ENT171

Introduction to OAuth 2.0

Download class materials from
university.xamarin.com

Microsoft

Xamarin University

Objectives

1. Describe the benefits of OAuth 2.0
2. Register your app with an OAuth 2.0 server
3. Choose an OAuth 2.0 flow for your mobile app

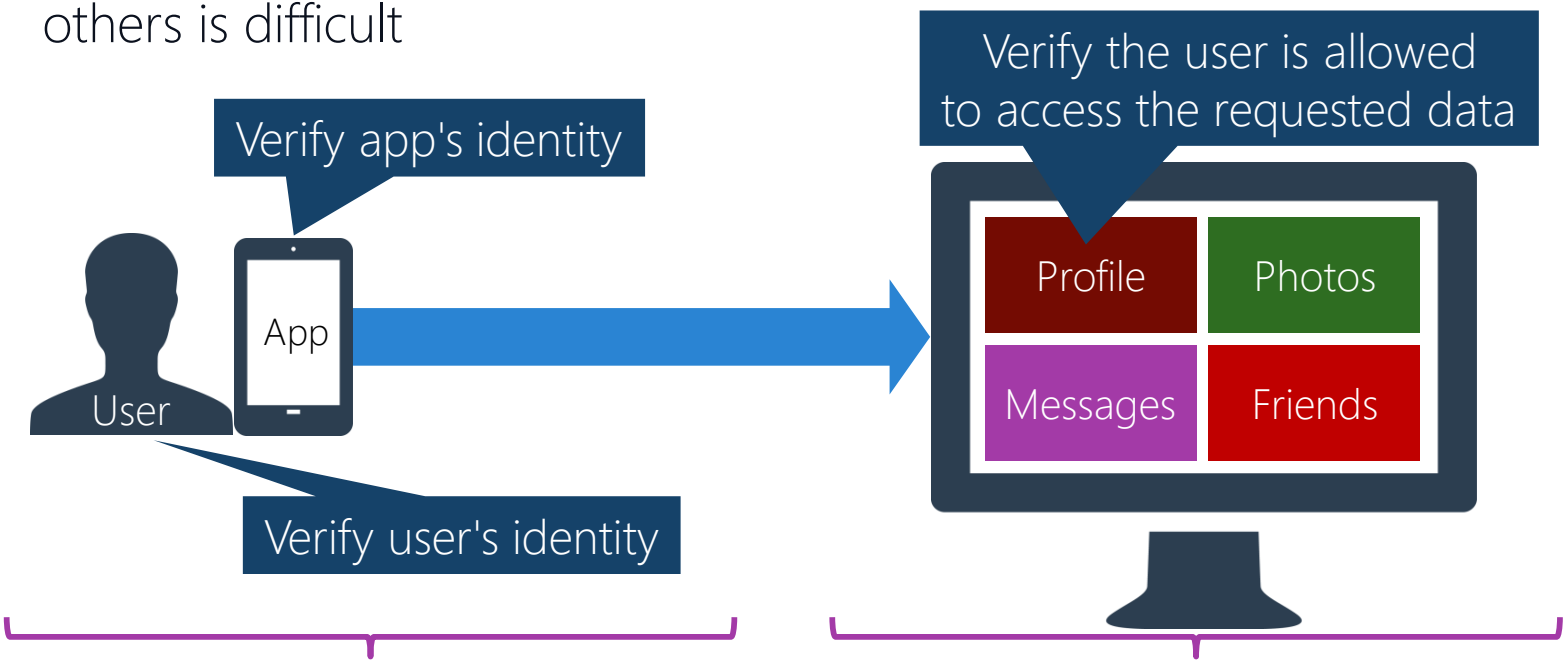
Describe the benefits of OAuth 2.0

Tasks

1. Discuss OAuth 2.0's role in industry
2. List some benefits of OAuth 2.0

Motivation

- Allowing a user to access their data on a server while securing it from others is difficult

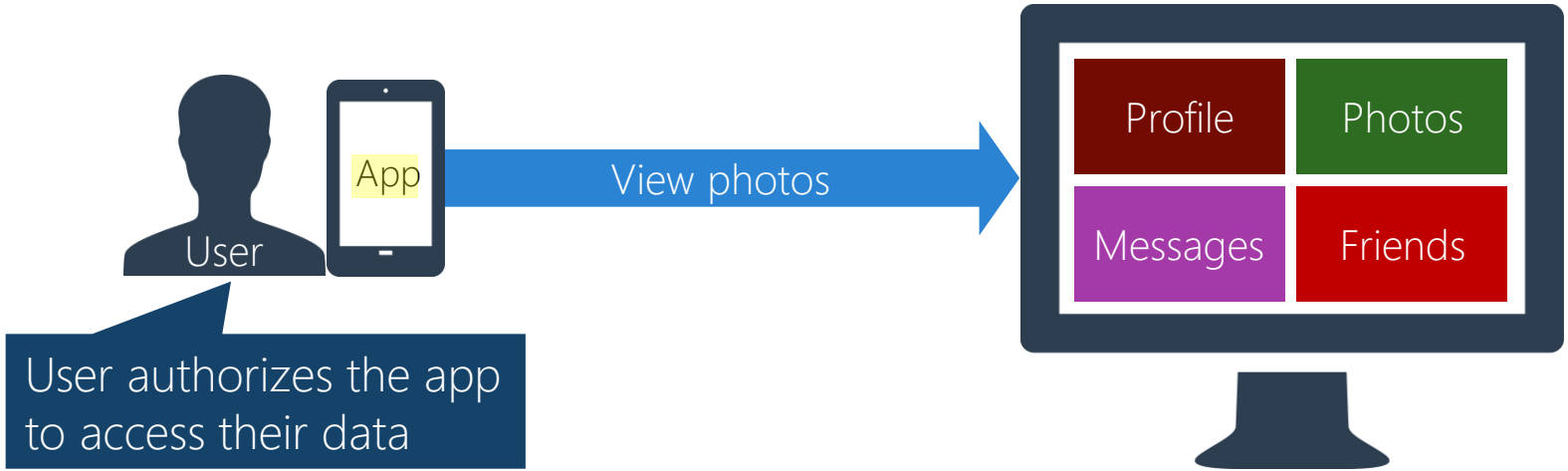


Verifying identity is called *Authentication*

Verifying access is called *Authorization*

What is OAuth?

- *Open Authorization (OAuth) 2.0* is an Authorization protocol that lets users give **another party**, like an app, permission to access their secure data



Who created OAuth?

- OAuth 2.0 was developed through a committee within the *Internet Engineering Task Force* (IETF), members were mostly large tech companies

Who uses OAuth?

- OAuth 2.0 has become the standard client-authorization mechanism for many popular web services

Why use OAuth?

