## Quick integration Guide (SDK v3.1)

AdDeals SDK for Windows & Windows Phone 8.1+ universal apps / Windows apps. C# XAML, Unity.

## HIGH QUALITY ADDEALS INTERSTITIAL ADS EXAMPLES:

#### WINDOWS TABLET/PC (Portrait & Landscape supported)



#### WINDOWS PHONE (Portrait & Landscape supported)



## PREREQUISITE:

- Register for free on <a href="http://www.addealsnetwork.com">http://www.addealsnetwork.com</a>
- Add your game/apps and get your appID/appKey

## SPECIAL NOTE FOR 3.5+ stars games (top games):

• 3.5+ games will benefit from **special high payout campaigns**, **including some guaranteed CPMs** targeted to top **Windows games**, which will guarantee higher revenues to studios and independent game developers on the Windows platform. I. Procedure to integrate AdDeals SDK from a Windows Universal app (native):

## 1. [Mandatory] Install AdDealsUniversalSDKW81.dll as a reference to

you Windows and/or Windows Phone Visual Studio project.

In your project references: click on 'Manage NuGet Packages', look for 'AdDeals' and click on Install



Or in your nuget Package Manager console, select your project and type:



2. [Mandatory] Get your mobile/desktop AppID/AppKey from your AdDeals account <a href="http://www.addealsnetwork.com">http://www.addealsnetwork.com</a>. Login, add your Windows app and get your credentials.

## 3. [Mandatory] Initialize AdDeals SDK @app launch.

Add the following code on you app first page displayed only, **usually in OnNavigatedTo()** (in most project, this is usually *MainPage.xaml* or *HubPage.xaml*)

```
protected async override void OnNavigatedTo(NavigationEventArgs e)
{
    await AdManager.InitSDK(this.LayoutRoot, "YOUR_APP_ID", "YOUR_APP_KEY");
}
```

YOUR_APP_ID	Provided by AdDeals, once you have added your app
YOUR_APP_KEY	Provided by AdDeals, once you have added your app
this.LayoutRoot	"LayoutRoot" is the name given to the Grid/Main Panel of your XAML page code. You can give the name you want, but this is required, for instance: <grid margin="0" x:name="LayoutRoot"></grid>

**For Unity apps older than 5.0**, OnNavigatedTo() code may not be called. Load this InitSDK() code once in another method that's called by Unity.

## Optional events when initializing AdDeals SDK (InitSDK())

AdManager.AppDownloadSourceDetected += AdManager\_AppDownloadSourceDetected; // [OPTIONAL - For exclusive offer / campaigns] AdManager.AppSessionSourceDetected += AdManager\_AppSessionSourceDetected; // [OPTIONAL - For exclusive offer / campaigns] AdManager.InitSDKSuccess += AdManager\_InitSDKSuccess; AdManager.InitSDKFailed += AdManager\_InitSDKFailed;

4. Show interstitial ads (from C# code, cross-promo supported)

Wherever you want to display AdDeals full screen interstitial ads (once you have initialized the SDK), you just have to call:

To preload/cache the ad (optional!)
AdDealsPopupAd cachePopupAd = await AdManager.GetPopupAd(this.LayoutRoot,
AdManager.AdKind.FULLSCREENPOPUPAD);
await cachePopupAd.CacheAd();

**To display the cached ad or a non-cached ad directly** (this will get the last ad cached) AdDealsPopupAd popupAdToShow = await AdManager.GetPopupAd(this.LayoutRoot, AdManager.AdKind.FULLSCREENPOPUPAD); popupAdToShow.ShowAd(); You can get notified about several events that can occur when interstitial ads are called, displayed or closed for instance. Here is how you can add events to get programmatic feedbacks.

### Optional events when trying to cache an ad (*CacheAd()*)

AdDealsPopupAd cachePopupAd = await AdManager.GetPopupAd(this.LayoutRoot, adKind); cachePopupAd.CacheAdSuccess -= CacheAdSuccess\_Event; cachePopupAd.CacheAdFailed -= CacheAdFailed\_Event; cachePopupAd.MinDelayBtwAdsNotReached -= MinDelayBtwAdsNotReached\_Event;

cachePopupAd.SDKNotInitialized -= SDKNotInitialized\_Event;

