
POKKT SDK v3.0 Integration Guide for Unity3d 5.2.x (Windows Phone)

Contents:

1. Overview
2. Configuration Steps
3. Code Integration
4. SDK Setup on Unity3d
5. Video-Ads Functionalities
6. Debugging and Logging

1. Overview:

Thank you for choosing Pokkt SDK Plugin v3.0 for Unity3d. Pokkt SDK supports Video-Ad campaigns feature for Windows Phone 8.1. This document contains all the information that is needed by you to setup the SDK with your project.

Kindly note that these instructions are for Unity3d Version 5.2.x and above, older versions of Unity3d are not supported at this moment.

There is a sample app provided with the SDK. We will be referencing this app during the course of explanation in this document. It is suggested that you should check that app to understand the following process in detail.

2. Configuration Steps:

Everything you need is in the file provided: **PokktUnityDemo_WindowsPhone8.1.unpackage**
For your convenience, there is a separate zip file containing the Release and Debug builds of the plugin.

Step 1. Export the content of this package onto your Unity project. The contents of “Script/Pokkt” directory and the “Plugins” are mandatory. You can choose to ignore all the other stuff, but for the sake of explanation in this document, it is suggested that you should export everything.

Step 2. Ensure that you have **UnityWindowsExtension.dll** and **UnityWindowsExtension.pdb** files directly inside the **Assets/Plugins** folder. This is the editor’s “placeholder” plugin for the original Windows Phone plugin. Select the **UnityWindowsExtension.dll** and set the following on the Inspector Panel (reference the following image):

Select platforms for plugin: Uncheck “Any Player” and then check only “Editor”



Step 3. Next, ensure that you have another **UnityWindowsExtension.dll** and a **UnityWindowsExtension.pdb** inside **Assets/Plugins/WSA** folder, along with **PokktSDK.dll**, **PokktSDK.pdb** and a **PokktSDK** folder with some **resources** (you will use these resources later on). Now select **PokktSDK.dll** and set the following on the Inspector Panel (reference the following image):

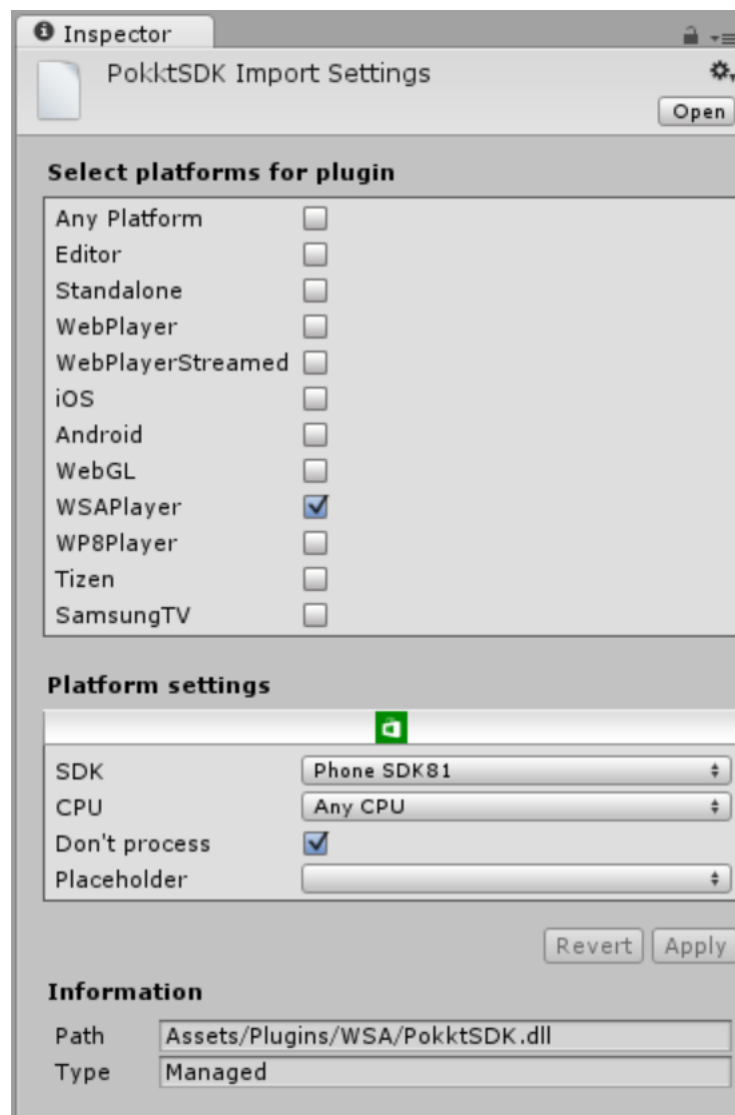
Select platforms for plugin: Check “WSAPlayer”

