



Using ArcGIS with OAuth 2.0

Aaron Parecki @aaronpk
CTO, Esri R&D Center Portland

**Esri Developer Summit
Middle East & Africa**

19-21 November 2013
Park Hyatt Dubai

Before OAuth

- Apps stored the user's password
- Apps got complete access to a user's account
- Users couldn't revoke access to an app except by changing their password
- Compromised apps exposed the user's password

Before OAuth

- Services recognized the problems with password authentication
- Many services implemented things similar to OAuth 1.0
 - Flickr: “FlickrAuth” frobs and tokens
 - Google: “AuthSub”
 - Facebook: requests signed with MD5 hashes
 - Yahoo: BBAuth (“Browser-Based Auth”)

The OAuth 2.0 Spec

<http://oauth.net/2/>

Definitions

- **Resource Owner:** The User
- **Resource Server:** The API
- **Authorization Server:** Often the same as the API server
- **Client:** The Third-Party Application

Use Cases

- Web-server apps
- Browser-based apps
- Username/password access
- Application access
- Mobile apps

Use Cases – Grant Types

- Web-server apps – `authorization_code`
- Browser-based apps – `implicit`
- Application access – `client_credentials`
- Mobile apps – `implicit`

Creating an App

developers.arcgis.com

ArcGIS for Developers

PLATFORM

FEATURES


PLANS


DOCUMENTATION

SUPPORT

 aaronpk




 Applications

 Hosted Data

 GIS Tools

 Usage Summary

 Licensing

 Downloads

< Register New Application

Name

My App

Required

Tags

mapping, iphone, android

Comma separated, e.g. "mapping, iphone, android"

Redirect URI

Optional. Adding redirect URIs to your application will allow users with ArcGIS online subscriptions to login to your application via OAuth 2. Otherwise leave this blank.

Description

Tell us about your application. 3000 is allowed.

Create an Application

ArcGIS for Developers

PLATFORM

FEATURES

PLANS

DOCUMENTATION

SUPPORT

aaronpk



Applications

Hosted Data

GIS Tools

Usage Summary

Licensing

Downloads

< OAuth Demo

[VIEW IN ARCGIS ONLINE](#)

Application Details

API Access

Redirect URIs

Usage Summary

DELETE APP

Edit Application Details

Name

OAuth Demo

Tags

test, aaronpk, oauth

Comma separated, e.g. "mapping, iphone, android"

Description

UPDATE APPLICATION

Get your app's client_id

ArcGIS for Developers

PLATFORM

FEATURES

PLANS

DOCUMENTATION

SUPPORT

Applications

Hosted Data

GIS Tools

Usage Summary

Licensing

Down



OAuth Demo

Application Details

API Access

Redirect URIs

Usage Summary

OAuth Credentials

Client ID

eKNjzFFjH9A1ysYd

Client Secret

DELETE APP

Set the redirect_uri

ArcGIS for Developers

PLATFORM

FEATURES

PLANS

DOCUMENTATION

SUPPORT

Applications

Hosted Data

GIS Tools

Usage Summary

Licensing

< OAuth Demo

Application Details

API Access

Redirect URIs

Usage Summary

oauthdemo://auth

http://your-site.com/redirect-url

Adding redirect URIs to your application will allow users with ArcGIS online subscriptions to application for users outside of ArcGIS online you dont need redirect urls.

