POKKT SDK Integration Guide (v 7.2.0)

Cordova

Overview

Thank you for choosing **Pokkt SDK** for **Cordova**. This document contains all the information required to setup the SDK with your project. We also support mediation for various third party networks. To know the supported third party networks and their integration process go to mediation section.
Before implementing plugins it is mandatory to go through project configuration and implementation steps, as these sections contain mandatory steps for basic SDK integration and are followed by every plugin.

minSdkVersion supported is **11**.

ScreenName: This one parameter is accepted by almost all API's of Pokkt SDK. This controls the placement of ads and can be created on Pokkt Dashboard.

 We will be referencing **PokktAds Demo** app provided with SDK during the course of explanation in this document. We suggest you go through the sample app for better understanding.

Project Configuration

Android

In the package downloaded above you will find:
1. Docs:
        Contains step wise step integration for SDK.
2. PokktCordovaDemo app code.
3. PokktCordovaDemo.apk:
        Application package of Pokkt Cordova Demo, so that you can directly install this apk on your device
        and have a look how our SDK works instead of compiling the source code.
4. SDK + Plugin:
    a. JAR
        ● PokktSDK_v7.2.0.jar
        ● pokktsdk360ext.jar
        ● PAPCordova.jar
    b. Dependencies
        ● Android-support-v4.jar
        ● google-play-services.jar

minSdkVersion supported is **11**.

Dependencies

● Extract the provided file "**cordova-plugin-pokkt.zip**" into a directory. Execute the following command from your terminal:
  $cordova plugin add /<*path-to-plugin-directory*>/cordova-plugin-pokkt/
● We expect **Google play services** integrated in project, although it is optional but we recommend you to integrate it, as it is required to fetch **AdvertisingID** for device,which is useful to deliver targeted advertising to Android users.

Permissions Declarations

We have already added the following mandatory permissions to the manifest via plugin.xml

1. Mandatory permissions.

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

- android.permission.INTERNET = Required for SDK communication with server.
- android.permission.ACCESS_NETWORK_STATE = Required to detect changes in network, like if WIFI is available or not.

2. Optional permissions. We have commented out these in plugin.xml. Please uncomment those for better ad delivery and ad experience.

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.VIBRATE" />
```

- android.permission.WAKE_LOCK = Required to prevent device from going into the sleep mode during video play.
- android.permission.WRITE_EXTERNAL_STORAGE = Required to store media files related to ads in external SD card, if not provided we will use app cache folder to store media files, which will result in unnecessary increase in application's size. It is recommended to ask for this permission as low end devices generally have less internal memory available.
- android.permission.WRITE_CALENDAR = Some Ads create events in calendar.
- android.permission.ACCESS_FINE_LOCATION = Some Ads show content based on user's location
- android.permission.CALL_PHONE = Some Ads are interactive and they provide you a way to call directly from the content.
- android.permission.SEND_SMS = Some Ads are interactive and they provide you a way to send message.
- android.permission.VIBRATE = Some Ads provide haptic feedback, so as to maintain their behavior we need this permission

Activity Declaration

We have already added the following activity in your AndroidManifest for Pokkt SDK integration via plugin.xml

```
<activity
      android:name="com.pokkt.sdk.PokktAdActivity"
      android:configChanges="keyboard|keyboardHidden|navigation|
      orientation|screenLayout|uiMode|screenSize|smallestScreenSize"
      android:hardwareAccelerated="true"
      android:label="Pokkt"
```

```
        android:screenOrientation="landscape"
        android:windowSoftInputMode="stateAlwaysHidden|adjustUnspecified" />
```

You can change the **android:screenOrientation="landscape"** to of your choice, the way you want to display the ads.

Service Declaration

We have already added the following service in your AndroidManifest for receiving InApp notifications. How to set up InApp notifications see "Pokkt Dashboard" document.

```
<service
        android:name="com.pokkt.sdk.notification.NotificationService"
        android:exported="false"
        android:label="PokktNotificationService" />
```

iOS

In the package downloaded above you will find:
1. Docs:
        Contains documentations for step wise step integration for SDK.
2. PokktSDK_v7.2.0:
        a. PokktSDK.framework

Dependencies

● Extract the provided file "**cordova-plugin-pokkt.zip**" into a directory. Execute the following command from your terminal:
   $cordova plugin add /<*path-to-plugin-directory*>/cordova-plugin-pokkt/

Framework

```
CoreData.framework
     Foundation.framework
     MediaPlayer.framework
     SystemConfiguration.framework
     UIKit.framework
     CoreTelephony.framework
     EventKit.framework
     AdSupport.framework
     CoreGraphics.framework
     CoreMotion.framework
     MessageUI.framework
     EventKitUI.framework
     CoreLocation.framework
     AVFoundation.framework
     libc++.tbd
```

Info.plist

Add the below exceptions to your application info.plist.

