Windows Unity Ad Network Plugin

A simple, flexible, easy to use plugin for Unity3D that allows Ads to be displayed in Windows Phone & Windows Store XAML C# Apps

Setting up the plugin:

Unity

Import the package into your Unity project by selecting *Assets->Import Package->Custom Package*.

Add either the **AdMobAd** or **pubCenterAd** prefab into your scene (Found in the *prefab* folder).

Once you have added them into your scene you must enter your correct credentials (AdUnitID, ApplicationID etc), the width/height of the ad and in the case of AdMob the format (Banner, or SmartBanner) See AdMob documentation

You can set the position of your ad with the horizontal and vertical alignment variables. If the horizontal alignment is center, the ad will be in the center of the screen, and if the vertical alignment is the top, the ad will be centered at the top of the screen.

Note: If the variables are not showing up in the Unity Inspector , make sure the project is set to the right platform. You can set this by selecting File->Build Settings (Ctrl-Shift-B)

Test Values

For pubCenter Ads you can leave the default values provided and you will get test ads. Currently using actual values will not work in the emulator, but they will work on a device. Windows Store test values, Windows Phone test values for PubCenter Ads.

For AdMob, you will need to change the values. If you check the "Test Ad Mob" check box it will do an *adRequest* (AdMob explanation). This is important to use during development to avoid generating false impressions.

Remember don't leave this selected when you submit the game to the store.

AdFiller

The AdFiller is for when no ad is available. An *adFiller* consists of an image and an url, allowing you to direct traffic to your site or another app/game. Use this responsibly, and comply with all EULAs

Once everything is set up, go head and hit build.

For debugging purposes errors will be printed out, you can turn this off by un-checking the printDebug.

Windows Phone Apps

First off, in Visual Studio, open your **WMAppManifest.xml** (which can be found in *Solution Explorer->Properties*) and add these capabilities

Required Capabilities (AdMob)

- ID_CAP_NETWORKING - Access to network services is required when requesting ads.
- ID_CAP_WEBBROWSERCOMPONENT - Required since the AdView is a web browser.
- ID_CAP_MEDIALIB_PLAYBACK - Provides access for currently playing media items.
- ID_CAP_MEDIALIB_AUDIO - Provides read access to audio items in media library.

Required Capabilites (pubCenter)

- ID_CAP_IDENTITY_USER
- ID_CAP_MEDIALIB_PHOTO
- ID_CAP_NETWORKING

- ID_CAP_PHONEDIALER
- ID_CAP_WEBBROWSERCOMPONENT

Next, go to your references, we will need to make some changes

Remove both of the **Microsoft.Advertising** and the **Windows_Ad_Plugin** dlls.

Next we will bring in the **Windows Phone Ad SDK**. To add the Microsoft Advertising SDK for Windows Phone 8(XAML) reference, in the Solution Explorer right click on *'References'* and then click *'Add Reference'*. In the window that appears select *Windows Phone - > Extensions - > Microsoft Advertising SDK for Windows Phone 8 (XAML)*.

Now we need to re-add the reference to the **Windows_Ad_Plugin.dll**. This is done the same way as the Ad SDK, except click the browse button on the bottom right hand corner of the window. Then navigate to *"Your Unity Project"/Assets/plugins/WP8* and select the Windows_Ad_Plugin.dll.

Now lets open the **MainPage.xaml.cs** and add some code to hook up the Dispatcher class to handle invokes on the App or UI thread

First off add these two functions, that tie into the Unity App thread and the UI thread of the solution

```
public void InvokeOnAppThread(Action callback)
{
    UnityApp.BeginInvoke(() => callback());
}

public void InvokeOnUIThread(Action callback)
{
    Dispatcher.BeginInvoke(() => callback());
}
```

Then add these lines of code, within the UnityLoaded function to wire it all up

```
Windows_Ad_Plugin.Dispatcher.InvokeOnAppThread = InvokeOnAppThread;
Windows_Ad_Plugin.Dispatcher.InvokeOnUIThread = InvokeOnUIThread;
```

Almost done, next we need to pass in a grid for the ad to be added to. For this we use the DrawingSurfaceBackground.

After the two previous lines add the following to the UnityLoaded function

```
Windows_Ad_Plugin.Helper.Instance.SetGrid(DrawingSurfaceBackground);
```

Now we can run the game and see the ads appear

Windows Store Apps

Note: only pubCenter ads are currently working in Windows Store games

In Visual Studio, open **Package.appmanifest** (found in *Solution Explorer*)and the following required capabilities

- Internet(client)

Now lets turn our attention to our references.

First delete the **Microsoft.Advertising.WinRT.UI** and **Microsoft.Advertising.WinRT** references. Then add our needed advertising sdk reference. To add the Microsoft Advertising SDK for Windows 8.1(XAML) reference, in the Solution Explorer right click on *'References'* and then click *'Add Reference'*. In the window that appears select *Windows - > Extensions - > Microsoft Advertising SDK for Windows 8.1 (XAML)*.

Next we will move on to the code. Open the **App.xaml.cs** file and paste the following line in the App constructor:

```
appCallbacks.Initialized += appCallbacks_Initialized;
```

Then right below the constructor add this function,

```
void appCallbacks_Initialized()
{
```

```
        Windows_Ad_Plugin.Dispatcher.InvokeOnAppThread = InvokeOnAppThread;
        Windows_Ad_Plugin.Dispatcher.InvokeOnUIThread = InvokeOnUIThread;
    }
```

Next we need to add the hookups so we can invoke on the UI and App thread. Add these two functions just below the *appCallbacks_Initialized* function

```
    public void InvokeOnAppThread(Action callback)
    {
        appCallbacks.InvokeOnAppThread(() => callback(), false);
    }
    public void InvokeOnUIThread(Action callback)
    {
        appCallbacks.InvokeOnUIThread(() => callback(), false);
    }
```

Finally lets go to your **MainPage.xaml.cs** and set the Grid for our Plugin

In the MainPage constructor add the following

```
    Windows_Ad_Plugin.Helper.Instance.SetGrid(DXSwapChainBackgroundPanel);
```

That's it! If we run the game the ads will now appear