DELETE APP

UPDATE REDIRECT URIS

Mobile Apps

Implicit Grant

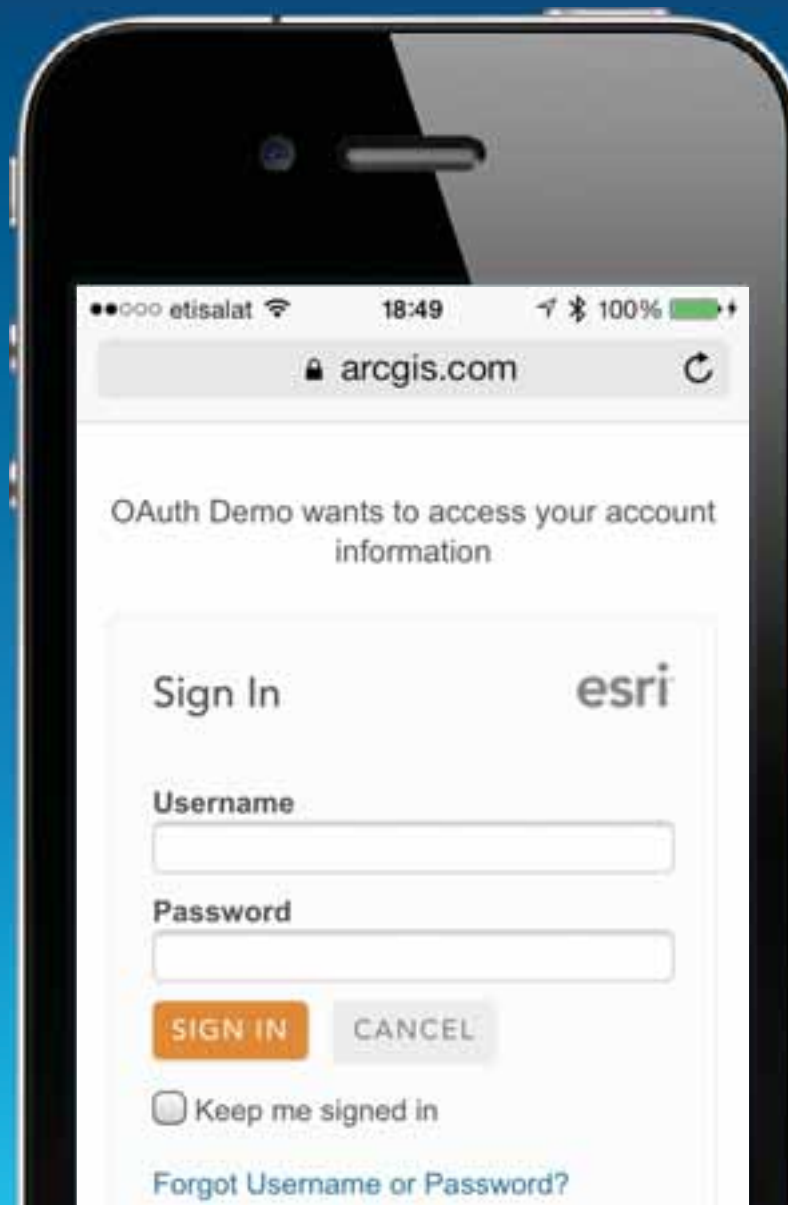
Create a Sign-In Button



Launch Safari to the ArcGIS Online Authorization Endpoint

```
- (IBAction)signInWasTapped:(id)sender
{
    NSURL *url = [NSURL URLWithString:@"https://www.arcgis.com/sharing/oauth2/authorize?"
                                     "response_type=token&"
                                     "client_id=eKNjzFFjH9A1ysYd&"
                                     "redirect_uri=oauthdemo://auth"];
    [[UIApplication sharedApplication] openURL:url];
}
```

The User Signs In



Redirect back to your app

ArcGIS Online redirects back to your app using a custom URI scheme.

Access token is included in the redirect, just like browser-based apps.

```
oauthdemo://auth  
#access_token=BAAEEemo2nocQBAFF0eRTd...
```