- OAuth helps protect user credentials and data

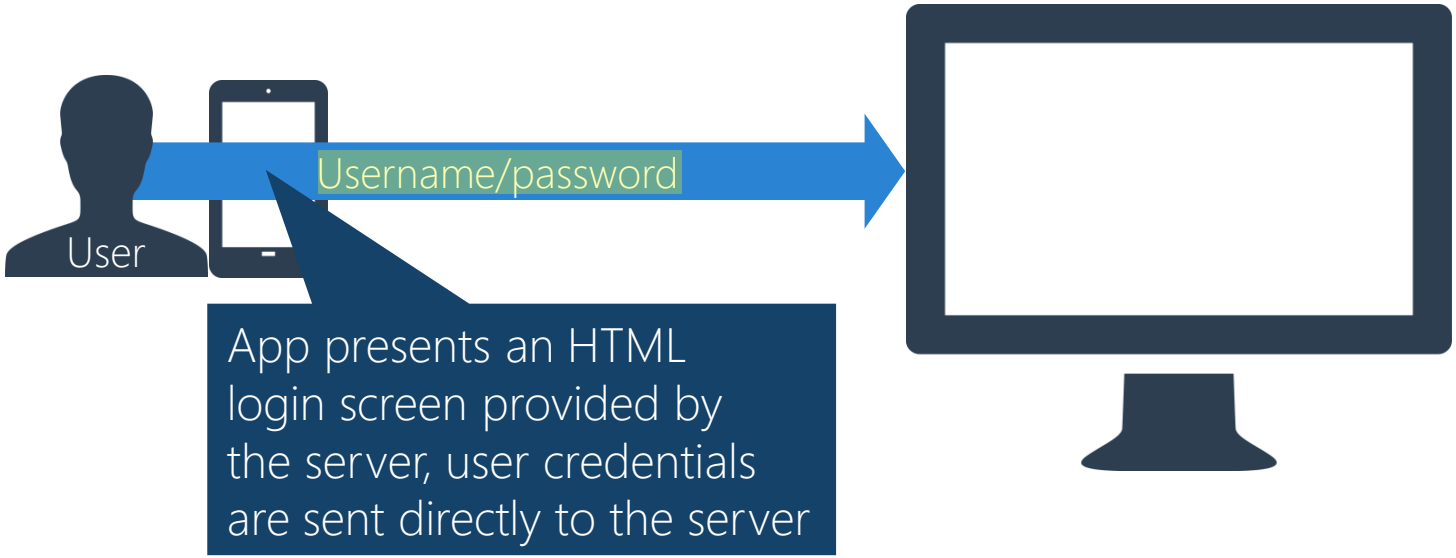
Credential Confidentiality

Access Granularity

Access Duration

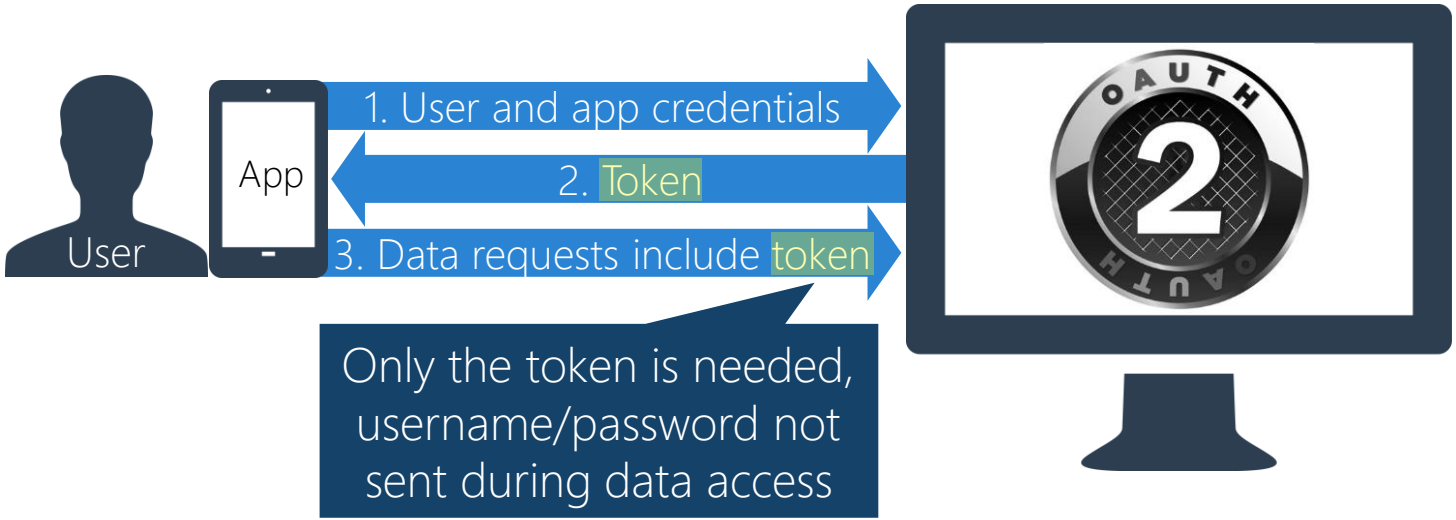
Credential confidentiality [login]

- The app does not see the user's **credentials** when the user authenticates



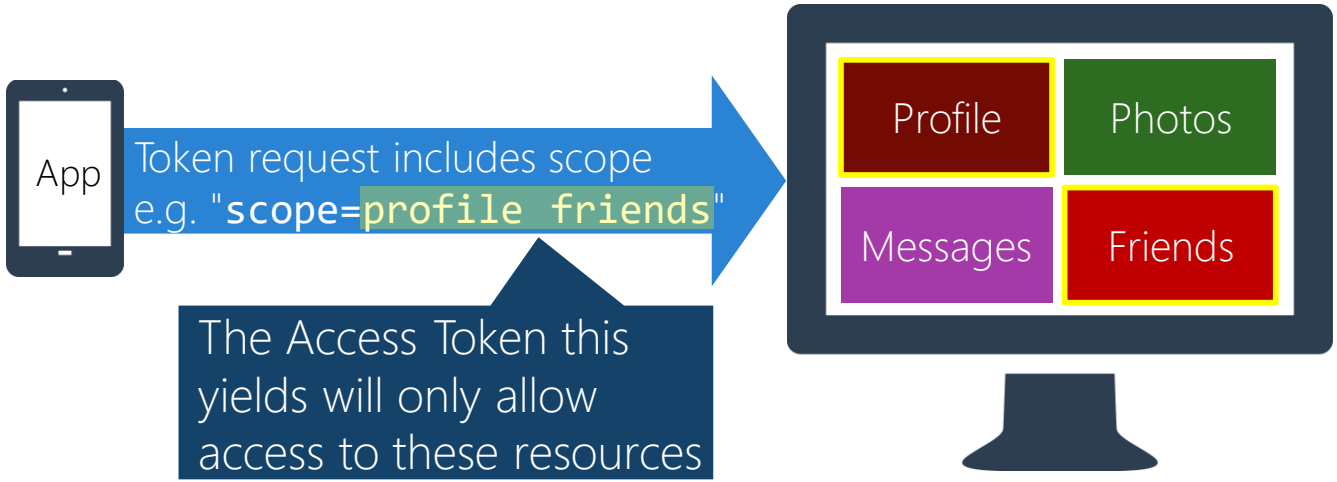
Credential confidentiality [access]

- The user and app send their credentials in order to get an **Access Token** which the app then uses to access the user's data



Access granularity

- OAuth client apps specify **the data they need** when they request an Access Token and their access is limited to just that data