```xml
<key>NSAppTransportSecurity</key>
    <dict>
        <key>NSExceptionDomains</key>
        <dict>
            <key>pokkt.com</key>
            <dict>
                <key>NSIncludesSubdomains</key>
                <true/>
                <key>NSExceptionAllowsInsecureHTTPLoads</key>
                <true/>
                <key>NSExceptionRequiresForwardSecrecy</key>
                <false/>
                <key>NSExceptionMinimumTLSVersion</key>
                <string>TLSv1.2</string>
                <key>NSThirdPartyExceptionAllowsInsecureHTTPLoads</key>
                <false/>
                <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>
                <true/>
                <key>NSThirdPartyExceptionMinimumTLSVersion</key>
                <string>TLSv1.2</string>
                <key>NSRequiresCertificateTransparency</key>
                <false/>
            </dict>
            <key>cloudfront.net</key>
            <dict>
                <key>NSIncludesSubdomains</key>
                <true/>
                <key>NSExceptionAllowsInsecureHTTPLoads</key>
                <true/>
                <key>NSExceptionRequiresForwardSecrecy</key>
                <false/>
                <key>NSExceptionMinimumTLSVersion</key>
                <string>TLSv1.2</string>
                <key>NSThirdPartyExceptionAllowsInsecureHTTPLoads</key>
                <false/>
                <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>
                <true/>
                <key>NSThirdPartyExceptionMinimumTLSVersion</key>
                <string>TLSv1.2</string>
                <key>NSRequiresCertificateTransparency</key>
                <false/>
            </dict>
        </dict> </dict>
```

Implementation Steps

SDK Configuration

1. For all invocation of Pokkt SDK developer will make use of methods available in pokktAds.js file using PokktExtension object.

2. Set **Application Id** and **Security key** in Pokkt SDK. You can get it from Pokkt dashboard from your account. These are unique per app registered.

```
pe = window.plugin.pokktExtension;
pe.setPokktConfig("<Pokkt Application ID>", "<Pokkt Security Key>");
```

3. Set *GDPR consent* in Pokkt SDK. **This must be called before calling any ad related API. Developers/Publishers must get the consent from user.** For more information on GDPR please refer https://www.eugdpr.org/ and https://www.eugdpr.org/gdpr-faqs.html. This API can again be used by publishers to revoke the consent. If this API is not called or invalid data provided then SDK will access the users personal data for ad targeting.

```
var gdprConsent =
{GDPRApplicable:<true/false>,GDPRConsentAvailable:<true/false>};

//GDPRApplicable true if GDPR is applicable.

//GDPRConsentAvailable true if user has given consent to use personal details
for ad targeting.

pe.setDataAccessConsent(JSON.stringify(gdprConsent));
```

4. If you are using server to server integration with Pokkt, you can also set **Third Party UserId** in PokktAds.

```
pe.setThirdPartyUserId("<Third party user Id>");
```

5. When your application is under development and if you want to see Pokkt logs and other informatory messages, you can enable it by setting **setDebug** to **true**. Make sure to disable debugging before release.

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

Ad Types

- Video ad can be rewarded or non-rewarded. You can either cache the ad in advance or directly call show for it.
- We suggest you to cache the ad in advance so as to give seamless play behaviour, In other case it will stream the video which may lead to unnecessary buffering delays depending on the network connection.

1. To cache rewarded ad call:

```
pe.cacheRewardedVideoAd("<ScreenName>");
```

2. To show rewarded ad call:

```
pe.showRewardedVideoAd("<ScreenName>");
```

3. To cache non-rewarded ad call:

```
pe.cacheNonRewardedVideoAd("<ScreeName>");
```

4. To show non-rewarded ad call:

```
pe.showNonRewardedVideoAd("<ScreeName>");
```

5. To check if video ad is cached:

```
pe.isVideoAdCached("<ScreenName>", <true / false>, callback);
```

Interstitial

1. To cache rewarded ad call:

```
pe.cacheRewardedInterstitial("<ScreenName>");
```

2. To show rewarded ad call:

```
pe.showRewardedInterstitial("<ScreenName>");
```

3. To cache non-rewarded ad call:

```
pe.cacheNonRewardedInterstitial("<ScreeName>");
```

4. To show non-rewarded ad call:

```
pe.showNonRewardedInterstitial("<ScreenName>");
```

5. To check if interstitial ad is cached:

```
pe.isInterstitialAdCached("<ScreenName>", <true / false>, callback);
```

Banner

● Add **PokktBannerView** to your layout, we use it as placeholder to populate banner ad into it.