Parse the token from the URL

```
// App launched from a URL
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication options:(NSDictionary *)options {
    if([[url host] isEqualToString:@"auth"]) {
        NSDictionary *params = [self parseQueryString:[url fragment] stringByReplacingPercentEscapesUsingEncoding:NSUTF8StringEncoding];

        // Store the access token
        NSLog(@"Saving new token: %@", params);
        [[NSUserDefaults standardUserDefaults] setObject:[params objectForKey:@"access_token"] forKey:@"access_token"];

        // Calculate the expiration date so we know if the token is invalid
        NSDate *expDate = [NSDate dateWithTimeIntervalSinceNow:[params objectForKey:@"expires_in"]];
        NSLog(@"Expires at: %@", expDate);
        [[NSUserDefaults standardUserDefaults] setObject:expDate forKey:@"expirationDefaultsName"];

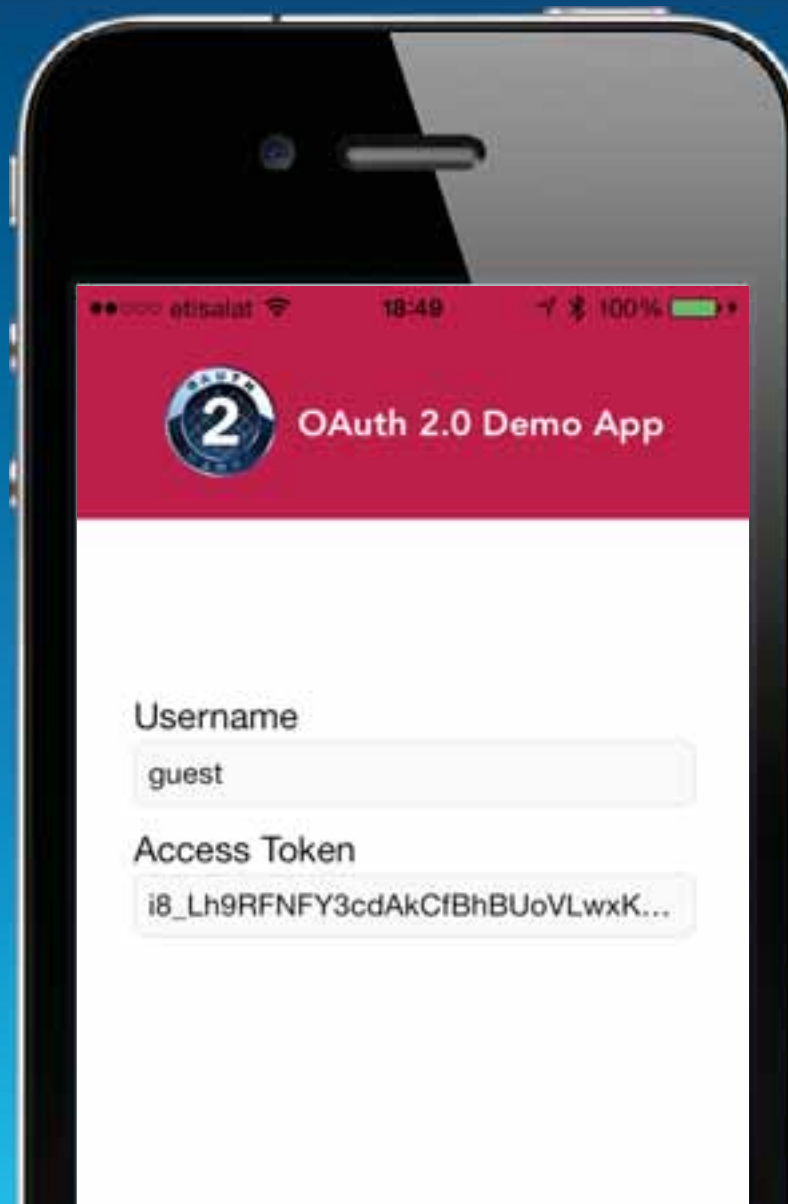
        // Store the username
        [[NSUserDefaults standardUserDefaults] setObject:[params objectForKey:@"username"] forKey:@"username"];

        // Save
        [[NSUserDefaults standardUserDefaults] synchronize];

        // Notify the view that a new token is available
        [[NSNotificationCenter defaultCenter] postNotificationName:OADNewTokenAvailable object:self];
    }

    return YES;
}
```

The User is Signed In!





Mobile Apps

- Use the “Implicit” grant type
- No server-side code needed
- Client secret not used
- Mobile app makes API requests directly

Web Server Apps

Authorization Code Grant



Create a “Log In” link

Link to:

```
https://www.arcgis.com/sharing/  
oauth2/authorize?  
response_type=code&client_id=YOUR_CLI  
ENT_ID&redirect_uri=REDIRECT_URI
```



Create a “Log In” link

Link to:

```
https://www.arcgis.com/sharing/  
oauth2/authorize?  
response_type=code&client_id=YOUR_CLI  
ENT_ID&redirect_uri=REDIRECT_URI
```



Create a “Log In” link

Link to:

```
https://www.arcgis.com/sharing/  
oauth2/authorize?  
response_type=code&client_id=YOUR_CLI  
ENT_ID&redirect_uri=REDIRECT_URI
```