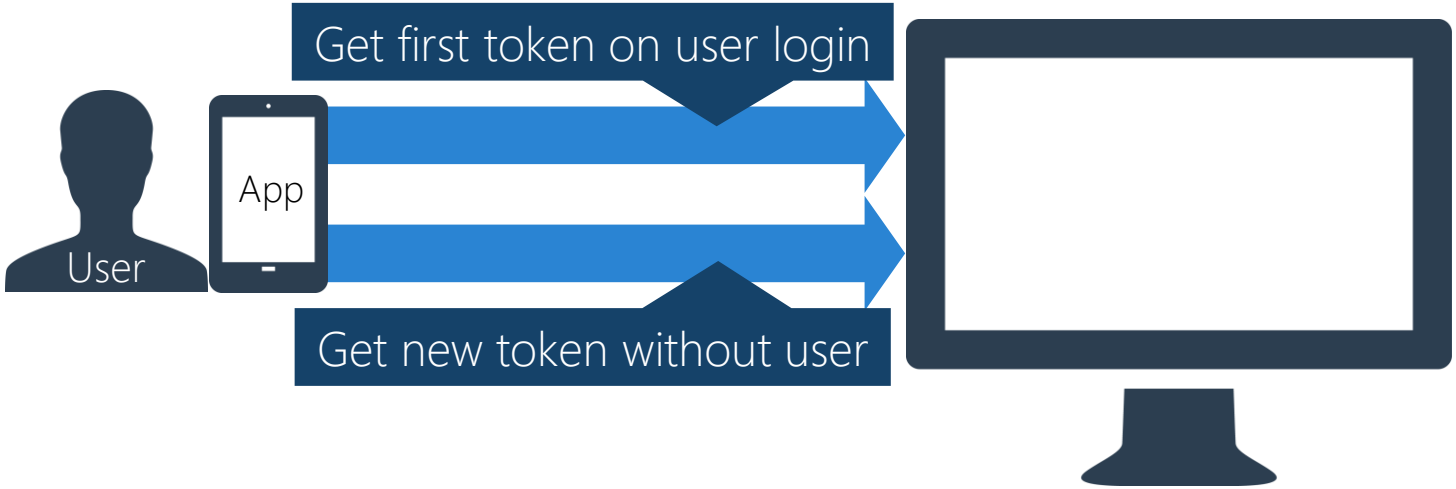


Access duration [expiration]

- OAuth lets the server limit the lifetime of an Access Token

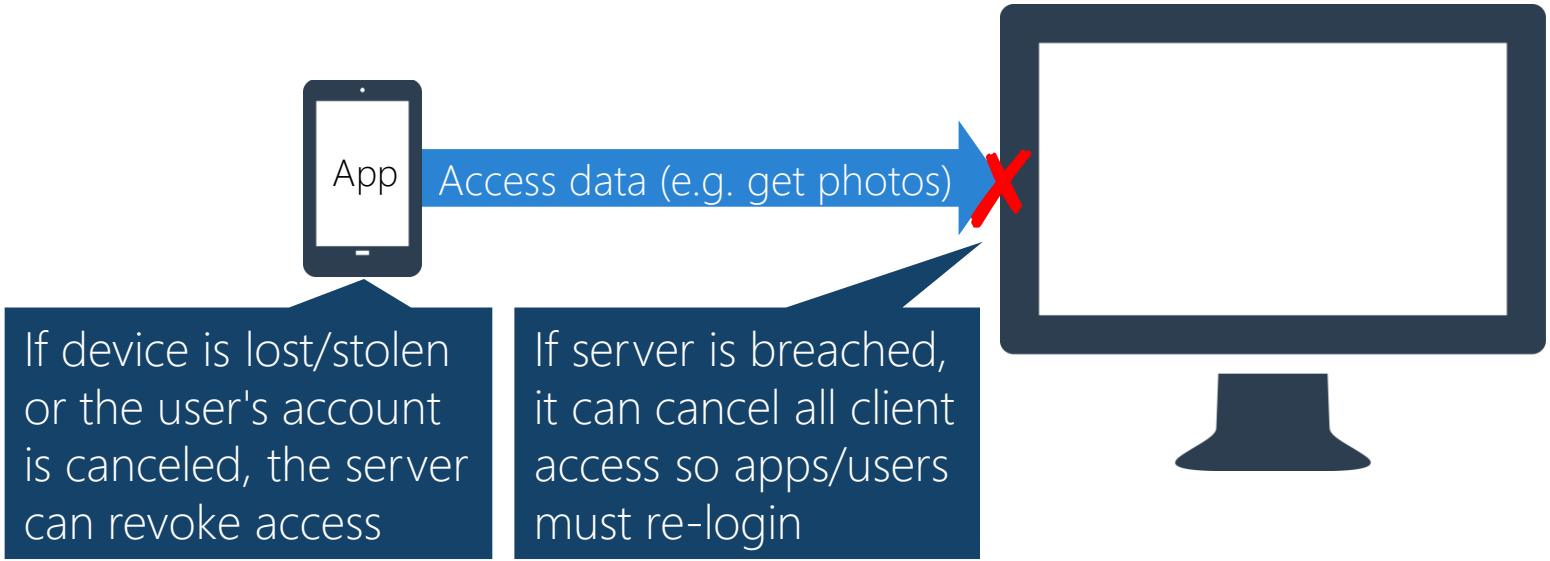
Access duration [refresh]

- A server can support token *refresh* so the app can get a new Access Token when theirs expires without requiring the user to login again



Access duration [revocation]

- OAuth lets the server revoke a client's access (some servers cancel outstanding Access Tokens and others just prevent refresh)



Register your app with an OAuth 2.0 server

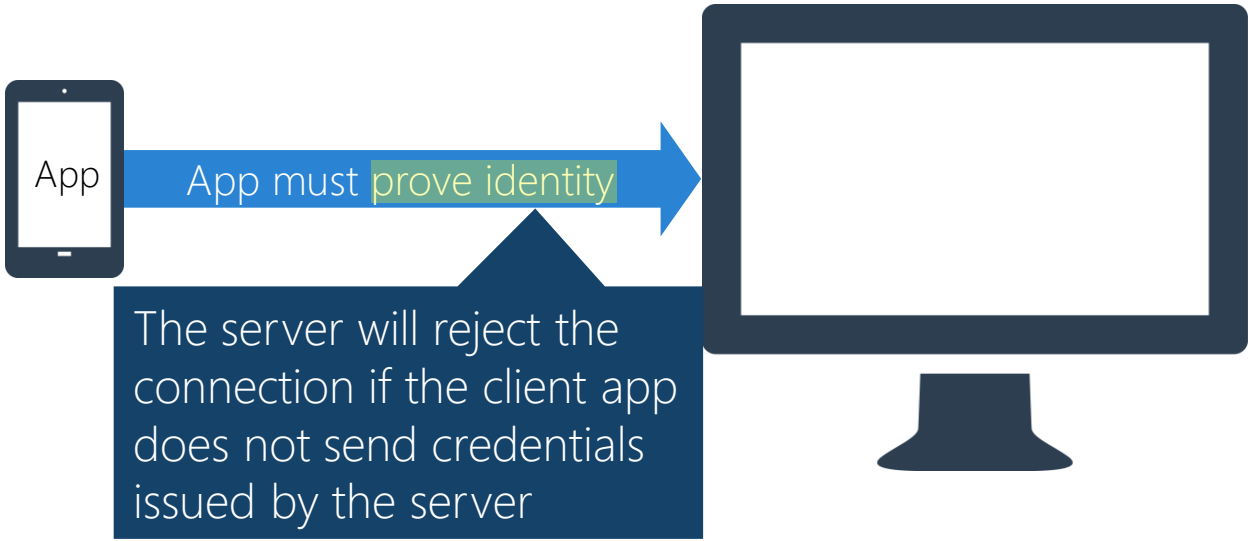
Xamarin University

Tasks

1. Define Client Identifier and Client Secret
2. Choose a Redirection Endpoint
3. Register your app with a server

Motivation

- Servers typically allow only **registered** apps to connect



Registration purpose

- Registration allows the client and server to agree on the **information** needed for the client to establish a connection

Client app author chooses a Redirection Endpoint and gives it to the server

Server generates Client Credentials (Client Identifier and Client Secret) and gives them to the app author

What is a Client Identifier?

