Pokkt SDK v1.0 for Windows Phone 8.1 Integration Guide

Contents:

1. Introduction:

Thank you for choosing Pokkt SDK for Windows Phone 8.1. This document contains all the information that is needed by you to setup the SDK with your project.

There is a sample app provided along with the SDK. We will be referencing to this app during the course of explanation in this document. It is suggested that you examine that app to understand the following process in detail. We assume that you are using Visual Studio 2013.

You can download our sample app (PokktSampleApp) on your Windows Phone 8.1 device. Follow the link mentioned below:

http://windowsphone.com/s?appid=be678e25-497a-4fbd-abf1-4a2fa66363de

OR

goo.gl/zub7kk

2. Installation:

The SDK comes in a zip file: PokktSDK_1.0_WP81.zip

Once you extract it, you will find two folders inside:

a. SDK
b. SampleApp

The 'SDK' folder contains the main SDK library (PokktSDK.dll) along with supporting files. You will be using these for your project. Add PokktSDK.dll in your project Reference.

The 'SampleApp' folder contains the Visual Studio 2013 solution (.sln) and project for the sample app, along with the source. You will be using this as reference.

Important: Please do not copy the code snippets from this PDF file as it may introduce unwanted characters and space in your code. Instead always refer to the sample app source code provided.

3. Code Integration:

Implementation Steps:

- Common:
    1. For all invocation of Pokkt SDK functionalities, the developer will make use of methods available in `PokktManager` class. This is a static class, thus, have static methods only.
    2. Make sure to call `PokktManager.InitPokkt` method before you make any other method calls, except session related methods (`PokktManager.StartSession` or `EndSession`).
    3. Make note of `PokktConfig` class! This holds all the values required by the SDK which you will be assigning to it. Then you will have to provide a properly filled instance of `PokktConfig` for almost every method calls that you will be making.
    4. In `PokktConfig`, you must assign the `ApplicationId`, `SecurityKey` and `IntegrationType`. These values are must for all type of integration.
    5. If you are doing a server-to-server integration with Pokkt, you can also mention `ThirdPartyUserId` in `PokktConfig`.
    6. Apart from above mentioned values/parameters, you can assign additional value/parameters based on your integration type.
    7. During the course of development, you can call `PokktManager.SetDebug(true)` to omit the SDK debug logs. Make sure to set it `false` for production build.

- Session:
    1. You should call `PokktManager.StartSession` at the start of your application and once only. You will have to provide your duly assigned `PokktConfig` object/instance for this method.
    2. You should call `PokktManager.EndSession` at the end of your application and once only.

- Video:
    1. These are video-ad related values/parameters that you can set:

        a. `AutoCacheVideo`: You can set it to true/false to control video caching on user's device. If you set it to true, the video will be cached automatically. Set it to false to disable auto-caching of video, but you have to manually call `PokktManager.CacheVideoCampaign` to cache the available video campaign. This is set to 'true' by default.

        b. `DefaultSkipTime`: Set this value (in seconds) to make it compulsory for a user to watch a video-ad for the given amount of time you have mentioned. A user then, may be able to skip the remaining video-ad if there is a 'skip' button available. It is recommended to set it to 10 or less.

        c. `SkipEnabled`: You can set it to true/false to show a 'skip' button over the video. A user can click this to skip the video-ad playback. You can control the time when you want to show it by setting above mentioned `DefaultSkipTime` parameter.

d. `BackButtonDisabled`: You can disabled the device's back-button by setting this `false`, while the video-ad playback is it progress.

e. `CustomSkipMessage`: If a video contains incentives, skipping it will present the user with a confirmation box, by default this box contains a generic message. You can change this message by setting this value to your desired message. You can also change the label of the buttons which will appear on the confirmation box by changing `VideoSkipYesLabel` and `VideoSkipNoLabel`.

f. `ScreenName`: It has a default value 'default'. It can be used by you to give different screen-name for different locations in your app where you want to show video-ads. You will control ad-targeting based on these screen-names which should match exactly with the screen-names defined in the Dashboard. Screen-names cannot contains whitespaces and only allowed special characters are: 'hyphen' and 'underscores'.

g. `Incentivised`: Set this value to 'true/false' to control the incentives that are being presented to the user after watching a video-ad. Video Gratification will happen only if you have set it to 'true'.

2. You will need to listen to the events mentioned in `VideoCampaignDelegate` to get all video-campaign related messages. This is discussed further is this document.

3. You can call `PokktManager.IsVideoAvailable` to check whether a video is available 'locally' for playback or not.

4. In order to play an available video, you have to call `PokktManager.PlayVideo`. You will have to provide your duly filled `PokktConfig` instance along with a reference to the `Page` control. This `Page` control will act as a parent container for the video.

- Optional Parameters:
  1. `PokktConfig` also has the provision for developers to provide extra user data available with them to Pokkt. Currently, the following data points are supported:
     a. `Name`
     b. `Age`
     c. `Sex`
     d. `MobileNo`
     e. `EmailAddress`
     f. `Location`
     g. `Birthday`
     h. `MaritalStatus`
     i. `FacebookId`
     j. `TwitterHandle`
     k. `Education`
     l. `Nationality`
     m. `Employment`
     n. `MaturityRating`

4. Video-ad Functionalities:

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

1. VideoClosedEvent
2. VideoDisplayedEvent
3. VideoSkippedEvent
4. VideoCompletedEvent
5. VideoGratifiedEvent
6. DownloadCompletedEvent
7. DownloadFailedEvent

Reference on how to consume them:

```
VideoCampaignDelegate.DownloadCompletedEvent += (float vc)
{
};

VideoCampaignDelegate.DownloadFailedEvent += (string message)
{
};

VideoCampaignDelegate.VideoDisplayedEvent += ()
{
};

VideoCampaignDelegate.VideoCompletedEvent += ()
{
};

VideoCampaignDelegate.VideoClosedEvent += (bool backPress)
{
};

VideoCampaignDelegate.VideoSkippedEvent += ()
{
};

VideoCampaignDelegate.VideoGratifiedEvent += (VideoResponse response)
{
        string coins = response.Coins;
};
```

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...");
```

Export Log File:

In order to export the log file generated by the SDK, two utility methods have been provided:

1. `PokktManager.ExportLogFile();`
2. `PokktManager.FinishExportingLogFile(file); // file is StorageFile`

You have to use these two in continuation. The first method is straightforward. When you call the `PokktManager.ExportLogFile()`, you are presented with a `FileSavePicker` message-box, select the desired location where you want to save the file and confirm it. Once you confirm, you will arrive at your application again and inside your app-entry class, override its `OnActivated` method and access the event-args parameter of this method. You will find a `StorageFile` object inside it. Take this and pass it to the second method mentioned above: `PokktManager.FinishExportingLogFile(file)`. Code Snippet:

```
protected override void OnActivated(IActivatedEventArgs args)
{
	base.OnActivated(args);

	try
	{
		FileSavePickerContinuationEventArgs fsArgs =
					args as FileSavePickerContinuationEventArgs;
		if (fsArgs != null)
		{
			StorageFile file = fsArgs.File;
			PokktManager.FinishExportingLogFile(file); // file is StorageFile
		}
	}
	catch (Exception e)
	{
		Logger.LogNoSave("failed to save file: " + e);
	}
}
```

Please check the implementation inside the provided SampleApp for better understanding of it.

Clearing Log File:

You can call `PokktManager.ClearLogFile()` to clear the log file.

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.