Create a “Log In” link

Link to:

```
https://www.arcgis.com/sharing/  
oauth2/authorize?  
response_type=code&client_id=YOUR_CLI  
ENT_ID&redirect_uri=REDIRECT_URI
```

User visits the authorization page

`https://www.arcgis.com/sharing/oauth2/authorize?`

`response_type=code&client_id=YOUR_CLIENT_ID
&redirect_uri=REDIRECT_URI`

OAuth Test wants to access your account information

Sign In

esri

Username

Password

SIGN IN

CANCEL

[Forgot Username or Password?](#)



**On success, user is redirected
back to your site with auth code**

`https://example.com/auth?code=AUTH_CODE_HERE`

**On error, user is redirected back to
your site with error code**

`https://example.com/auth?error=access_denied`



Server exchanges auth code for an access token

Your server makes the following request

```
POST https://www.arcgis.com/  
sharing/oauth2/token
```

Post Body:

```
grant_type=authorization_code  
&code=CODE_FROM_QUERY_STRING  
&redirect_uri=REDIRECT_URI  
&client_id=YOUR_CLIENT_ID  
&client_secret=YOUR_CLIENT_SECRET
```



Server exchanges auth code for an access token

Your server gets a response like the following

```
{  "access_token": "RsT5030zqMLgV3Ia",  
    "expires_in": 3600,  
    "refresh_token": "e1qok2RRua48lXI",  
    "username": "aaronpk"  
}
```

or if there was an error

```
{  
    "error": "invalid_request"  
}
```

Browser-Based Apps

Implicit Grant



Create a “Log In” link

Link to:

```
https://www.arcgis.com/sharing/  
                                oauth2/authorize?  
response_type=token&client_id=YOUR_CLIENT_ID&redirect_uri=REDIRECT_URI
```

User visits the authorization page

`https://www.arcgis.com/sharing/oauth2/authorize?`

`response_type=token&client_id=YOUR_CLIENT_ID
&redirect_uri=REDIRECT_URI`

OAuth Test wants to access your account information

Sign In

esri

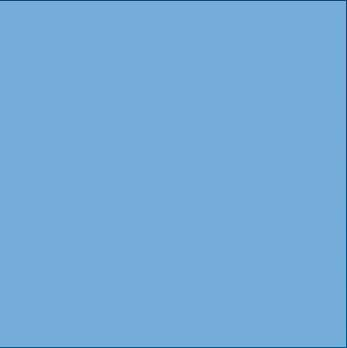
Username

Password

SIGN IN

CANCEL

Forgot Username or Password?



**On success, user is redirected
back to your site with the access
token in the fragment**

`https://example.com/auth#token=ACCESS_TOKEN`

**On error, user is redirected back to
your site with error code**

`https://example.com/auth#error=access_denied`



Browser-Based Apps

- Use the “Implicit” grant type
- No server-side code needed
- Client secret not used
- Browser makes API requests directly

Grant Type Summary

- `authorization_code`:
Web-server apps
- `implicit`:
Mobile and browser-based apps
- `password`:
Username/password access
- `client_credentials`:
Application access

Authorization Code

- User visits auth page
`response_type=code`
- User is redirected to your site with auth code
`http://example.com/?code=xxxxxxx`
- Your server exchanges auth code for access token
`POST /token`

`code=xxxxxxx&grant_type=authorization_code`

Implicit

- User visits auth page
`response_type=token`
- User is redirected to your site with access token
`http://example.com/#token=xxxxxxx`
- Token is only available to the browser since it's in the fragment

Client Credentials

- Your server exchanges client ID/secret for access token
POST /token

```
client_id=xxxxxxx&client_secret=yyyyyyy&  
grant_type=client_credentials
```

Application Access

Client Credentials Grant



Client Credentials Grant

POST `https://www.arcgis.com/sharing/oauth2/token`

Post Body:

```
grant_type=client_credentials  
&client_id=YOUR_CLIENT_ID  
&client_secret=YOUR_CLIENT_SECRET
```

Response:

```
{  
  "access_token": "RsT5OjbzRn430zqMLgV3Ia",  
  "expires_in": 3600  
}
```


Accessing Resources

So you have an access token. Now what?



