# Unity Windows Ad Plugin

A simple, flexible, easy to use plugin for Unity3D that allows Ads to be displayed in Windows Phone 8.1 and Windows Store 8.1 XAML C# Apps.

- **Unity Setup**
- **Windows Phone 8.1 Setup**
- **Windows Store 8.1 Setup**
- **Unity and Visual Studio Quirks**

## Setting up the plugin:

## Unity

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

Add either the **Windows Store Ad** or **Windows Phone Ad** prefab (Found in *WindowsAdPlugin->Prefabs*)

Once you have added them into your scene you must enter the correct credentials (AdUnitID, ApplicationID etc), and the width and the height.

You position your ad with the horizontal and vertical alignment variables. For example, if the horizontal alignment is center and if the vertical alignment is top, the ad will be centered at the top of the screen.

**Note:** If the variables are not showing up in the Unity Inspector, ensure you are in the correct platform. You can set this by selecting File->Build Settings (Ctrl-Shift-B).

### Test Values

The default values within both prefabs will allow you to get test ads, only on devices however not emulators. Windows Store test values, Windows Phone test values for PubCenter test values.

### AdFiller Component

The AdFiller script is there for when no ad is available. It simple consists of an image and a URL. If the service fails to get an ad, the AdFiller will display your own image. If the user then clicks on the AdFiller, they will be directed to the supplied URL.

In order for the AdFiller to work you must add a reference to it on your ad object.
**Use this responsibly, and comply with all EULAs**

## DestroyAd Component

The DestroyAd script will remove the Ad from the screen when the gameobject the script is attached to is destroyed. Useful if you only want the ad to appear on certain screens.

## Player Settings

Prior to building, we need to do some set up.

Open your Player Settings (*File->Build Settings->Player Settings->Windows Store*)

First we need to add our plugin to the Unprocessed Plugins list, navigate to it *Misc->Unprocessed Plugins*
Add:

- Microsoft.Advertising.WinRT.UI.dll
- Microsoft.Advertising.WintRT.dll
- Windows_Ad_Plugin.dll

Add them in this order as well

We add the Windows_Ad_Plugin because if Unity attempts to build it out, it will throw Reference Rewriter errors.

The other two are there because they are fake dlls there so our Windows_Ad_Plugin for Windows Store will build. If they are removed, Unity will throw Missing References error because the Windows_Ad_Plugin requires them to exist.

Once that is done, we need to set up our Capabilities, which are found directly below the Unprocessed Plugins list.
Add:

- InternetClient
- InternetClientServer

**Once everything is setup, hit build.**

## Windows Phone 8.1

Build the game as **Windows Store-> XAML C# Solution-> Phone 8.1**

**Reference the Unity Quirks Section for any errors you encounter.**

After you deal with the Unity/Visual Studio quirks, we can get to plugin setup.

First, we need to change our References
Add a reference to the **Windows Advertising SDK**
To add the **Microsoft Advertising SDK for Windows Phone 8.1 (XAML)** reference, in the Solution Explorer right click on the *References* and click *Add Reference*. In the window that appears select **Windows-> Extensions-> Microsoft Advertising SDK for Windows Phone 8.1 (XAML).**

After that, remove the **Microsoft.Advertising.WinRT.UI.dll** and **Microsoft.Advertising.WintRT.dll** references. Unlike the Windows Store build, we do not need them.

Once that is done, remove the reference to **Windows_Ad_Plugin.dll**, and then add a new reference, using the **Windows_Ad_Plugin.dll** within the **Plugins/WP8**.

Next, we add in our hooks for the plugin.

Open the App.xaml.cs file and paste the following line into the App constructor

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

Below the constructor add this function, this connects our plugin to the UI and App threads which are needed to add the Ad to the game.

```
void appCallbacks_Initialized()
{
        Windows_Ad_Plugin.Dispatcher.InvokeOnAppThread = InvokeOnAppThread;
        Windows_Ad_Plugin.Dispatcher.InvokeOnUIThread = InvokeOnUIThread;
}
```

Then add the actual functions

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

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

Finally, we give the plugin a reference to the DXSwapChainPanel (or any other grid, the plugin was written in such a way that any grid will work)

```
Windows_Ad_Plugin.Helper.Instance.BaseGrid = DXSwapChainPanel;
```

## Windows Store

Build the game out as a **Windows Store-XAML C# Solution-8.1**

**Reference the Unity Quirks Section for any errors you encounter.**

After dealing with any Unity/Visual Studio quirks, we can do plugin setup.

We need to add a reference to the **Windows Advertising SDK**.

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 add in our hooks for the plugin.

Open the App.xaml.cs file and paste the following line into the App constructor

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

Below the constructor add this function, this connects our plugin to the UI and App threads which are needed to add the Ad to the game.

```
void appCallbacks_Initialized()
{
        Windows_Ad_Plugin.Dispatcher.InvokeOnAppThread = InvokeOnAppThread;
        Windows_Ad_Plugin.Dispatcher.InvokeOnUIThread = InvokeOnUIThread;
 }
```

Then add the actual functions

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

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

Finally, we give the plugin a reference to the DXSwapChainPanel (or any other grid, the plugin was written in such a way that any grid will work)

```
Windows_Ad_Plugin.Helper.Instance.BaseGrid = DXSwapChainPanel;
```

## Unity and Visual Studio Quirks

### Shared

As of February 2015, there seems to be an acknowledged bug within Visual Studio when Windows Store Unity projects are created.

Link to the forum post here.

The error will be as follows:

*Requested Windows Runtime type 'UnityPlayer.AppCallbacks' is not registered.*

The short of it is this, within file explorer search for your AppManifest.xml file, and open it up. Will be located in your

- \bin\<Solution Configuration>\ <Solution Platform>

For example,

- \bin\ARM\Debug

Navigate through until you see this chunk of code:

```
<Extension Category="windows.activatableClass.inProcessServer">
        <InProcessServer>
                <Path>UnityEngineDelegates.dll</Path>
                <ActivatableClass ActivatableClassId="UnityEngineDelegates.FunctionDefsDictionary"
ThreadingModel="both" />
                <ActivatableClass ActivatableClassId="UnityEngineDelegates.PlatformInvoke"
ThreadingModel="both" />
        </InProcessServer>
</Extension>
```

And below that, add this:

```
<Extension Category="windows.activatableClass.inProcessServer">
        <InProcessServer>
                <Path>UnityPlayer.dll</Path>
                <ActivatableClass ActivatableClassId="UnityPlayer.AppCallbacks"
ThreadingModel="both" />
                <ActivatableClass ActivatableClassId="UnityPlayer.StreamListenerContext"
ThreadingModel="both" />
                <ActivatableClass ActivatableClassId="UnityPlayer.MulticastSocketContext"
ThreadingModel="both" />
                <ActivatableClass ActivatableClassId="UnityPlayer.XamlPageAutomationPeer"
ThreadingModel="both" />
        </InProcessServer>
</Extension>
```

## Windows Phone 8.1

After building you will encounter this error

```
App manifest is missing required element '/Package/Applications/Application/m3:VisualElements/m3:DefaultTile/m3:ShowNameOnTiles/m3:ShowOn'.
```

To fix this error, simply click on it and remove this line of code

```
<ShowNameOnTiles />
```

Check your Capabilities within *Package.appxmanifest* and ensure that the following is selected

- Internet (Client & Server)

It seems that the Player Settings Capabilites are not properly built out from Unity.