SDK: PhoneSDK8.1

CPU: AnyCPU

Don't Process: [CHECK]

Placeholder: [EMPTY]



Step 4. Select **UnityWindowsExtension.dll** inside **Assets/Plugins/WSA** folder and set the following on the Inspector Panel (reference the image below):

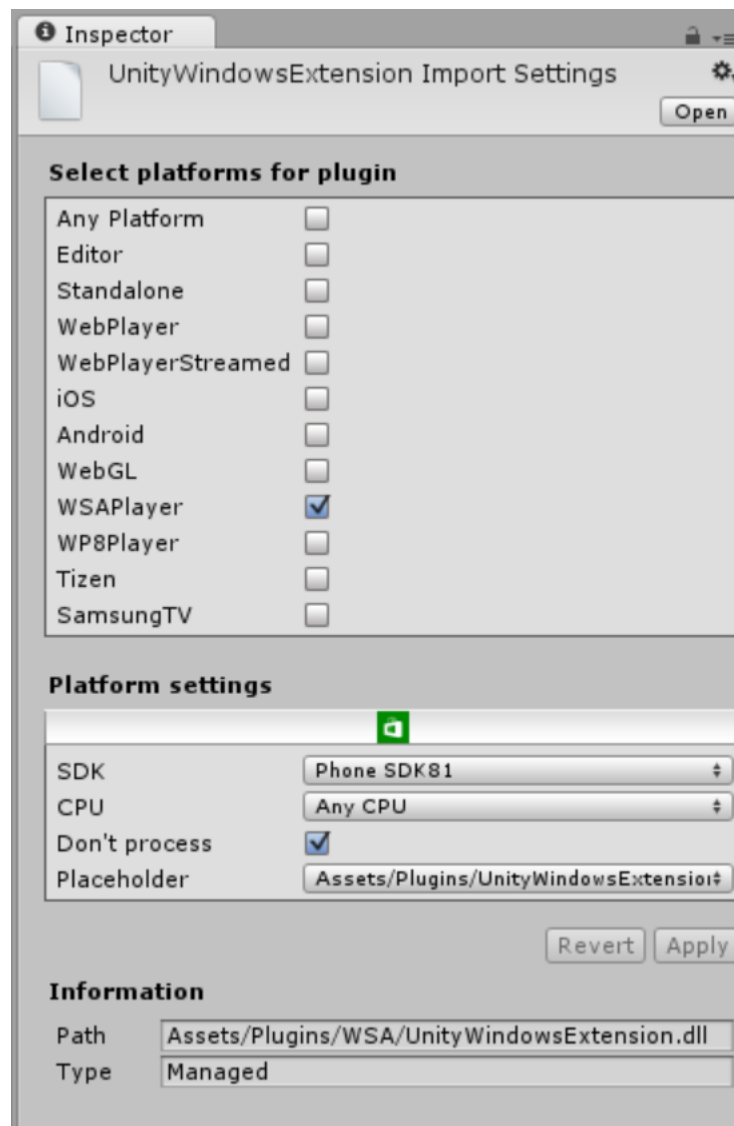
Select platforms for plugin: Check “WSAPlayer”

SDK: PhoneSDK8.1

CPU: AnyCPU

Don't Process: [CHECK]

Placeholder: [MAP TO: **Assets/Plugins/UnityWindowsExtension.dll**]