- The *Client Identifier* is a string issued by the server that uniquely identifies the client app (roughly analogous to a username)

```
class MyApp
{
  string ClientIdentifier = "xxxxxxxxxxxxxxxxxxx";
  ...
}
```

Not considered a secret, can be embedded in your mobile app's source code

The OAuth 2.0 spec uses the term "Client Identifier". Most implementations follow this; however, some use variations like "App Key", "Consumer Key", etc.

What is a Client Secret?

- The *Client Secret* is a string issued by the server that proves the identity of the client app (roughly analogous to a password)

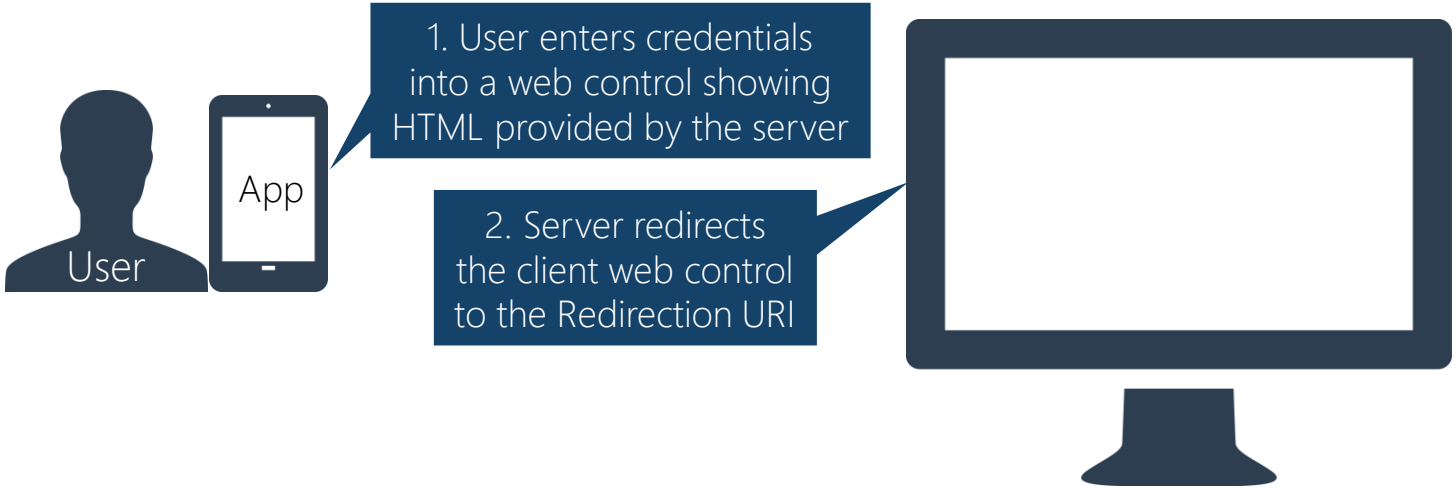
```
class MyApp
{
  string ClientSecret = "xxxxxxxxxxxxxxxxxxx";
  ...
}
```

Some OAuth 2.0 authentication procedures require that you send the Client Secret. If you embed it in your app's source code, it is no longer considered a secret.

The OAuth 2.0 spec uses the term "Client Secret". Most implementations follow this; however, some use variations like "App Secret", "Consumer Secret", etc.

What is Redirection Endpoint?

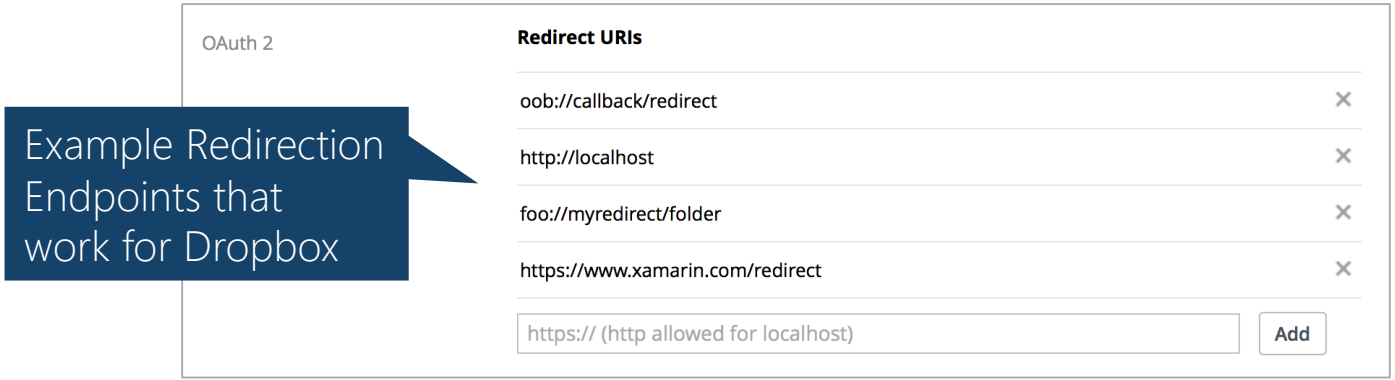
- The *Redirection Endpoint* is a URI chosen by the app author and used by the server to signal that its interaction with the user is complete



The OAuth 2.0 spec uses the term "Redirection Endpoint". Many implementations use "Callback URI" or "Redirect URI" instead.

Redirection Endpoint format

- OAuth 2.0 says the Redirection Endpoint must be an absolute URI without a fragment



Each server can have additional requirements; read the server documentation for details.

