

---

# **POKKT SDK v4.0 Integration Guide for Cordova/PhoneGap (Android)**

---

## Contents:

---

1. Overview
2. Configuration Steps
  - a. AndroidManifest.xml
3. Implementation Steps
4. Functionalities:
  1. Offerwall
  2. Video
5. Debugging and Logging

---

## 1. Overview:

---

Thank you for choosing Pokkt SDK Plugin v4.0 for Cordova/PhoneGap. Pokkt SDK supports Offerwall as well as Video-Ad campaigns feature. This document contains all the information that is needed by you to setup the SDK with your project. Please follow these steps as per your integration requirement (Video/Offerwall/Both). The current plugin supports mediation for various third party ad-networks. These are:

- AdColony
- AppLovon
- Chartboost
- Fyber
- InMobi
- SuperSonic
- UnityAds
- TapJoy
- Vungle
- FusePowered (not available for iOS)
- MoPub (not available for iOS)

A separate set of documents is provided for each of these, explaining the implementation process.

Kindly note that these instructions are for Cordova Version 4.x and above, older versions of are not supported.

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.

**Note:** Please do not copy the code points from this 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.

---

## 2. Configuration Steps:

---

Extract the provided file “PokktCordovaPlugin.zip” into a directory. Execute the following command from your terminal:

```
$phonegap plugin add /<path-to-plugin-directory>/PokktCordovaPlugin/
```

This should install the plugin with your project and you should be able to use it inside your project.

**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.

### AndroidManifest.xml

AndroidManifest.xml is **pre-setup** and should there is nothing else you are required to do yourself. Just make sure that in your config.xml, “android-minSdkVersion” is set to 14 or above as mentioned below:

```
<preference name="android-minSdkVersion" value="14" />
```

### Overview of the items added to the manifest:

#### Added required permission:

Required permissions:

- INTERNET
- ACCESS\_NETWORK\_STATE

Recommended Permissions:

- READ\_PHONE\_STATE
- WRITE\_EXTERNAL\_STORAGE
- WAKE\_LOCK

Optional Permissions:

- WRITE\_CALENDAR

---

## Added Activity definitions:

Offerwall:

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

Video:

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

## Added BroadcastReceiver (required only for offerwall)

```
<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>
```

## Added meta-data information

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

## About Google Play Services SDK

Please follow the instructions below to include the Google Play Services SDK

<http://developer.android.com/google/play-services/setup.html>

*Why is this required?*

Google has now changed policy w.r.t recognizing the devices. It no longer allows the developer to read the Android\_ID. Instead a new Advertisers ID is needed to be use.

Please find more details below

<https://developer.android.com/google/play-services/id.html>



---

### 3. Implementation Steps:

---

#### Common

1. For all invocation of Pokkt SDK developer will make use of methods available in **pokktNativeExtension.js** file inside **PokktExtension** object.
2. In **PokktExtension** you can set **setApplicationId**, **setSecurityKey**, **setIntegrationType** and **setAutoCacheVideo**. which are must for all type of integrations
3. Make sure to invoke **initPokkt** method before you invoke any other methods from the **PokktExtension**. 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 **PokktExtension**.
5. Apart from above mentioned parameters you can assign additional ones based on your integration type. Refer to OfferWall and Video sections below.
6. While in development please call **PokktExtension.setDebug(true)** to see Pokkt debug logs and toast messages. please make sure to change this to **false** for production build.
7. Please call **PokktExtension.sendAppInfo()** to send your application installation information to Pokkt.
8. Android **MinSDKVersion** should be  $\geq 9$ .
9. To use google analytics, please set **setAnalyticsType** and **setAnalyticsID** in **PokktExtension**.
10. To use flurry analytics please set **setAnalyticsType** and **setFlurryApplicationKey** in **PokktExtension**.
11. To use mix panel analytics please set **setAnalyticsType** and **setMixPanelProjectToken** in **PokktExtension**.

#### Session

1. We have option to start session for tracking for which we have **startSession** and **endSession** methods in **PokktExtension**.
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 **PokktExtension** for Offerwall you can set two additional parameters which are **setOfferWallAssetValue** and **setCloseOnSuccessFlag**. Setting of **setOfferWallAssetValue** is only required if you only want to show campaign of certain value on the Offerwall. **setCloseOnSuccessFlag** is required if you wish to auto-close the Offerwall after user has completed one offer. It's default value is false.
2. Make sure to call **initPokkt** before calling another method for Offerwall in **PokktExtension**.
3. You will need to implement **IOfferwallDelegate** interface as mentioned in Step-4 in configuration steps.
4. To show Offerwall you can call **PokktExtension.getCoins()**.
5. In the screen or activity where you have button to show Offerwall, in that activity **onResume** you should call **PokktExtension.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 **PokktExtension.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 **PokktExtension** 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 **PokktExtension.CacheVideoCampaign()** to start caching videos on device.
3. If you want to enable/disable the skip button on video screen please set **setSkipEnabled** to **true/false**. The default value for **setSkipEnabled** is false.
4. If you have enabled skipped button by setting **setSkipEnabled** to **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 **PokktExtension**.

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** to **true/false**. Video gratification will only happen for incentivised playback.
7. You can disable the back button while video is playing by setting **setBackButtonDisabled** on **PokktExtension**.
8. You will need to create **IVideoDelegate** implementation class as mentioned in Step-4 in configuration steps.
9. You can call **PokktExtension.isVideoAvailable()** to check if the campaigns are available before you try to play video.
10. You can call **PokktExtension.getVideo()** to play video.
11. You will get different callbacks as given in **IVideoDelegate** implementation class for video playback.
12. Reward user only from the **onVideoGratified** method in **IVideoDelegate** implementation class.