# Optional events when trying to show an ad (non-cached or cached ad) (ShowAd())

AdDealsPopupAd showAd = await AdManager.GetPopupAd(this.LayoutRoot, adKind); showAd.AdClosed += AdClosed\_Tap; // OPTIONAL. This is triggered when the popup ad is closed. showAd.AdClicked += AdClicked\_Tap; //OPTIONAL. This is triggered when an ad is clicked by end user. showAd.ShowAdFailed += ShowAdFailed\_Event; // OPTIONAL. This is triggered when no ad is available or an issue occurs (slow network connection...) showAd.ShowAdSuccess += ShowAdSucess\_Event; // OPTIONAL. This is triggered when an ad is displayed to end user. showAd.MinDelayBtwAdsNotReached += MinDelayBtwAdsNotReached Event; // OPTIONAL. This is triggered when you try to call more than 1 ad in a very short period of time (less than 3 sec). showAd.SDKNotInitialized += SDKNotInitialized\_Event; // OPTIONAL. This is triggered when you try to load an ad without initilizing the SDK. showAd.VideoRewardGranted += ShowAdVideoRewardGranted Event; // REQUIRED FOR REWARDED VIDEOS (not available at this stage) If you want to notify the end user that a video view has been completed.

### 5. Show AdDeals Wall (WPhone only, cross-promo supported)

/\* Hide status bar \*/
StatusBar statusBar = Windows.UI.ViewManagement.StatusBar.GetForCurrentView();
await statusBar.HideAsync();

AdDealsWall wall = AdManager.GetWallAd();
wall.AdClosed -= AdClosed\_Wall;
wall.AdClosed += AdClosed\_Wall;
wall.ShowAd();

/\* OPTIONAL - Delegated event when AdDeals ad is closed by end user \*/
private async void AdClosed\_Wall(object sender, EventArgs e) {
 StatusBar statusBar = Windows.UI.ViewManagement.StatusBar.GetForCurrentView();
 await statusBar.ShowAsync(); }

## 6. Show banner ads (XAML UserControl) – No cross-promo

Go to your XAML Page, where you want to show an ad banner and add: xmlns:addealsad="using:AdDealsUniversalSDKW81.Views.UserControls" in the namespaces in the Page item.

**a. Windows Phone** app supported XAML codes/Ad formats (copy and paste this code inside your XAML page)

#### 320x50 standard BANNER AD:

<addealsad:AdDealsBannerAd AdType="BANNER\_WINDOWS\_PHONE\_320x50"></addealsad:AdDealsBannerAd>

**b.** Windows Tablet/PC app supported XAML codes/Ad formats (copy and paste this code inside your XAML page)

#### 160x600 PC/Tablet WIDE SKYSCRAPER AD:

<addealsad:AdDealsBannerAd AdType="WIDE\_SKYSCRAPER\_WINDOWS\_TABLET\_PC\_160x600"></addealsad:AdDealsBannerAd>

#### 728x90 PC/Tablet LEADERBOARD AD:

<addealsad:AdDealsBannerAd AdType="LEADERBOARD\_WINDOWS\_TABLET\_PC\_728x90"></addealsad:AdDealsBannerAd>

#### 250x250 PC/Tablet SQUARE AD:

<addealsad:AdDealsBannerAd AdType="SQUARE\_WINDOWS\_TABLET\_PC\_250x250"></addealsad:AdDealsBannerAd>

#### 300x250 PC/Tablet MEDIUM RECTANGLE AD:

<addealsad:AdDealsBannerAd AdType="MEDIUM\_RECTANGLE\_WINDOWS\_TABLET\_PC\_300x250"></addealsad:AdDealsBannerAd>

### TESTING MODE AVAILABLE:

Before you start implementing the SDK, please note you can activate the **Ad testing mode** while developing. Go to your AdDeals account: <u>http://www.addealsnetwork.com</u>, then to your app, and app settings. Ad testing mode allows you to see test ads worldwide while developing.

- Ш. Procedure to call AdDeals SDK methods from a Unity3D app (bridge Windows - Unity) (We provide this as a helper without specific support for Unity but this can help you a lot to integrate AdDeals SDK)
- 1. Open your Unity Windows generated project in Visual Studio
- 2. Follow previous steps I. 1. to I. 3. by adding AdDeals SDK and initializing the SDK from your Windows Universal app C# code.
- 3. Here is an example of how you can show AdDeals interstitials from your Unity project:
  - a. Sample code

#### 1/ On the Unity side, create a static callback, for instance:

```
public class WinRTGateway
{
  public static Action OnShowAdDealsInterstitial = null;
  public static void ShowAdDealsInterstitial()
  {
    if (OnShowAdDealsInterstitial != null)
    {
      OnShowAdDealsInterstitial();
    }
  }
}
```