Redirection Endpoint guidance

- Choose a Redirection Endpoint that maps to a server you own or does not map to a real page at all – this avoids having the Code/Token added to the server's log

Some servers allow localhost which is a good choice when available

```
string RedirectUri = "http://localhost";
```

Some servers let you use a fictitious URI which is also a reasonable choice

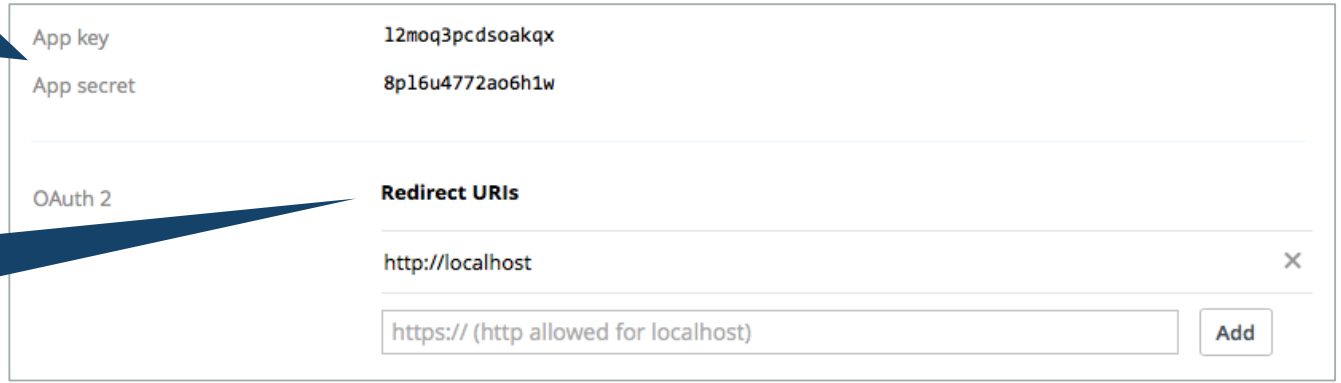
```
string RedirectUri = "foo://myredirect/folder";
```

How to register

- The registration process differs for every server; however, it typically involves creating an account and entering app info into a web portal

Server gives you Client Id/Secret

You give the server your Redirect URI(s)



Choose an OAuth 2.0 flow for your mobile app

Xamarin University

Tasks

1. Survey the OAuth 2.0 flow types
2. Choose a flow for your mobile app

Motivation [overview]

- OAuth 2.0 servers have multiple ways for clients to authorize because apps vary in where they run and what type(s) of access they need

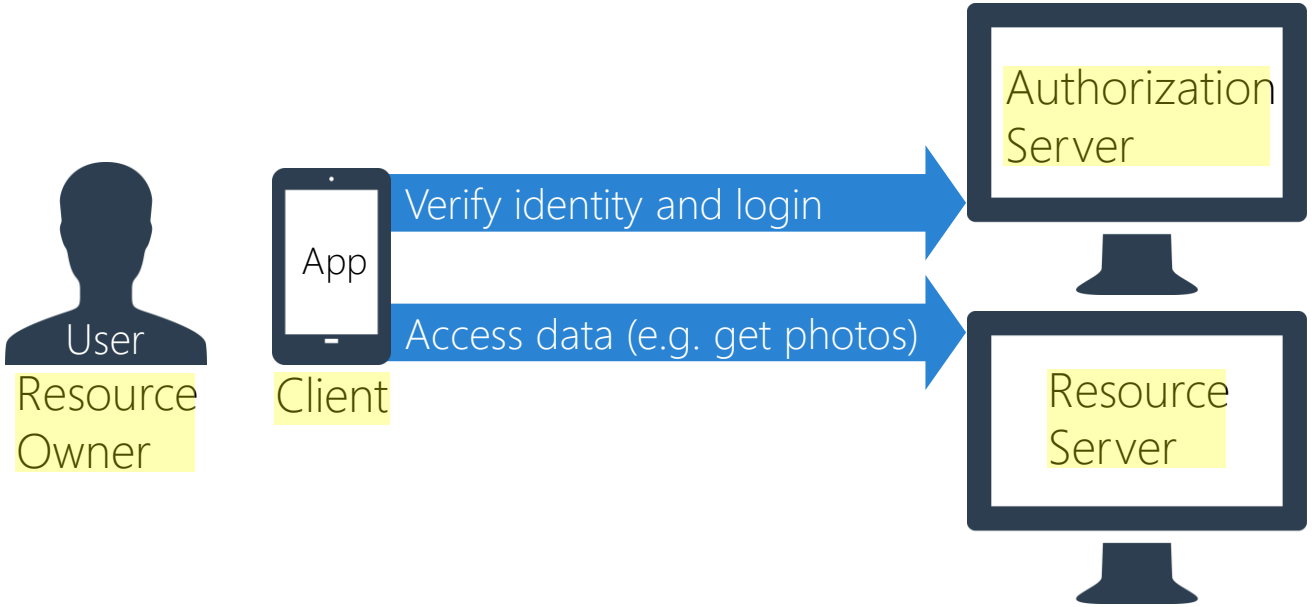
Client type

User trust

Refresh needed?

OAuth roles

- OAuth 2.0 defines 4 **roles** involved in granting access to secured resources



Motivation [client types]

- OAuth 2.0 designates clients as *public* or *confidential* based on their ability to maintain confidentiality of their client credentials

Installed app
mobile, laptop, or desktop

Web app
client-side JavaScript

Website
code on a server

Attacker could extract client credentials from app's binary

Public

Confidential

Motivation [user trust]

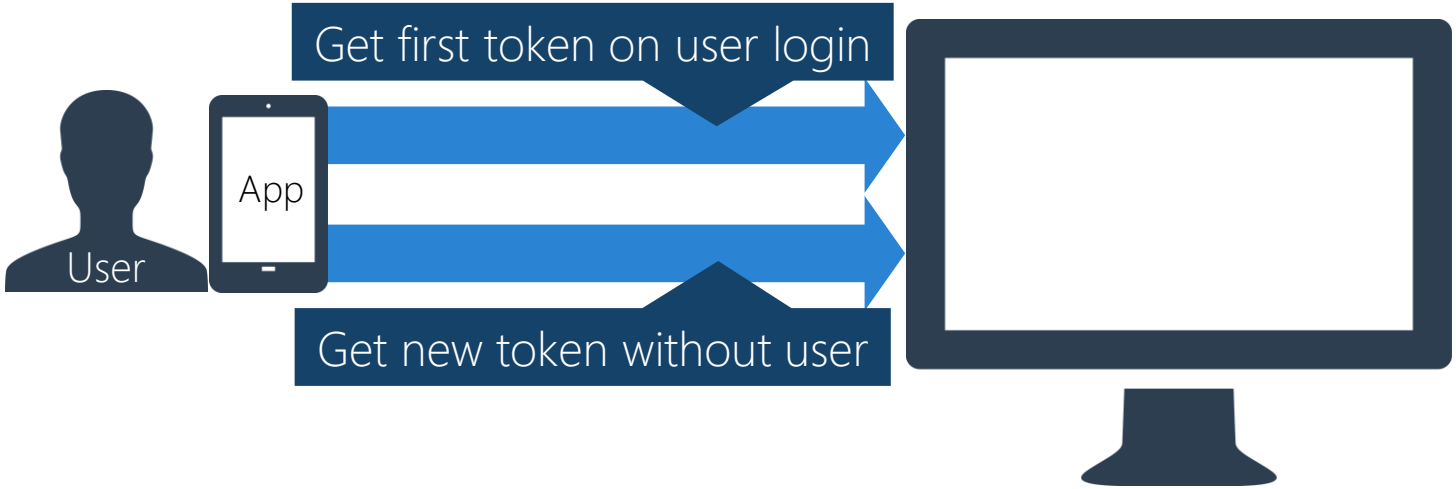
- A user may trust an app enough to provide their username/password, this is rare, but may be reasonable for "1st-party" apps

E.g. an app made by Xamarin...

