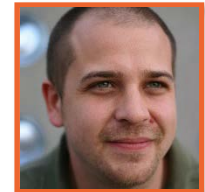


# Token-based Authentication – Part 1

Dominick Baier  
<http://leastprivilege.com>  
@leastprivilege



**pluralsight**   
hardcore dev and IT training

# Agenda

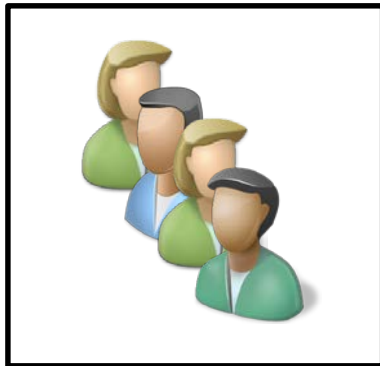
- Mindset and challenges
  - OAuth2
  - Flows
- 
- Using the Katana OAuth2 middleware
  - Using an external authorization server

# Enterprise Security

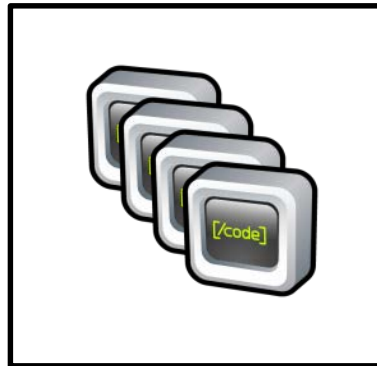


# Modern Applications

## Users



## Clients



## Web APIs

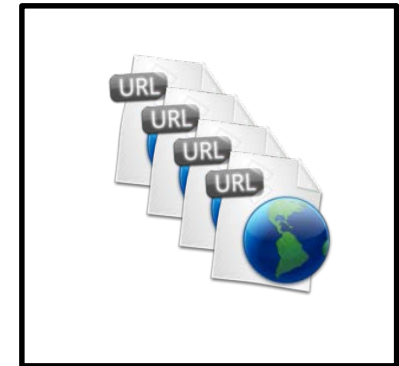
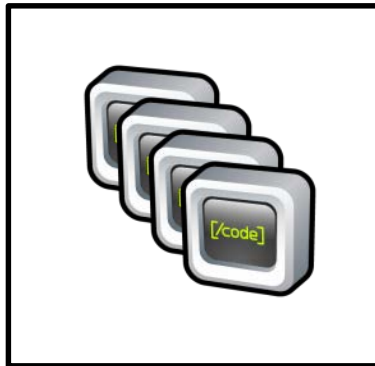


# "Simple" Requirements

- Which app do I want to use today?
- Do I trust this app?