## **<u>2</u>/On the Universal project side** set this callback to a method that actually calls the

#### proper SDK code:

{

WinRTGateway.OnShowAdDealsInterstitial = OnShowAdDealsInterstitial;

#### An here is the SDK code on the Universal project side that shows the ad:

```
private void OnShowAdDealsInterstitial()
    AppCallbacks.Instance.InvokeOnUIThread(new AppCallbackItem(async () =>
    {
       try
```

```
{
    // Show previously loaded AdDeals interstitial
    AdDealsPopupAd popupAdToShow = await
AdManager.GetPopupAd(DXSwapChainPanel, AdManager.AdKind.FULLSCREENPOPUPAD);
    popupAdToShow.ShowAd();
    }
    catch (Exception) { }
}), false);
}
```

## 3/ On the Unity side

To show an ad invoke this action <u>from the Unity side</u> when needed (e.g. when level is completed, at launch, when the user lost a game...): WinRTGateway.ShowAdDealsInterstitial();

Please note you can use the exact same technic to cache ads, ar call other AdDeals SDK method since this is how you can call Windows Universal methods from Unity when no Unity plugin is available.

#### b. Give a name to your main Panel: here, DXSwapChainPanel

## The Universal project that is generated by Unity contains a default MainPage.xaml template page that looks like this:

<Page

```
x:Class="Template.MainPage"
IsTabStop="false"
xmlns="<u>http://schemas.microsoft.com/winfx/2006/xaml/presentation</u>"
xmlns:x="<u>http://schemas.microsoft.com/winfx/2006/xaml</u>"
xmlns:local="using:Template"
xmlns:d="<u>http://schemas.microsoft.com/expression/blend/2008</u>"
xmlns:mc="<u>http://schemas.openxmlformats.org/markup-compatibility/2006</u>"
mc:Ignorable="d">
```

```
<SwapChainPanel x:Name="DXSwapChainPanel">

<Grid x:Name="ExtendedSplashGrid">

<Image x:Name="ExtendedSplashImage" Source="Assets/SplashScreen.png"

VerticalAlignment="Center" HorizontalAlignment="Center"/>

</Grid>

</SwapChainPanel>
```

</Page>

The important part is the SwapChainPanel which is by default called **DXSwapChainPanel** (this is where you can change it if needed). The SwapChainPanel is responsible for rendering you XAML specfic UI (including all ad units used by the ad SDKs) above you 3D/2D game content.

## c. How to get notified about AdDeals SDK events to my Unity code/side?

*Let's assume:* **adClosed(object sender, object e)** is a callback method called after AdDeals SDK notifies that the user closed the interstitial ad. Then the following method is called inside the Windows Universal app and inside this method you can call the function in the Unity side:

```
Here are 2 examples to call an action from the Universal project side to the Unity side: private void AdClosed_Tap(object sender, object e)
```

```
{
    AppCallbacks.Instance.InvokeOnAppThread(new AppCallbackItem(() =>
    {
        WinRTGateway.interstitialAdState = WinRTGateway.InterstitialAdState.Closed;
        WinRTGateway.OptionallyDoSomethingUnitySide();
    }), false);
}
```

private void VideoAd\_Completed(object sender, object e)

```
{
    AppCallbacks.Instance.InvokeOnAppThread(new AppCallbackItem(() =>
    {
        WinRTGateway.VideoAdAvailable = false;
        WinRTGateway.videoAdState = WinRTGateway.VideoAdState.Completed;
        WinRTGateway.UnlockPremiumLevel();
    }), false);
}
```

This example is used to handle **adClosed** (or videoad\_completed – 'fake sample for understanding') from the Universal project side and signal to the Unity project side where the unity logic is handled. You can replace this easily with any event. The important part is that the WinRTGateway in the example is implemented on the Unity side an to access it you need to use the **AppCallbacks.Instance.InvokeOnAppThread()** method that can handle calls from the Universal project's UI thread to Unity's main thread. Alternatively you can use the exact opposite **AppCallbacks.Instance.InvokeOnUIThread()** method that handled calls from Unity's main thread to the Universal project's UI thread. The correct thread handling is important because without these methods all kind of nasty exceptions will occur.

Official docs for the AppCallbacks class: http://docs.unity3d.com/Manual/windowsstore-appcallbacks.html

• void InvokeOnAppThread(AppCallbackItem item, bool waitUntilDone)

Invokes a delegate on application thread, useful when you want to call your script function from UI thread.

#### • void InvokeOnUIThread(AppCallbackItem item, bool waitUntilDone)

Invokes a delegate on UI thread, useful when you want to invoke something XAML specific API from your scripts.