User may feel comfortable providing their credentials to this app

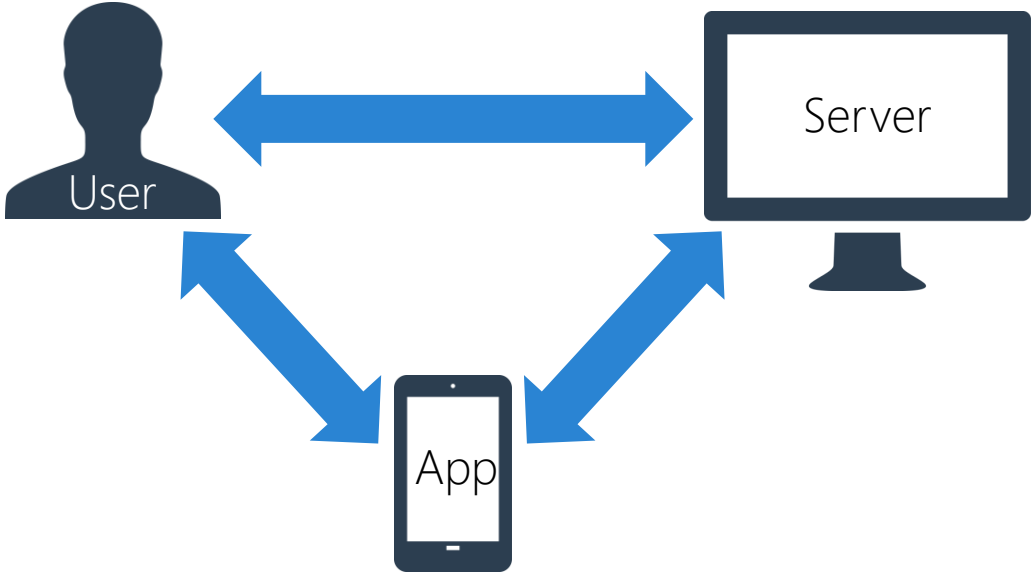
Motivation [refresh]

- A server can support token *refresh* so the app can get a new Access Token when theirs expires without requiring the user to login again



What is a flow?

- An OAuth *flow* defines the sequence of steps between the User, the App, and the Server needed to authorize a user



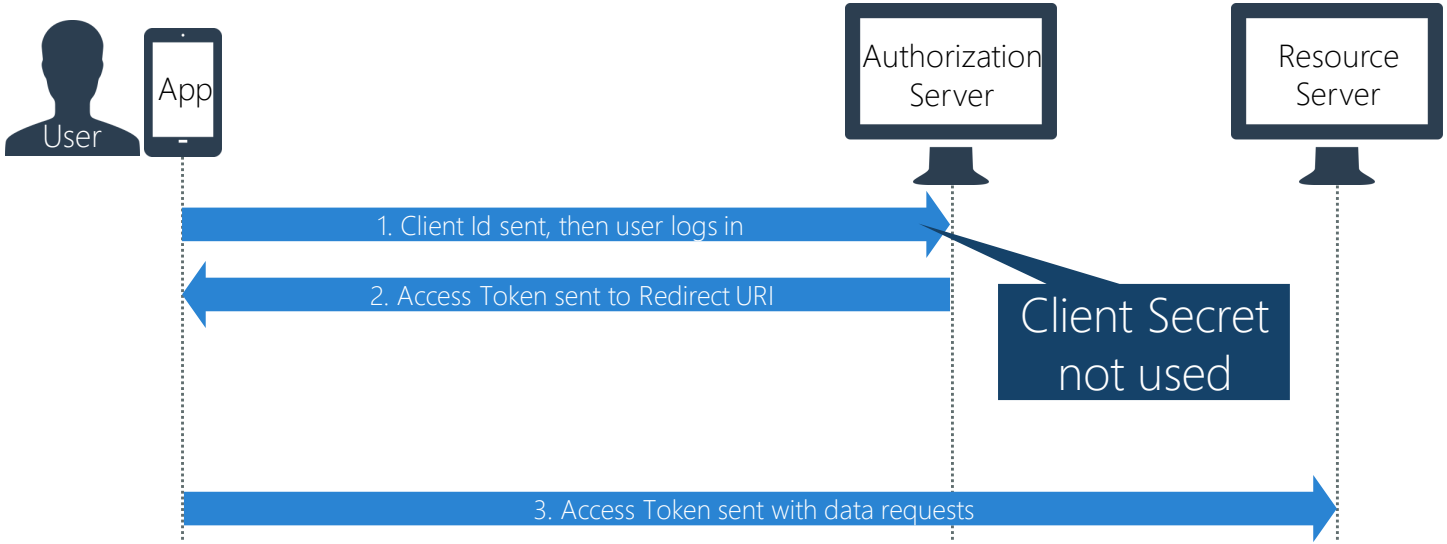
Flows

- OAuth 2.0 defines four standard **flows**

The OAuth 2.0 spec uses the terms shown here; however many servers use different names for the flows.

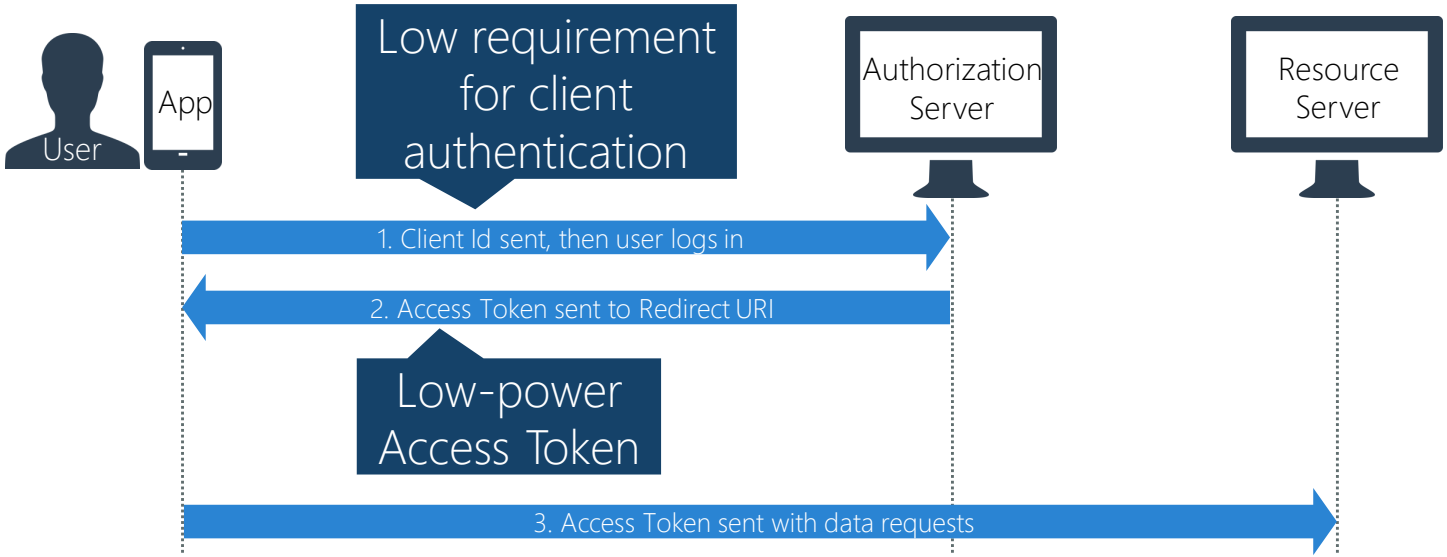
Implicit flow [mechanics]