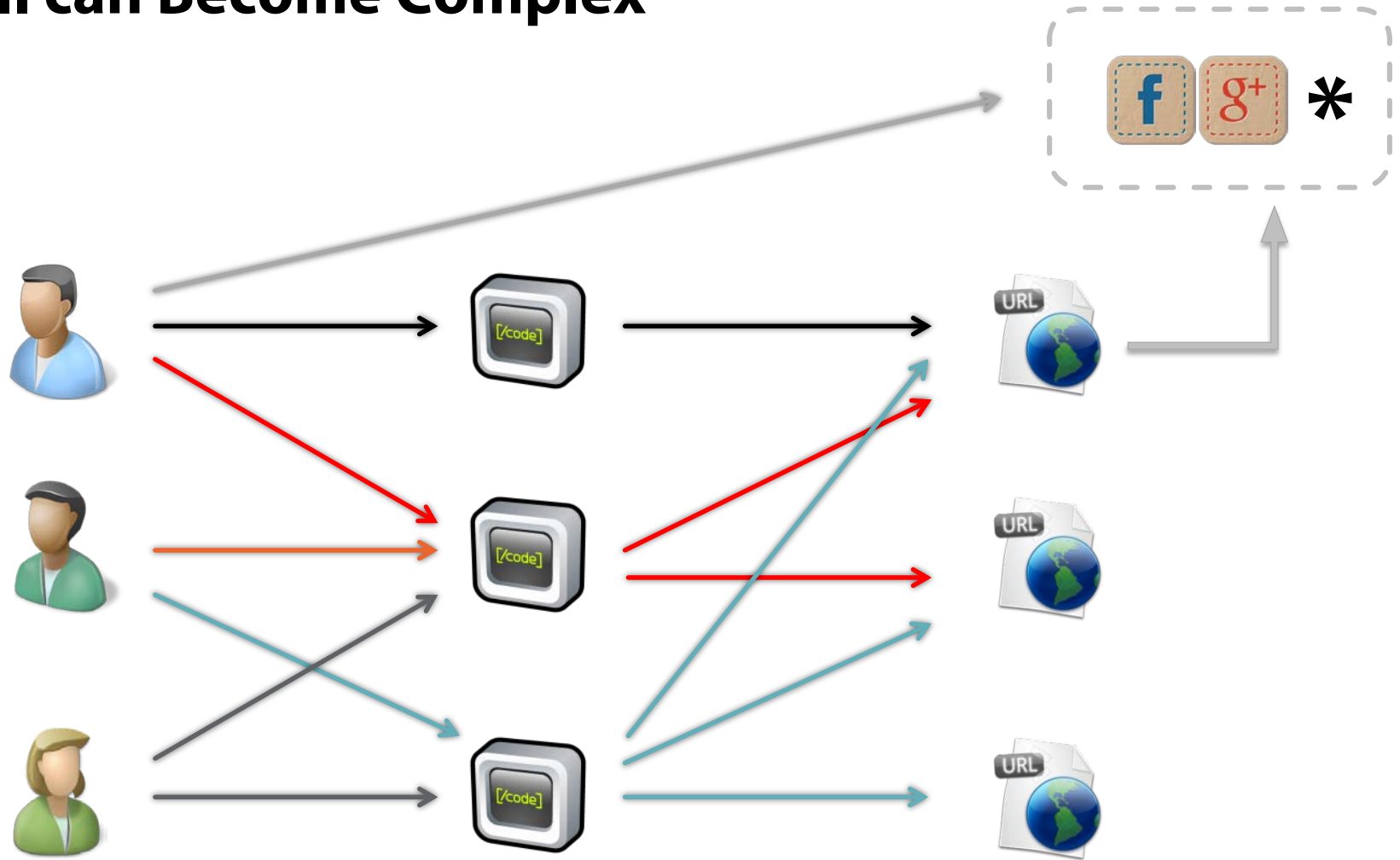


- Who is the user?
- Who is the client?
- What are they allowed to do?



- Who is the user?
- How can I securely communicate with the back-end

## ...Still can Become Complex



# OAuth2



OAUTH

[About](#) [Advisories](#) [Documentation](#) [Code](#) [Community](#)

An **open protocol** to allow **secure authorization** in a **simple** and **standard** method from web, mobile and desktop applications.

