

# **POKKT SDK v3.0.0 Integration Guide for Xamarin (Android)**

---

## Contents:

1. Overview
2. Configuration steps
3. Implementation steps
4. Functionalities
  - I. Offerwall
  - II. Video
5. Debugging and Logging
6. Important Points

## 1. Overview

Pokkt Xamarin SDK v3.0.0 supports OfferWall as well as Video ad campaigns feature. With help of this document, any application developer / publisher can integrate either feature or both the features in their application. Please follow these steps as per your integration requirement(Video/OfferWall/Both).

## 2. Configuration Steps

All we need is the file provided: *PluginExtension.zip*. This zip file contains two file one is *PokktSDKPlugin.dll* file which you need to add it in project and 2nd is *res.zip*.

1. Add the *PokktSDKPlugin.dll* to your project's Reference directory.
2. Extract the contents of *res.zip* folder and copy them to respective folders in your project.
3. Add following permissions in your AndroidManifest, If not already there.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

4. Add the following activity in your AndroidManifest for OfferWall Integration

```
<activity android:name="com.app.pokktsdk.ShowOfferwallActivity"
    android:configChanges="keyboard|keyboardHidden|navigation|orientation|screenLayout|uiMode|screenSize"
    android:label="@string/app_name"
    android:windowSoftInputMode="adjustPan" >
</activity>
```

5. Add the following activity in your AndroidManifest for Video Integration

```
<activity android:name="com.app.pokktsdk.PlayVideoCampaignActivity"
    android:configChanges="keyboard|keyboardHidden|navigation|orientation|screenLayout|uiMode|screenSize"
    android:label="@string/app_name"
    android:screenOrientation="landscape"
    android:windowSoftInputMode="adjustPan" >
</activity>
```

## 6. Add Following Broadcast receiver for OfferWall Integration in AndroidManifest

```
<receiver android:name="com.app.pokktsdk.AppInstallBroadcastReceiver" >

    <intent-filter android:priority="1000" >

        <action android:name="android.intent.action.PACKAGE_INSTALL" />

        <action android:name="android.intent.action.PACKAGE_ADDED" />

        <data android:scheme="package" />

    </intent-filter>

</receiver>
```

## 7. Add Following meta tag for google play services (Goole play services is required and should be part of your project, if not, please add the GooglePlayService component in project.)

```
<meta-data android:name="com.google.android.gms.version" android:value="@integer/
google_play_services_version" />
```

## 8. Add a meta data tag for offerwall campaign. you will have to implement the *IOfferwallDelegate* interface in your project to listen for all offerwall related events. (refer to pokkt sample for example).

```
<meta-data android:name="offerwallDelegate" android:value="<com.pokkt.xamarin.OfferwallEvenstHandler>" />
```

## 9. Add a meta data tag for video campaign. you will have to implement the *IVideoDelegate* interface in your project to listen for all video related events (refer to sample app for example).

```
<meta-data android:name="videoDelegate" android:value="< com.pokkt.xamarin.VideoEventsHandler>" />
```

## 10. Add following Service and receiver in manifest for google analytics (Optional).

```
<receiver android:name="com.google.android.gms.analytics.AnalyticsReceiver" android:enabled="true">

    <intent-filter>

        <action android:name="com.google.android.gms.analytics.ANALYTICS_DISPATCH" />

    </intent-filter>

</receiver>

<service android:name="com.google.android.gms.analytics.AnalyticsService"

    android:enabled="true"

    android:exported="false"/>
```

## 11. Add following Service in manifest for in-app notification.

```
<service android:name="com.app.pokktsdk.notification.NotificationService"

    android:label="PokktNotificationService"

    android:exported="false"/>
```

