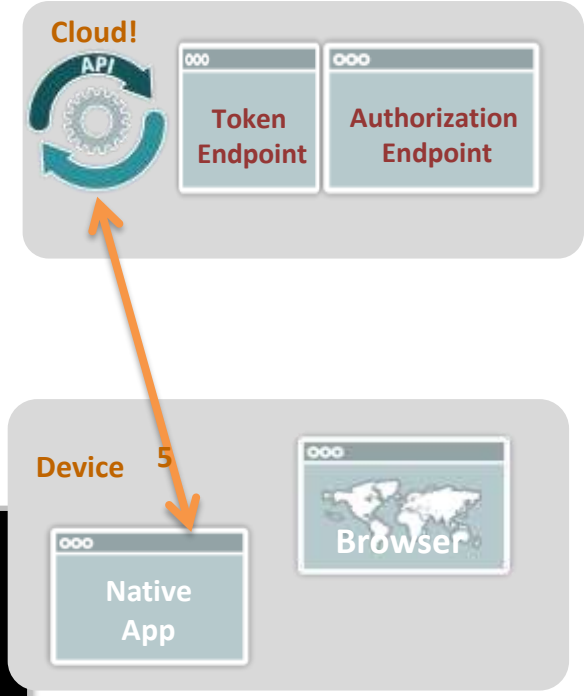
```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
 "token_type":"Bearer",
 "expires_in":3600,
 "access_token":"PeRTSD9RQrbiuoaHVPxV41MzW1qS”,
 "refresh_token":"uyAVrtyLZ2qPzI8rQ5UUTckCdGaJsz8XE8S58ecnt8”
}
```

http://

Reference - http://www.slideshare.net/briandavidcampbell/is-that-a-token-in-your-phone-in-your-pocket-or-are-you-just-glad-to-see-me-oauth-20-and-mobile-devices

Using an Access Token

- Once an access token is obtained, it can be used to authenticate/authorize calls to the protected resources at the RS by including it in HTTP Authorization header




```
POST /api/update-status HTTP/1.1
Host: rs.example.com
Authorization: Bearer PeRTSD9RQrbiuoaHVPxV41MzW1qS
Content-Type: application/x-www-form-urlencoded;charset=UTF-8

status=Almost%20done.
```

http://

```
NSString *authzHeader = [NSString stringWithFormat:@"Bearer %@", accessToken];

NSMutableURLRequest *request = [[[NSMutableURLRequest alloc] init] autorelease];
[request setURL:[NSURL URLWithString:@"https://rs.example.com/api/update-status"]];
[request setValue:authzHeader forHTTPHeaderField:@"Authorization"];
```

iOS

```
DefaultHttpClient httpClient = new DefaultHttpClient();
HttpPost post = new HttpPost("https://rs.example.com/api/update-status");
post.setHeader("Authorization", "Bearer " + accessToken);
```