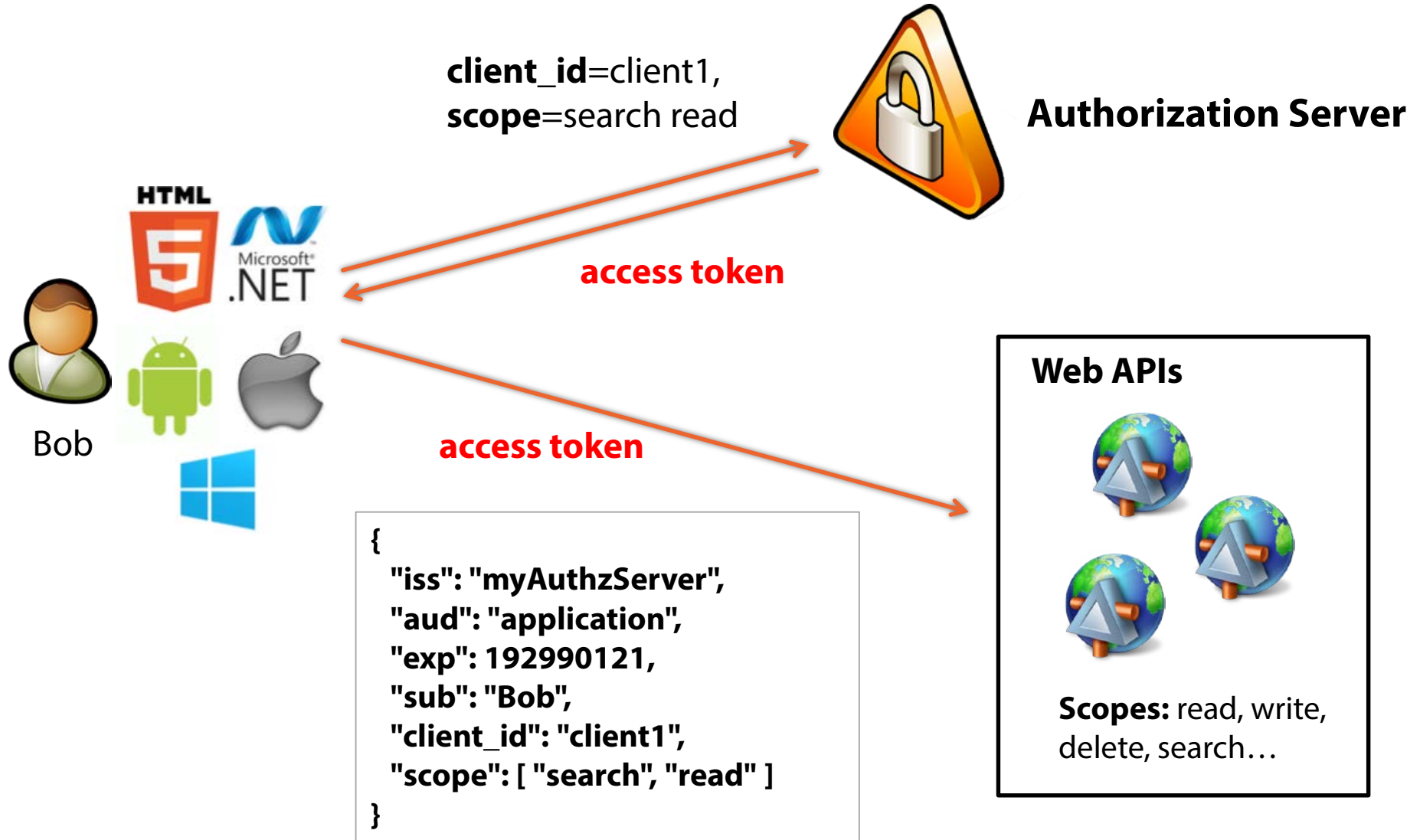
[Read the OAuth 2 specification »](#)

## The OAuth 2.0 Authorization Framework

### Abstract

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. This specification replaces and obsoletes the OAuth 1.0 protocol described in [RFC 5849](#).

# OAuth2 approach





# Flows

- **Patterns for orchestrating communication between client and authorization server**
  - server-rendered web applications
  - user-agent based web applications
  - native applications
  - machine-to-machine communication
  - federation
- **Ability to treat the client as partially trusted**
  - as well as client authentication