### 3. Implementation Steps

- Common

1. For all invocation of Pokkt SDK developer will make use of methods available in *PokktManagerHandler* class. This class only have static methods.
2. In *PokktManagerHandler* you can set [\*SetApplicationId\*](#), [\*SetSecurityKey\*](#), [\*SetIntegrationType\*](#) and [\*SetAutoCacheVideo\*](#). which are must for all type of integrations
3. Before calling any other methods from the *PokktManagerHandler* please make sure that you have called the *InitPokkt* already.(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 [\*SetThirdPartyUserId\*](#) in *PokktManagerHandler*.
5. Apart from above mentioned parameters you can assign additional ones based on your integration type.(please refer to OfferWall and Video sections below.)
6. While in development please call *PokktManagerHandler.SetDebug(true)*; to see pokkt debug logs and toast messages. please make sure to change this to *PokktManagerHandler.SetDebug(false)*; for production build.
7. Please call *PokktManagerHandler.SendAppInfo()* to send your application installation information to Pokkt.
8. Android MinSDKVersion should be  $\geq 9$ .
9. To use google analytics, please set AnalyticsType and Analytics ID in *PokktManagerHandler*.

```
SetSelectedAnalyticsType(AnalyticsName.GOOGLE_ANALYTICS)
SetGoogleAnalyticsID("ID").
```

10. To use flurry analytics please set AnalyticsType and Flurry Application Key in *PokktManagerHandler*.

```
SetSelectedAnalyticsType(AnalyticsName.FLURRY)
SetFlurryApplicationKey("key").
```

11. To use mix panel analytics please set AnalyticsType and Mix PanelProject Token in *PokktManagerHandler*

```
setSelectedAnalyticsType(AnalyticsType.MIXPANEL)
setMixPanelProjectToken().
```

- Session

1. We have option to start session for tracking for which we have *StartSession* and *EndSession* methods in *PokktManagerHandler*.
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 *SetApplicationId*, *SetSecurityKey* and *SetIntegrationType*.
3. You should call *EndSession* at the end of his application and once only.

- OfferWall

1. In *PokktManagerHandler* for OfferWall you can set two additional parameters which are *SetOfferWallAssetValue* and *SetCloseOnSuccessFlag*. *SetOfferWallAssetValue* is only required if you only want to show campaign of certain value on offerwall. *SetCloseOnSuccessFlag* is required if you wish to close the OfferWall after user has completed one offer. It's default value is false.
2. Before calling another method for offerWall in *PokktManagerHandler*, please make sure that you have already called *InitPokkt* first.
3. You will need to implement *IOfferwallDelegate* interface as mentioned in [step 8](#) in [configuration steps](#).
4. To show OfferWall you can call *PokktManagerHandler.getCoins();*.
5. In the screen or activity where you have button to show offer wall, in that activity onResume you should call *PokktManagerHandler.GetPendingCoins();* so that you get a callback to award points to the user after he has come back to your game after finishing with OfferWall. You will get a callback for this call in your *IOfferwallDelegate* implementation class in method *EarnedCoins* or *CoinResponseFailed*
6. You can call *PokktManagerHandler.CheckCampaignAvailable();* to check whether the campaigns are available before showing OfferWall button to user. You will get a callback for this call in your *IOfferwallDelegate* implementation class in method *OnOfferwallCampaignCheck*.

- Video

1. In *PokktManagerHandler* for Video you can set five additional parameters which are *SetAutoCacheVideo*, *SetSkipEnabled*, *SetDefaultSkipTime*, *SetScreenName* and *SetIncentivised*.
2. *SetAutoCacheVideo* 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 *PokktManagerHandler.CacheVideoCampaign()*; to start caching videos on device.
3. If you want to enable/disable the skip button on video screen please set *SetSkipEnabled* as true/false. The default value for *SetSkipEnabled* is false.
4. If you have enabled skipped button by setting *SetSkipEnabled* as true then you can control after how many seconds the skip button will be visible in video by setting *SetDefaultSkipTime* to appropriate value. Since most videos will be 30 sec or less please set *SetDefaultSkipTime* as 10 or less. You can also give your own skip message by setting *SetCustomSkipMessage* on *PokktManagerHandler*.
5. *SetScreenName* 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. *SetScreenName* 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 *SetIncentivised* as true or false. Video gratification will only happen for incentivised playback.
7. You can disable the back button while video is playing by setting *SetBackButtonDisabled* on *PokktManagerHandler*.
8. You will need to create *IVideoDelegate* implementation class as mentioned in [step 9](#) in [configuration steps](#).
9. You can call *PokktManagerHandler.IsVideoAvailable()* to check if the campaign are available before you try to play video.
10. You can call *PokktManagerHandler.GetVideo()*; to play video.
11. You will get different callbacks as given in *IVideoDelegate* implementation class for video playback.