- The *Implicit* flow is an authorization process with low requirements for app authentication



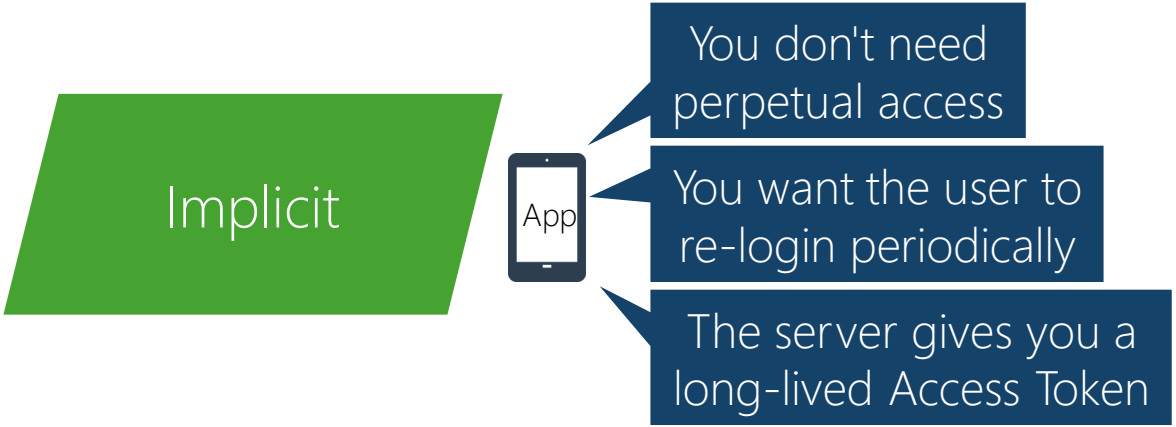
Implicit flow [power]

- The Access Tokens provided through the Implicit flow are usually short-lived and don't support refresh (these details vary by server)



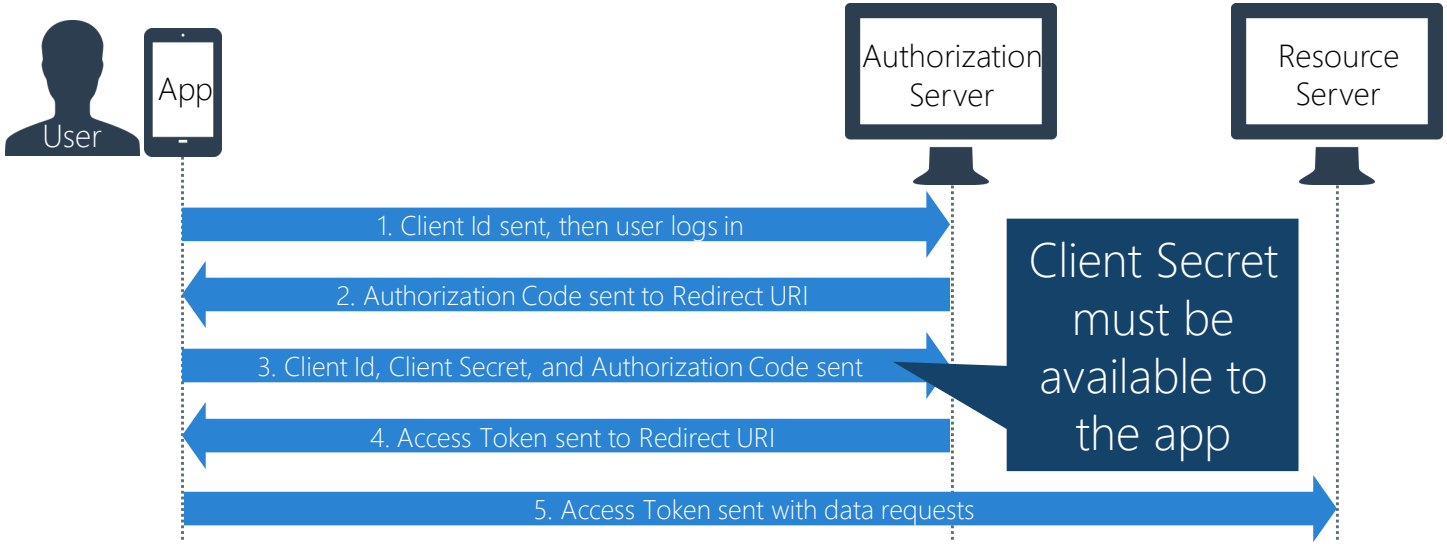
Implicit flow [guidance]

- The Implicit flow is the recommended flow for public clients like mobile apps because it requires only the Client Id (not the Client Secret)



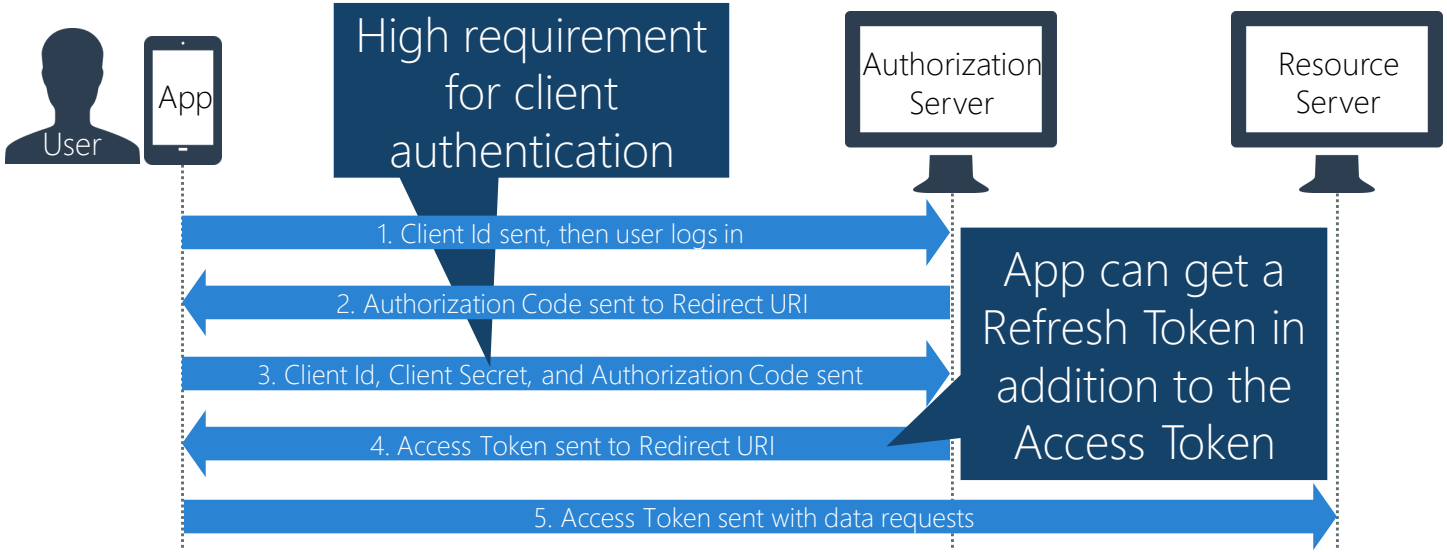
Authorization Code flow [mechanics]