# thinktecture

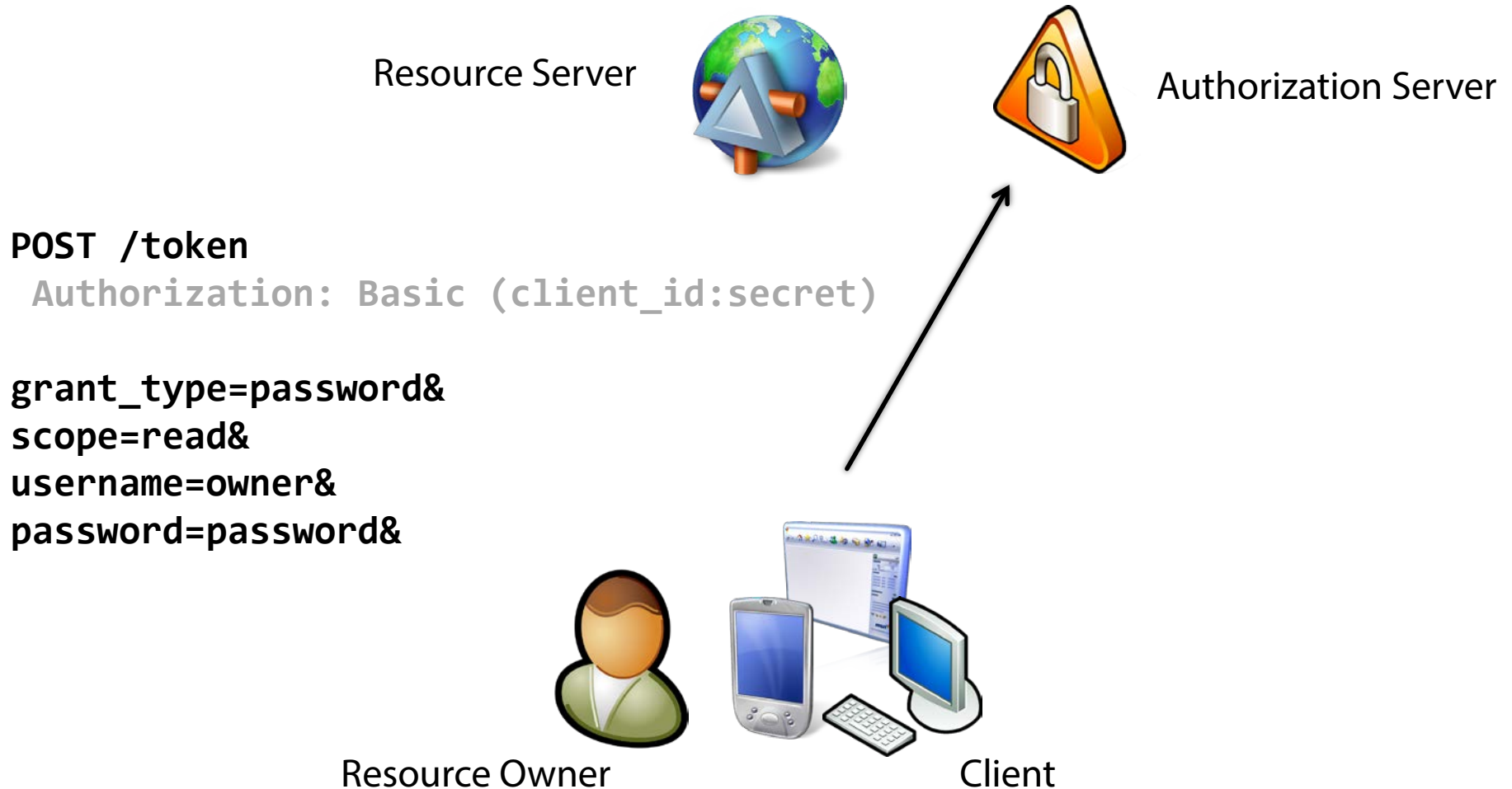
## AuthorizationServer

**<https://github.com/thinktecture/Thinktecture.AuthorizationServer>**

# Starting Simple...

- **“Trusted clients”**
  - Resource Owner Password Credential Flow

# Step 1a: Token Request



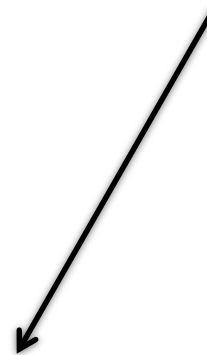
# Step 1b: Token Response

Resource Server



Authorization Server

```
{  
  "access_token" : "abc",  
  "expires_in" : "3600",  
  "token_type" : "Bearer",  
  "refresh_token" : "xyz"  
}
```



Resource Owner



Client

## Step 2: Use Token



# Server-side

```
public void Configuration(IAppBuilder app)
{
    // token generation
    app.UseOAuthAuthorizationServer(new OAuthAuthorizationServerOptions
    {
        TokenEndpointPath = new PathString("/token"),
        AccessTokenExpireTimeSpan = TimeSpan.FromHours(8),

        Provider = new SimpleAuthorizationServerProvider()
    });

    // token consumption
    app.UseOAuthBearerAuthentication(
        new OAuthBearerAuthenticationOptions());

    app.UseWebApi(WebApiConfig.Register());
}
```

# Client-side

- Request token

```
private async Task<string> GetTokenAsync()
{
    var client = new HttpClient();

    var post = new Dictionary<string, string>
    {
        { "grant_type", "password" },
        { "username", "bob" },
        { "password", "bob" }
    };

    var response = await client.PostAsync("http://localhost:2727/token",
        new FormUrlEncodedContent(post));
    var content = await response.Content.ReadAsStringAsync();

    var json = JObject.Parse(content);
    return json["access_token"].ToString();
}
```