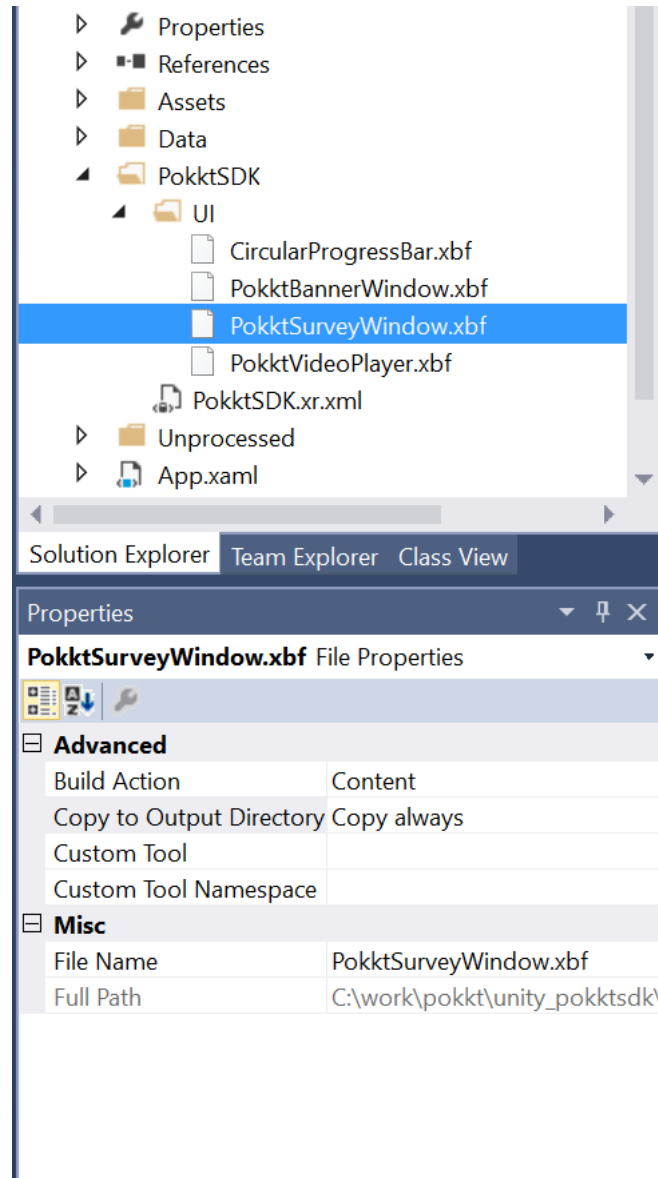


Note: Ignore any exception (System.Reflection.TargetInvocationException) you encounter while importing this DLL file.

Step 5. Once you have built/generated the C# Project, Add the **PokktSDK** folder inside **Assets/Plugins/WSA** folder to the project. This **PokktSDK** folder should have a **UI** folder inside it. Once added to the generated C# Project, select each of the XBF files inside the **UI** folder and set the following in the Properties Window (reference the image below):

Build Action: Content

Copy to Output Directory: Copy Always



This should integrate the Plugin with your project. Code related integration is mentioned below.

Note: Please **do not copy** the code points from this Doc/PDF file as it may introduce unwanted characters and space in your code. Instead please refer to sample app source code provided with the sample app.

3. Implementation Steps:

Common

1. For all invocation of Pokkt SDK developer will make use of methods available in **PokktManager** class. This class only have static methods.
2. In **PokktConfig** you can set **ApplicationId**, **SecurityKey** and **AutoCacheVideo**. which are must for all type of integrations.
3. Make sure to invoke **InitPokkt** method before you invoke any other methods from the **PokktManager**. This does not apply to session related methods namely **StartSession** and **EndSession**.
4. If you are doing server to server integration with Pokkt you can also set **ThirdPartyUserId** in **PokktConfig**.
5. Refer to Video sections below for information on additional parameters.
6. While in development please call **PokktManager.SetDebug(true)** to see Pokkt debug logs and toast messages. please make sure to change this to **false** for production build.

Session

1. We have option to start session for tracking for which we have **StartSession** and **EndSession** methods in **PokktManager**.
2. You should call **StartSession** at the start of his application and once only. You will need to call this method after setting required details like **ApplicationId**, **SecurityKey** and **IntegrationType**.
3. You should call **EndSession** at the end of his application and once only.