```
<com.pokkt.sdk.banners.PokktBannerView
        android:id="@+id/pokkt_banner_view_top"
        android:layout_width="320dp"
        android:layout_height="50dp"
        android:layout_centerHorizontal="true"/>
```

● Load banner

```
pe.loadBanner("<screenName>", <pokktBannerView>);
```

● You can remove Banner using:

```
pe.destroyBanner("<screenName>");
```

Ad Delegates

Ad delegates are optional, but we suggest to implement them as it will help you to keep track of the status of your ad request.

Video

```
document.addEventListener('VideoAdCachingCompleted', this.handleAdCachingCompleted,
false);
document.addEventListener('VideoAdCachingFailed', this.handleAdCachingFailed,
false);
document.addEventListener('VideoAdDisplayed', this.handleAdDisplayed, false);
document.addEventListener('VideoAdSkipped', this.handleAdSkipped, false);
document.addEventListener('VideoAdCompleted', this.handleAdCompleted, false);
document.addEventListener('VideoAdClosed', this.handleAdClosed, false);
document.addEventListener('VideoAdGratified', this.handleAdGratified, false);
document.addEventListener('VideoAdFailedToShow', this.handleAdShowFailed, false);
```

Interstitial

```
document.addEventListener('InterstitialCachingCompleted',
this.handleAdCachingCompleted, false);
document.addEventListener('InterstitialCachingFailed', this.handleAdCachingFailed,
false);
document.addEventListener('InterstitialDisplayed', this.handleAdDisplayed, false);
document.addEventListener('InterstitialSkipped', this.handleAdSkipped, false);
document.addEventListener('InterstitialCompleted', this.handleAdCompleted, false);
document.addEventListener('InterstitialClosed', this.handleAdClosed, false);
document.addEventListener('InterstitialGratified', this.handleAdGratified, false);
document.addEventListener('InterstitialFailedToShow', this.handleAdShowFailed,
false);
```

Banner

```
document.addEventListener('BannerLoaded', this.handleBannerLoaded, false);
document.addEventListener('BannerLoadFailed', this.handleBannerLoadFailed, false);
```

Pokkt ad player configuration

Pokkt Ad player works the way App is configured at Pokkt dashboard, but we provide a way to override those settings using **PokktAdPlayerViewConfig**.

Application should prefer configuration provided through code by developer or what's configured for the app in dashboard, can be controlled any time through the dashboard itself. If you want to make changes to this configuration after your app distribution, you can contact **Pokkt Team** to do the same for your app through admin console.

```
var adPlayerViewConfig = pe.createAdPlayerViewConfig();
// set properties values to adPlayerViewConfig
pe.setAdPlayerViewConfig(JSON.stringify(adPlayerViewConfig));
```

Various setters for the properties that can be managed through this are:

1. **Back button**
   Defines if user is allowed to close the Advertisement by clicking on back button or not.
   Setter Name : setBackButtonDisabled(boolean backButtonDisabled)
   Values:

   > True = Back button is disabled and user cannot close the Ad.
   > False = Back button is not disabled and user can close the Ad.

2. **Default skip time**
   Defines the time after which user can skip the Ad.
   Setter name: setDefaultSkipTime(int defaultSkipTime)
   Values:

   > Any Integer value.
   > Default value is 10 seconds.

3. **Should allow skip**
   Defines if user is allowed to skip the Ad or not.
   Setter name: setShouldAllowSkip(boolean shouldAllowSkip)
   Values:

   > True = User can skip Ad.
   > False = User can't skip Ad.

4. **Should allow mute**
   Defines if user is allowed to mute the Video Ad or not.
   Setter name: setShouldAllowMute(boolean shouldAllowMute)
   Values:

   > True = User can mute video Ad.
   > False = User can't mute video Ad.

5. **Should confirm skip**
   Defines if confirmation dialog is to be shown before skipping the Ad.
   Setter name: ShouldConfirmSkip
   Values:

   > True = Confirmation dialog will be shown before skipping the video.
   > False = Confirmation dialog will not be shown before skipping the video.

6. **Skip confirmation message**
   Defines what confirmation message to be shown in skip dialog.
   *Setter name***:** setShouldSkipConfirm(boolean shouldSkipConfirm)
   *Values***:**
   > Any String message.
   > Default value is "Skipping this video will earn you NO rewards. Are you sure?".

7. **Affirmative label for skip dialog**
   Defines what should be the label for affirmative button in skip dialog.
   *Setter name***:** setSkipConfirmYesLabel(String skipConfirmYesLabel)
   *Values***:**
   > Any String message.
   > Default value is "Yes".

8. **Negative label for skip dialog**
   Defines what should be the label for affirmative button in skip dialog.
   *Setter name***:** setSkipConfirmNoLabel(String skipConfirmNoLabel)
   *Values***:**
   > Any String message.
   > Default value is "No".

9. **Skip timer message**
   Defines message to be shown before enabling skip button. Don't forget to add placeholder "**##**" in your custom message.
   This placeholder is replaced by property "Default skip time" assigned above.
   *Setter name***:** setSkipTimerMessage(String skipTimerMessage)
   *Values***:**
   > Any String message.
   > Default value is "You can skip video in ## seconds"

10. **Incentive message**
    Defines message to be shown during video progress, that after what time user will be incentivised.
    *Setter name***:** setIncentiveMessage(String incentiveMessage)
    *Values***:**
    > Any String message
    > Default value is "more seconds only for your reward !"

11. **Should collect feedback**
    Defines message to be shown during video progress, that after what time user will be incentivised.
    *Property name***:** setShouldCollectFeedback
    *Values***:**
    > True = If you want to collect feedback from the user for the Ad.
    > False = If you don't want to collect feedback from the user for the Ad.

12. **Audio Enabled**
    Provides a medium to disable audio for video ad without user interaction.
    *Property name***:** setAudioEnabled
    *Values***:**
    > True = If you want to play audio for video ad.
    > False = If you don't want to play audio for video ad.

User Details

For better targeting of ads you can also provide user details to our SDK using.