# Client-side

- Call Service

```
private async Task<string> CallServiceAsync(string token)
{
    var client = new HttpClient();

    client.DefaultRequestHeaders.Authorization =
        new AuthenticationHeaderValue("Bearer", token);

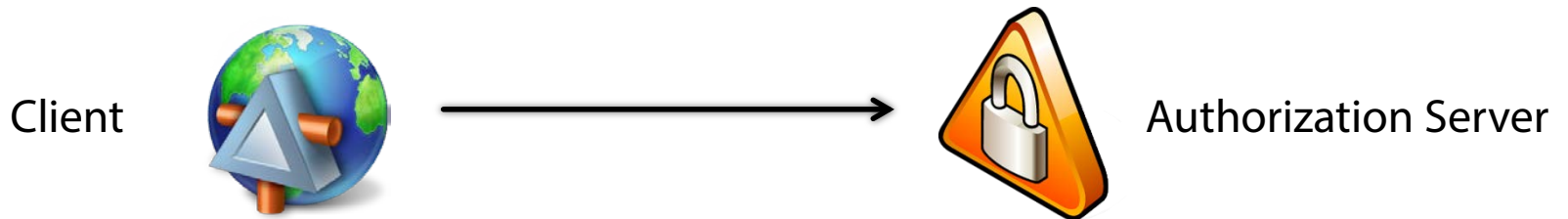
    return await client.GetStringAsync(
        new Uri("http://localhost:2727/api/identity"));
}
```

# Summary

- **Allows exchanging a password with a token**
  - short lived or long lived
- **Better than dealing with passwords directly**
  - e.g. storing the password
  - client still can "see" the password
  - maybe not what you want

# Adding Refresh Tokens

- **Refresh tokens are long lived tokens**
  - are used to periodically refresh the short lived access token
  - allows updating token contents
  - allows revocation

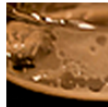


**POST /token**

**Authorization: Basic (client\_id:secret)**

**grant\_type=refresh\_token&  
refresh\_token=xyz**

# Refresh Token Management (Flickr)



**leastprivilege**

[Apps By You](#)

[Apps You're Using](#)

[Your Favorite Apps](#)

Below is a list of applications that you've given permission to interact with your Flickr account. It doesn't include apps that only use public photos and don't need to be authorized.




If you want to stop using one of these apps, click its "Remove permission" link.

Application	Permissions	
<b>Adobe Photoshop Lightroom</b> <a href="http://www.adobe.com/products/photoshoplightroom/">http://www.adobe.com/products/photoshoplightroom/</a>	delete	<a href="#">Remove permission?</a>
<b>Flickr for Windows Phone 7</b> <a href="http://social.zune.net/redirect?type=phoneApp&amp;id=2e49fb07-592b-e011-854c-00237de2db9e">http://social.zune.net/redirect?type=phoneApp&amp;id=2e49fb07-592b-e011-854c-00237de2db9e</a>	delete	<a href="#">Remove permission?</a>
<b>Photorank.me</b>	read	<a href="#">Remove permission?</a>
<b>Microsoft</b> <a href="http://aka.ms/flickr">http://aka.ms/flickr</a>	write	<a href="#">Remove permission?</a>

# Refresh Token Management (Dropbox)

## My apps

You have given these apps access to your Dropbox account.

App name	Publisher	Access type	
 1Password	<a href="#">AgileBits</a>	Full Dropbox	×
 1Password for Android	<a href="#">AgileWebSolutions Inc</a>	Full Dropbox	×
 Dropbox Windows 8	Dropbox Windows 8	Official app	×



# JSON Web Token (JWT)

## Header

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}
```

## Claims

```
{  
  "iss": "http://myIssuer",  
  "exp": "1340819380",  
  "aud": "http://myResource",  
  "sub": "alice",  
  
  "client_id": "xyz",  
  "scope": ["read", "search"]  
}
```

eyJhbGciOiJIub251In0.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMD.4MTkzODAsDQogImh0dHA6Ly9leGFt

**Header**

**Claims**

**Signature**