Video

1. In **PokktConfig** there are five additional parameters are made available for Video Ads, these are **AutoCacheVideo**, **SkipEnabled**, **DefaultSkipTime**, **ScreenName** and **Incentivised**.
2. **AutoCacheVideo** is required if you want to automatically cache video on user device. It has default value as true. if you set it as false then video will not be automatically cached and you will have to call **PokktManager.CacheVideoCampaign()** to start caching videos on device.

3. If you want to enable/disable the skip button on video screen please set **SkipEnabled** to **true/false**. The default value for **SkipEnabled** is false.
4. If you have enabled skipped button by setting **SkipEnabled** to **true**, then you can control after how many seconds the skip button will be visible in video by setting **DefaultSkipTime** to appropriate value. Since most videos will be 30 sec or less please set **DefaultSkipTime** as 10 or less. You can also give your own skip message by setting **CustomSkipMessage** on **PokktConfig**.
5. **ScreenName** has default value as **default** and can be used by you to give different screen-name for different places in your app where you are showing video-ads. You will control ad targeting based on these screen names which should match exactly with screen names defined in dashboard. **ScreenName** can not contain white spaces and only special characters allowed are hyphen and underscore.
6. You can choose to show video with or without incentive to user by setting **Incentivised** as **true/false**. Video gratification will only happen for incentivised playback.
7. You can call **PokktManager.IsVideoAvailable()** to check if the campaigns are available before you try to play video.
8. You can call **PokktManager.GetVideo()** to play video.
9. Reward user only from the **OnVideoGratified** event.

Export Logs

NOT YET IMPLEMENTED!

Optional Parameters

PokktConfig also has provision for developers to provide extra user data available with them to Pokkt. We currently support following data points: **Name, Age, Sex, MobileNo, EmailAddress, Location, Birthday, MaritalStatus, FacebookId, TwitterHandle, Education, Nationality, Employment** and **MaturityRating**.

5. Video-Ad Functionalities:

There are 7 events to manage the video caching and its playback, these are:

- VideoClosedEvent
- VideoDisplayedEvent
- VideoSippedEvent
- VideoCompletedEvent
- VideoGratifiedEvent
- DownloadCompletedEvent
- DownloadFailedEvent

(Ideally) Add handlers to these events in the Awake() method of your MonoBehaviour class. Below are the references on how to use them:

Reference on how to consume them:

```
// handle pokkt video ad events
PokktManager.Dispatcher.VideoClosedEvent +=
(string message) =>
{
};

PokktManager.Dispatcher.VideoSkippedEvent +=
(string message) =>
{
};

PokktManager.Dispatcher.VideoCompletedEvent +=
(string message) =>
{
};

PokktManager.Dispatcher.VideoGratifiedEvent +=
(string message) =>
{
};

PokktManager.Dispatcher.VideoDisplayedEvent +=
(string message) =>
{
};

PokktManager.Dispatcher.DownloadCompletedEvent +=
(string message) =>
{
    string text = "Video VC is: " + message;
};

PokktManager.Dispatcher.DownloadFailedEvent +=
(string message) =>
{
};
```

Remarks:

A video file is cached on user's device. You can set the auto-caching option in the beginning, as mentioned earlier in this document. In case of manual caching, call the following to start video caching:

```
PokktManager.CacheVideoCampaign(pokktConfig);
```

Before playing video or showing button to play video, Application should check whether video is cached or not by calling the following:

```
PokktManager.IsVideoAvailable();
```

You should listen to "DownloadCompletedEvent" to check whether download is completed or not, you can show the play buttons once you receive this event.

Furthermore, Application can decide to play video as incentivised (user will be gratified after watching complete video) or non- incentivised (user will not be gratified after watching complete video). You must provide the screen-name parameter for it. Followings are the method calls to me made:

```
pokktConfig.Incentivised = true/false;  
pokktConfig.ScreenName = "your_screen_name";
```

```
PokktManager.PlayVideo(pokktConfig, containerPage);
```

Next, you can listen to VideoGratifiedEvent to get the coins earned, if at all, by watching the last video.

6. Debugging and Logging

You can enable the SDK logs by setting the debugging option to true anytime. Ref.:

```
PokktManager.SetDebug(true/false);
```

You can use the following command to log some debug messages:

```
Logger.Log("pokkt init...");
```

This concludes the integration documentation. It is highly suggested that you should check the sample app that is provided to you to understand it better.