#### Export Logs

1. Developer should call **PokktExtension.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

**PokktExtension** 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

Min

Max

**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

Title

Add Image



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 image shows two stacked configuration screens. The top screen, 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' O'clock. The bottom screen, titled 'Message', has a 'Title' field, a large text area, and an 'Add Image' button. To the right is a preview of a smartphone displaying the notification. At the bottom 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 image shows two stacked configuration screens. The top screen, 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 row of seven buttons for days of the week: 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', and 'Sun'. The 'Time' is set to '12' O'clock. The bottom screen, titled 'Message', is identical to the one in the first image, with a 'Title' field, a text area, an 'Add Image' button, and a smartphone preview. At the bottom 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	<a href="#">Edit</a> <a href="#">Deactivate</a>
2	in app	1000125	Hi There	ACTIVE	<a href="#">Edit</a> <a href="#">Deactivate</a>

---

## 4. Functionalities:

---

### 4.1 Offerwall

There are five events to listen too. These are:

- CoinResponse
- CoinResponseWithTransId
- CoinResponseFailed
- CampaignAvailability
- OfferwallClosed

(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:

```
document.addEventListener('CoinResponse',
    this.onCoinResponse, false);

document.addEventListener('CoinResponseWithTransId',
    this.onCoinResponseWithTransId, false);

document.addEventListener('CoinResponseFailed',
    this.onCoinResponseFailed, false);

document.addEventListener('CampaignAvailability',
    this.onCampaignAvailability, false);

document.addEventListener('OfferwallClosed',
    this.onOfferwallClosed, false);

...

onCoinResponse: function(params) {
    console.log('onCoinResponse: ' + params.param);
    var labelPointsEarned =
        document.getElementById('labelPointsEarned');
    labelPointsEarned.innerHTML = params.param;
},

onCoinResponseWithTransId: function(params) {
    console.log('onCoinResponseWithTransId: ' + params.param);
    var labelPointsEarned =
        document.getElementById('labelPointsEarned');
    labelPointsEarned.innerHTML = params.param;
},

onCoinResponseFailed: function(params) {
    console.log('onCoinResponseFailed: ' + params.param);
},
```

---

```
onCampaignAvailability: function(params) {  
    console.log('onCampaignAvailability: ' + params.param);  
},  
onOfferwallClosed: function(params) {  
    console.log('onOfferwallClosed: ' + params.param);  
},
```

**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:

```
var pe = window.plugins.pokktExtension;  
pe.getCoins(0);
```

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

Sample code snippet:

```
var pe = window.plugins.pokktExtension;  
pe.getCoins(<asset_value>);
```

**Pending Coins:** In case after completing activity, if status of transaction is pending, then call getPendingCoins() method.

```
var pe = window.plugins.pokktExtension;  
pe.getPendingCoins();
```

**Check for campaign availability:** If you are using optional meta-tag as mentioned in Step 5, you can call the following to check for campaign availability:

```
var pe = window.plugins.pokktExtension;  
pe.checkOfferWallCampaign();
```

The result will be noticed with the event:

CampaignAvailability

Moreover, you can listen to the following event to know whether offerwall has been closed or not:

OfferwallClosed

---

## 4.2 Video:

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:

```
document.addEventListener('DownloadCompleted',
    this.onDownloadCompleted, false);

document.addEventListener('DownloadFailed',
    this.onDownloadFailed, false);

document.addEventListener('VideoDisplayed',
    this.onVideoDisplayed, false);

document.addEventListener('VideoClosed',
    this.onVideoClosed, false);

document.addEventListener('VideoSkipped',
    this.onVideoSkipped, false);

document.addEventListener('VideoCompleted',
    this.onVideoCompleted, false);

document.addEventListener('VideoGratified',
    this.onVideoGratified, false);

onDownloadCompleted: function(params) {
    console.log('onDownloadCompleted: ' + params.param);

    // set button state
    videoController.toggleButtons(true);
},

onDownloadFailed: function(params) {
    console.log('onDownloadFailed: ' + params.param);

    // set button state
    videoController.toggleButtons(false);
},
```

---

```

onVideoDisplayed: function(params) {
    console.log('onVideoDisplayed: ' + params.param);

    // set button state
    videoController.toggleButtons(false);
},

onVideoClosed: function(params) {
    console.log('onVideoClosed: ' + params.param);

    var pe = window.plugins.pokktExtension;
    if (pe.getAutoCaching()) {

        // set button state
        videoController.toggleButtons(true);
    } else {
        var buttonStartCaching =
            document.getElementById('buttonStartCaching');
        buttonStartCaching.disabled = false;
    }
},

onVideoSkipped: function(params) {
    console.log('onVideoSkipped: ' + params.param);
},

onVideoCompleted: function(params) {
    console.log('onVideoCompleted: ' + params.param);
},

onVideoGratified: function(params) {
    console.log('onVideoGratified: ' + params.param);
    var labelPointsEarned =
        document.getElementById('labelPointsEarned');
    labelPointsEarned.innerHTML = params.param;
},

```

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:

```

var pe = window.plugins.pokktExtension;
pe.cacheVideoCampaign();

```

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

```

var pe = window.plugins.pokktExtension;
pe.checkIsVideoAvailable(function(isAvailable) {
    console.log('checkIsVideoAvailable result: ' + isAvailable);
});

```

You should listen to "DownloadCompleted" 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 incent (user will be gratified after watching complete video) or non-incent (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:

```
var pe = window.plugins.pokktExtension;  
if (incentivised)  
    pe.getVideo('sampleApp');  
else  
    pe.getVideoNonIncent('sampleApp');
```

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

---

## 5. Debugging and Logging

---

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

```
pe.setDebug(true);
```

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

```
pe.showLog('sampleApp');
```

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

```
pe.showToast('sampleApp');
```

---

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.