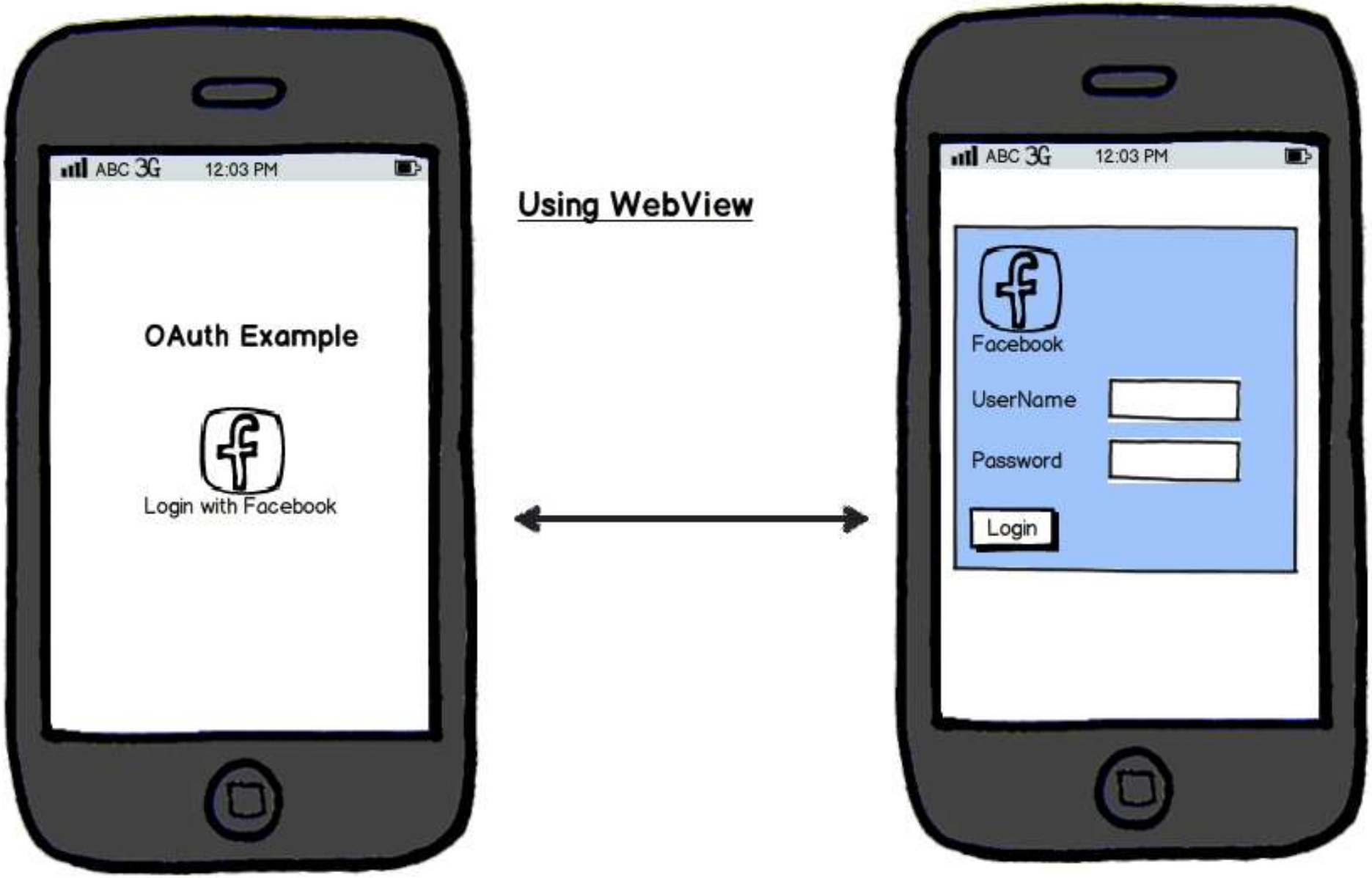
Reference - http://www.slideshare.net/briandavidcampbell/is-that-a-token-in-your-phone-in-your-pocket-or-are-you-just-glad-to-see-me-oauth-20-and-mobile-devices

Pros and Cons

- Pros
  - User may be already logged in most cases
  - User will trust as he/she sees https and domain name
- Cons
  - Complicated Custom URI schema

Popular Approaches

- Using User Agent (Stock Browser)
- Using Embedded WebView


Using WebView
ABC 3G
12:03 PM
OAuth Example
Login with Facebook
ABC 3G
12:03 PM
Facebook
UserName
Password
Login

Pros and Cons

- Pros
  - Easier to monitor pages and extract authorization or access codes
- Cons
  - May not appeal since neither https or domain name is visible
  - WebView has separate cookie and history leading to client entering credentials each time

Reference

- Book – Getting Started with OAuth 2.0
- Facebook Documentation
- Google Documentation
- Brian David Campbell's Presentation