```
var pokktUserDetails = pe.createPokktUserDetails();
pokktUserDetails.Name = "";
pokktUserDetails.Age = "";
pokktUserDetails.Sex = "";
pokktUserDetails.MobileNo = "";
pokktUserDetails.EmailAddress = "";
pokktUserDetails.Location = "";
pokktUserDetails.Birthday = "";
pokktUserDetails.MaritalStatus = "";
pokktUserDetails.FacebookId = "";
pokktUserDetails.TwitterHandle = "";
pokktUserDetails.Education = "";
pokktUserDetails.Nationality = "";
pokktUserDetails.Employment = "";
pokktUserDetails.MaturityRating = "";

pe.setUserDetails(pokktUserDetails);
```

Pokkt Server Callback Params

Developer can set some values in POKKT SDK that they need to be sent to their server via POKKT Server callbacks.

```
var extraParams = {};
//set your values here. eg: network name can be given as pokkt
extraParams.Network = "POKKT";
pe.setCallbackExtraParams(JSON.stringify(extraParams));
```

Debugging

Other than enabling debugging for Pokkt SDK, it can also be used to:
1. Export log
   Export your log to your desired location, we generally have it in root directory of SD card, if permission for external storage is provided and in cache folder otherwise.

```
pe.exportLog();
```

2. Export log to cloud
   You can also export log to cloud.

```
pe.exportLogToCloud();
```

Analytics

We support various analytics in Pokkt SDK.
Below is mentioned how to enable various analytics with Pokkt SDK.

Google Analytics

Google analytics Id can be obtained from Google dashboard.

```
var analyticsDetail = pe.createAnalyticsDetails();
analyticsDetail.selectedAnalyticsType = "GOOGLE_ANALYTICS";
analyticsDetail.googleAnalyticsID = "Google Analytics Id";

pe.setAnalyticsDetails(analyticsDetail);
```

Flurry Analytics

Flurry application key can be obtained from Flurry dashboard.

```
var analyticsDetail = pe.createAnalyticsDetails();
analyticsDetail.selectedAnalyticsType = "FLURRY";
analyticsDetail.flurryApplicationKey = "flurry Application Key";

pe.setAnalyticsDetails(analyticsDetail);
```

MixPanel Analytics

MixPanel project token can be obtained from MixPanel dashboard.

```
var analyticsDetail = pe.createAnalyticsDetails();
analyticsDetail.selectedAnalyticsType = "MIXPANEL";
analyticsDetail.mixPanelProjectToken = "mixpanel project token";
pe.setAnalyticsDetails(analyticsDetail);
```

Fabric Analytics

Analytics Id is not required in case of Fabric.

```
var analyticsDetail = pe.createAnalyticsDetails();
analyticsDetail.selectedAnalyticsType = "FABRIC";
pe.setAnalyticsDetails(analyticsDetail);
```

IAP(In App Purchase)

Call trackIAP to send any In App purchase information to Pokkt.

```
var inAppPurchaseDetails = pe.createInAppPurchaseDetail();
inAppPurchaseDetails.productId = "<productId>";
inAppPurchaseDetails.price = <price>;
inAppPurchaseDetails.currencyCode = "<currencyCode>";
inAppPurchaseDetails.title = "<title>";
inAppPurchaseDetails.description = "<description>";
inAppPurchaseDetails.purchaseData = "<purchaseData>";
inAppPurchaseDetails.purchaseSignature = "<purchaseSignature>";
inAppPurchaseDetails.purchaseStore = "<purchaseStore>";//NONE,GOOGLE,IOS,AMAZON

pe.trackIAP(inAppPurchaseDetails);
```