Use the access token to make requests

Now you can make requests using the access token.

```
GET http://www.arcgis.com/sharing/rest/portals/self  
?token=RsT5OjbzRn430zqMLgV3Ia
```



Eventually the access token may expire

When you make a request with an expired token, you will get this response

```
{  
  "error": "expired_token"  
}
```

Now you need to get a new access token!



Get a new access token using a refresh token

Your server makes the following request

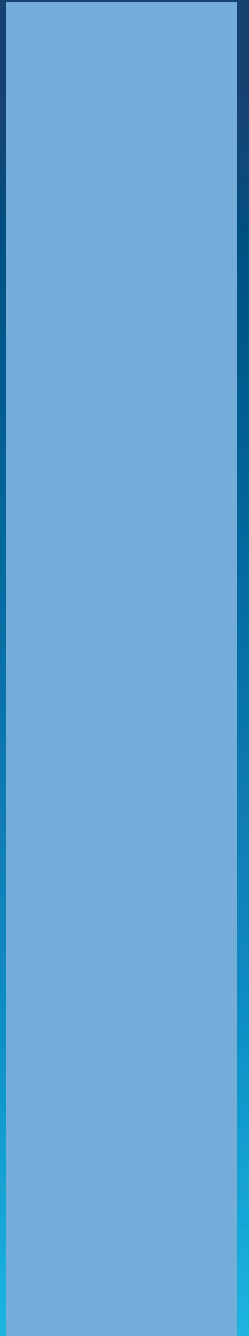
```
POST https://www.arcgis.com/sharing/
oauth2/token
```

```
grant_type=refresh_token
&refresh_token=e1qoXg7Ik2RRua48lXIV
&client_id=YOUR_CLIENT_ID
&client_secret=YOUR_CLIENT_SECRET
```

Your server gets a similar response as the original call to oauth/token with new tokens.

```
{
  "access_token": "RsT5OjbzRn430zqMLgV3Ia",
  "expires_in": 3600,
  "username": "aaronpk"
}
```

Other



Authentication

OAuth 2.0-based apps

Getting Started

User logins

User logins and OAuth 2.0

JavaScript, Flex, and other
browser-based apps

iOS, Android, and WPF apps

PHP, JSP, ASP.NET, and other
server-based web apps

App logins

App logins and OAuth 2.0

Non OAuth 2.0-based
apps

Authentication, OAuth 2.0, and ArcGIS

As a developer using the ArcGIS platform, you can build the following two types of applications (apps):

- Applications that target end users of the ArcGIS platform. These applications need to allow users to login to the platform via the application. These types of logins are known as **user logins**.
- Applications that target end users who are unknown to the ArcGIS platform. These applications need to login to the platform on behalf of the application. These kinds of logins are known as **app logins**.

You can build these apps using JavaScript, iOS, and Android, as well as Flex and Silverlight. The platform in this context means ArcGIS Online, which is available at arcgis.com, or an ArcGIS Portal available at a portal-specific URL along with all associated services.

This guide will teach you how to manage both user and app logins using the new OAuth 2.0-based ArcGIS APIs that are being introduced in ArcGIS.com in the March 2013 release. All new applications against ArcGIS Online should be developed using these OAuth 2.0-based APIs. Existing applications that implement these concepts using the existing APIs will continue to work and be supported but the identity of the application making the requests will remain unknown to the platform limiting what the application can do and participate in.

All apps that use OAuth 2.0 must be registered with the platform and have a platform-assigned AppID. You can register your applications by



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 enables a third-party application to obtain limited access to an HTTP service.

For Consumer developers...

If you're building...

- web applications
- desktop applications
- mobile applications
- Javascript or browser-based apps
- webpage widgets

OAuth is a simple way to publish and interact with protected data. It's also a safer and more secure way for people to give you access. We've kept it simple to save you time.

For Service Provider developers...

If you're supporting...

- web applications
- mobile applications
- server-side APIs
- mashups

If you're storing protected data on your users' behalf, they shouldn't be spreading their passwords around the web to get access to it. Use OAuth to give your users access to their data while protecting their account credentials.

[Get started...](#)



Links

github.com/Esri/OAuth2-Demo-iOS

developers.arcgis.com

Thanks.



@aaronpk

aparecki@esri.com

github.com/aaronpk