- The *Authorization Code* flow is an authorization process with high requirements for client authentication



Authorization Code flow [power]

- The Access Tokens provided through the Authorization Code flow are usually short-lived; however, most servers will let the app refresh to get a new Access Token when the current one expires

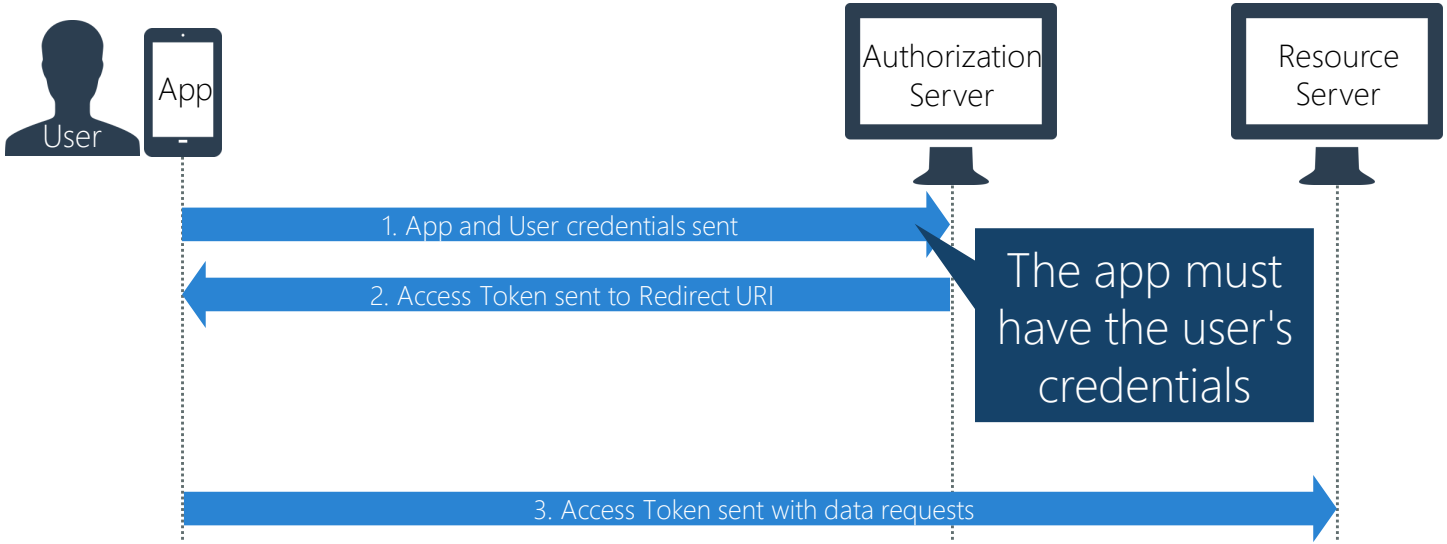


Authorization Code flow [guidance]

- Use the Authorization Code flow if you need the extra power it provides (e.g. refresh); however, note that your Client Secret must be available to the app which likely means it should no longer be considered a secret

Password Credentials flow [mechanics]

- The *Password Credentials* flow lets the app send the user's credentials directly instead of the user logging in via a web page

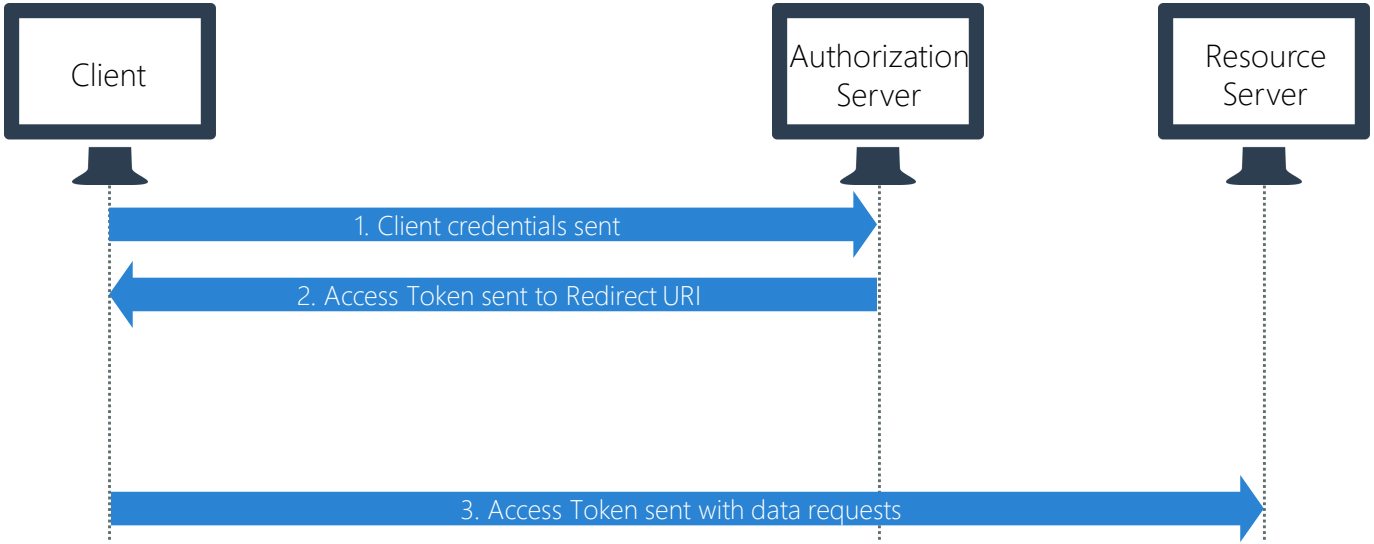


Password Credentials flow [guidance]

- The Password Credentials flow is rarely used by mobile apps since the user must trust the app enough to give it their credentials

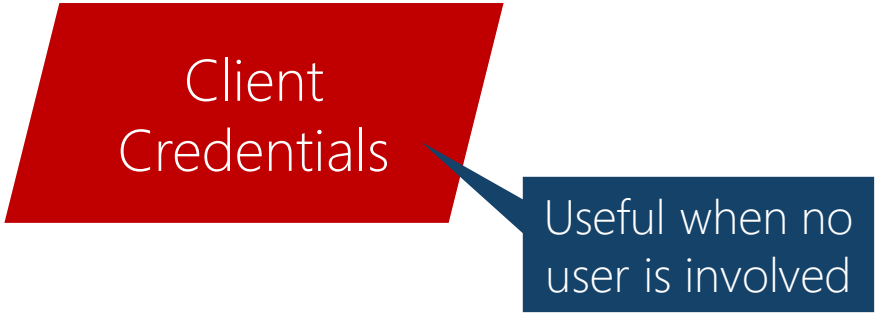
Client Credentials flow [mechanics]

- The *Client Credentials* flow lets the client send its own credentials in order to access data on a server (there is no user involved)



Client Credentials flow [guidance]

- The Client Credentials flow is not typically used for mobile apps; it is mainly used for server-to-server communication



Guidance for mobile apps

- Prefer **Implicit flow** when it meets your needs, use the Authorization Code flow if you need the extra power (e.g. you need to refresh the token)

Thank You!

Please complete the class survey in your profile:
university.xamarin.com/profile

Microsoft