12. Please reward user only from the *OnVideoGratified* method in *IVideoDelegate* implementation class.

- Export Logs

1. Developer should call *PokktManagerHandler.ExportLog()* to export the Pokkt SDK logs to folder of your choice.
2. This API shows a folder chooser dialog where user can choose a particular folder.
3. User can also create a new folder where user wants to export the logs

### Optional Parameters

- *PokktManagerHandler* also has provision for developers to provide extra user data available with them to pokkt. We currently support following data points: *SetName, SetAge, SetSex, SetMobileNo, SetEmailAddress, SetLocation, SetBirthday, SetMaritalStatus, SetFacebookId, SetTwitterHandle, SetEducation, SetNationality, SetEmployment and SetMaturityRating.*



- In-App Notifications

Developer can add In-App notifications in their dashboard.

Add Notification

**Basic**

Name

App

Platform

☐ iOS ☐ Android ☒ All

**Filters**

Countries

App Version

Last Seen

**Schedule**

Repeat

Repeat

Dates

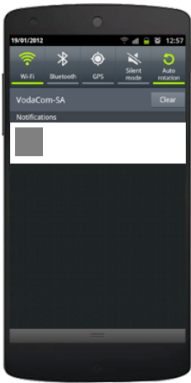
1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31					

Time

O'clock

**Message**

Message



Cancel

Save

Repeat schedule can be daily, weekly monthly.

Daily Repeat can have options like frequency of repeat and time in hours of notification.

The screenshot shows two forms. The top form, titled 'Schedule', has a 'Repeat' dropdown set to 'Daily'. Below it, 'Every' is set to '1' with a unit of 'Day(s)'. The 'Time' is set to '12' with a unit of 'O'clock'. The bottom form, titled 'Message', has a 'Title' field, a large text area, and an 'Add Image' button. To the right of the message form is a preview of a smartphone screen showing a notification for 'VodaCom-SA' with a 'Clear' button. At the bottom right of the forms are 'Cancel' and 'Save' buttons.

Weekly repeat can have options like frequency of repeat in weeks, days of repeat and time in hours of notification.

The screenshot shows two forms. The top form, titled 'Schedule', has a 'Repeat' dropdown set to 'Weekly'. Below it, 'Every' is set to '1' with a unit of 'Week(s)'. There is a grid of days: Mon, Tue, Wed, Thu, Fri, Sat, and Sun. The 'Time' is set to '12' with a unit of 'O'clock'. The bottom form, titled 'Message', has a 'Title' field, a large text area, and an 'Add Image' button. To the right of the message form is a preview of a smartphone screen showing a notification for 'VodaCom-SA' with a 'Clear' button. At the bottom right of the forms are 'Cancel' and 'Save' buttons.

Monthly repeat can have options like frequency of repeat in months dates of repeat and time in hours for notification .

Repeat: Monthly

Every 1 Month(s)

Dates:

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31					

Time: 12 O'clock

**Message**

Message: Title

Add Image

Cancel Save

For don't repeat case, there are options like dates and time in hour for notification.

The notifications are listed and can be edited. Notifications can also be deactivated/activated.

#### List Notifications

Id	Name	App Id	Header	Status	Action
1	push	1000125	Hi There	ACTIVE	Edit Deactivate
2	in app	1000125	Hi There	ACTIVE	Edit Deactivate

## 4. Functionalities:

### 4.1 Offerwall

There are five methods in *IOfferwallDelegate* which need to implement in class.

