POKKT SDK v4.1 Integration Guide for Cordova/PhoneGap (iOS)

Contents:

1. Overview:

Thank you for choosing Pokkt SDK Plugin v4.1 for Cordova/PhoneGap. Pokkt SDK supports Video/Interstitial-Ad campaigns feature. This document contains all the information that is needed by you to setup the SDK with your project. The current plugin supports mediation for various third party ad-networks. These are:

- AdColony
- AppLovin
- Chartboost
- Fyber
- InMobi
- SuperSonic
- UnityAds
- TapJoy
- Vungle
- AdMob
- Facebook
- MoPub

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 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.

2. Configuration Steps:

Step 1. 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.

Step 2. Once your project is built and the XCode project is exported, open this exported project using XCode. Ensure the followings frameworks are present(linked) inside project setting's "**Build-Phases -> Link Binaries With Libraries**", if not than add them manually:

- **CoreData**.framework
- **Foundation**.framework
- **MediaPlayer**.framework
- **SystemConfiguration**.framework
- **UIKit**.framework
- **CoreTelephony**.framework
- **EventKit**.framework
- **AdSupport**.framework
- **CoreGraphics**.framework

Step 3. Ensure that "**-ObjC**" flag is set inside project setting's "**Build Settings -> Other Linker Flags**".

Step 4. In order to use PokktSDK's background fetch functionality, enable "**Capabilities -> Background Modes -> Background Fetch**" inside project settings. Then write the following code-snippet inside "**didFinishLaunchingWithOptions**" method of app-delegate class:

```
[application setMinimumBackgroundFetchInterval:
UIApplicationBackgroundFetchIntervalMinimum];
```

Step 5. Further, implement/update the background-fetch delegate methods in app-delegate class. Invoke "**callBackgroundTaskCompletionHandler**" method from "**performFetchWithCompletionHandler**". Observe the following code-snippet for reference:

```
-(void)application:(UIApplication *) application
performFetchWithCompletionHandler:
(void(^)(UIBackgroundFetchResult))completionHandler
{
  [PokktManager callBackgroundTaskCompletionHandler:
        ^(UIBackgroundFetchResult result)
        {
              completionHandler(result);
  }];
}
```

Step 6. In order to enable local notifications for **InApp Notifications**, mention the following inside "**didFinishLaunchingWithOptions**" method of the app-delegate class:

```
[application setMinimumBackgroundFetchInterval:
       UIApplicationBackgroundFetchIntervalMinimum];

UIUserNotificationSettings *settings =
[UIUserNotificationSettings settingsForTypes:
  (UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeSound |
                  UIRemoteNotificationTypeAlert)
  categories:nil];

[application registerUserNotificationSettings:settings];
```

Step 7. Invoke "**inAppNotificationEvent**" if the user taps on local notification, do this in the "**didReceiveLocalNotification**" inside app-delegate class. Check the following reference:

```
-(void)application:(UIApplication*) application
didReceiveLocalNotification:(UILocalNotification*) notification
{
  [PokktManager inAppNotificationEvent:notification];
}
```

Step 8. Ensure that **Enable Bitcode** is set to **false** in the generated project's **Build Settings**.

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 **pokktNativeExtension.js** file using **PokktExtension** object.

2. Create a **PokktConfig** object using c**reatPokktConfig()** method**,** then provide **applicationId** and **securityKey,** these are must for initializing Pokkt.

3. Once you have your **pokktConfig** ready, invoke **initPokkt** method before you invoke any other methods from the **PokktExtension.** This does not apply to session related methods namely **startSession** and **endSession** and few utility methods.

4. In order to know whether Pokkt is initialized or not, listen to '**PokktInitialised'** event, if the provided Boolean parameter is true only then move ahead with other operations.

5. If you are doing server to server integration with Pokkt you can also set **thirdPartyUserId** in **pokktConfig** object.

6. Apart from above mentioned parameters you can assign additional ones based on your integration type. Refer to Offerwall and Video sections below.

7. While in development please call **PokktExtension.setDebug(true)** to see Pokkt debug logs and toast messages. Make sure to change this to **false** for production build.

8. Call **PokktExtension.notifyAppInstall()** to log your application installation information with Pokkt.

9. Call **PokktExtension.trackIAP(details)** to log any in-app purchase details with Pokkt. Accepted values are (all caps): "NONE", "GOOGLE", "IOS", "AMAZON".

10. To use google analytics, please set **selectedAnalyticsType** to "GOOGLE_ANALYTICS" and **googleAnalyticsID** in **pokktConfig**.

11. To use flurry analytics please set **selectedAnalyticsType** to "FLURRY" and **flurryApplicationKey** in **pokktConfig**.

12. To use mix panel analytics please set **selectedAnalyticsType** to "MIXPANEL" and **mixPanelProjectToken** in **pokktConfig**.

13. To use mix panel analytics please set **selectedAnalyticsType** to "FABRIC".

Session

1. Invoke **PokktExtension.startSession()** at the start of his application and once only.
2. You should call **PokktExtension.endSession()** at the end of his application and once only.

Rewarded/Non-Rewarded Ads

1. You will have to configure an **adConfig** object to request for any ad to be displayed. Get one using **createAdConfig()** method. It also provides options for customizing your ad-screen. It is recommended to have different **adConfig** objects for each screens. Followings are the values that you can set with **adConfig**:
    - **screenName** (Required): This controls the placement of ads and can be created on Pokkt Dashboard.
    - **isRewarded** (Required): Requested ad type. Ad gratification will happen only for rewarded ads.
    - **backButtonDisabled**: Disable 'back' button press while on ad-screen.
    - **defaultSkipTime**: If ad-skipping is allowed, this provides the seconds it will wait before the skip button appears.
    - **shouldAllowSkip**: Whether skipping-ad is allowed or not. If set to 'false', user will be forced to watch the ad till it finishes.
    - **shouldAllowMute**: Whether to allow sound-mute while on ad-screen, it controls the 'mute' button. Cannot contains whitespaces and only special characters allowed are hyphens (-) and underscores (_).
    - **shouldConfirmSkip**: Controls whether to show the skip-confirmation dialog box. If set to 'false', the ad will be silently closed without prompting for confirmation
    - **skipConfirmMessage**: The message that will appear on skip-confirmation dialog box.
    - **skipConfirmYesLabel**: 'Yes' Label of skip-confirmation dialog box.
    - **skipConfirmNoLabel**: 'No' Label of skip-confirmation dialog box.
    - **skipTimerMessage**: The message on countdown-timer before the skip button appears. The message must contain a '##'-placeholder to show timer value.

- **incentiveMessage**: If set, the message will be displayed while prompting user to watch the ad for certain time before it can be rewarded.

2. Invoke **PokktExtension.cacheAd(adConfig)** to cache the ad on device. Cached-ads provide better user experience than streaming-ads

3. You can call **PokktExtension.checkAdAvailability(adConfig)** to check if the ad-campaigns are available or not. The result will be notified via **'AdAvailability'**.

4. Invoke **PokktExtension.showAd(adConfig)** to show ad.

5. Following are ad-related events, you can refer to the provided sample-code to understand the ideal implementation on how to consume these:

    - 'AdCachingCompleted'
    - 'AdCachingFailed'
    - 'AdClosed'
    - 'AdCompleted'
    - 'AdDisplayed'
    - 'AdSkipped'
    - 'AdGratified'
    - 'AdAvailability'

6. Reward user ONLY from the **'AdGratified'** event.

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

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**.

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.