1. CoinResponseFailed.
2. EarnedCoins : with earned points
3. EarnedCoins : with earned points and transaction ID.
4. OnOfferwallCampaignCheck
5. OnOfferwallClosed

Implement these methods where you want to implement *IOfferwallDelegate* interface.

```
void IOfferwallDelegate.CoinResponseFailed(string message) {
}

void IOfferwallDelegate.EarnedCoins(int coins) {
}

void IOfferwallDelegate.EarnedCoins(int coins, string transaction_id) {
}

void IOfferwallDelegate.OnOfferwallCampaignCheck(bool status) {
}

void IOfferwallDelegate.OnOfferwallClosed() {
}
```

### There are two ways to invoke offerwall:

**Open Asset Value:** In this case, POKKT platform provides all offers with any asset value.

Sample code snippet:

```
PokktManagerHandler.SetOfferWallAssetValue = "";
PokktManagerHandler.GetCoins();
```

**Fixed Asset Value:** In this case, POKKT platform provides all offers with fixed asset value.

Sample code snippet:

```
PokktManagerHandler.SetOfferWallAssetValue = "400";
PokktManagerHandler.GetCoins();
```

**Pending Coins:** In case after completing activity, if status of transaction is pending, then call *GetPendingCoins()* method. You should always call this method from your activity's onResume method so that you can check for the pending coins for user.

Sample code snippet:

```
PokktManagerHandler.GetPendingCoins();
```

**Check for campaign availability:** If you are using optional meta-tag as mentioned in Step 4 of manifest changes, you can call the following to check for campaign availability:

```
PokktManagerHandler.CheckCampaignAvailable ();
```

And once you call this then *OnOfferwallCampaignCheck* function will get called with the status of either true or false.

And if you want to listen offerwall close event then you need to set *PokktManagerHandler SetCloseOnSuccessFlag(true);* and on close of Offerwall *OnOfferwallClosed()* method will get called.

#### 4.2 Video:

There are five methods in *IVideoDelegate* which need to implement in class.

1. *OnDownloadCompleted.*
2. *OnDownloadFailed.*
3. *OnVideoClosed.*
4. *OnVideoCompleted.*
5. *OnVideoDisplayed.*
6. *OnVideoGratified.*
7. *OnVideoSkipped.*

Implement these methods where you want to implement *IOfferwallDelegate* interface.

```
void IVideoDelegate.OnDownloadCompleted (float vc) {
    }

void IVideoDelegate.OnDownloadFailed () {
    }

void IVideoDelegate.OnVideoClosed () {
    }

void IVideoDelegate.OnVideoCompleted () {
    }
```

```

void IVideoDelegate.OnVideoDisplayed () {
}

void IVideoDelegate.OnVideoGratified (String earnedPoint) {
}

void IVideoDelegate.OnVideoSkipped () {
}

```

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:

```
PokktManagerHandler.CacheVideoCampaign();
```

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

```
PokktManagerHandler.IsVideoAvailable();
```

**OnDownloadCompleted** method will get called once video gets cached locally on device.

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 and set the incentive as true or false. Followings are the method calls to be made:

```

PokktManagerHandler.SetScreenName = "screenname";
PokktManagerHandler.SetIncentivised(<true/false>);
PokktManagerHandler.GetVideo();

```

Next, **OnVideoGratified** method will get called once you earn the points after completing the video.

## 5. Debugging and Logging

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

```
PokktManagerHandler.SetDebug(<true/false>);
```

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

```
PokktManagerHandler.ShowLog("pokkt init...");
```

You can use the following command to display a message as Toast on android device:

```
PokktManagerHandler.ShowToast("pokkt init...");
```

## 6. Important Points

- Please do not copy the code points from this pdf as it may introduce unwanted characters and space in your code. Instead please refer to sample app source code in pokkt bundle.
- Please also refer to sample app source code for better understanding of implementation.