POKKT SDK Integration Guide (v 8.0.0)

Android

Overview

Thank you for choosing **Pokkt SDK** for **Android**. This document contains all the information required to set up 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 the 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.

You can download our SDK from pokkt.com.

Downloaded SDK package will contain:

1. Docs:
   Contains documentations for step by step integration for SDK.
2. PokktSDK_v8.0.0.jar
   Pokkt SDK in *jar* format.
3. PokktSDK_v8.0.0.aar
   Pokkt SDK in *aar* format.
4. Pokktsdk360ext.jar / Pokktsdk360ext.aar
   Add these if you want to support 360 video ads.
5. PokktAds Demo
   Source code for *PokktAds Demo*(Sample app) which showcases implementation of Pokkt SDK through code for better understanding.
6. PokktAds Demo.apk
   Application package of PoktkAds Demo, so that you can directly install this apk on your device and have a look at how our SDK works instead of compiling the source code.
7. Dependency jars:
   Please add *google play services*.

minSdkVersion supported is **11**.

ScreenId: 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 the **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

Dependencies

- Add *PokktSDK_v8.0.0.jar* or *PokktSDK_v8.0.0.aar* to your project.
- We expect **Google play services** integrated in the project, although it is optional but we recommend you to integrate it, as it is required to fetch **AdvertisingID** for devices,which is useful to deliver targeted advertising to Android users.

Manifest

Android 9.0 (API 28) blocks cleartext (non-HTTPS) traffic by default, which can prevent ads from serving correctly. To mitigate that, publishers whose apps run on Android 9.0 or above should ensure to add a network security config file or you should set the usesCleartextTraffic attribute in your application tag to true. Doing so whitelists cleartext traffic and allows non-HTTPS ads to serve.

```
<application
    ...
    android:usesCleartextTraffic="true">
    ...
</application>
```

Permissions Declarations

Add the following permissions to your project manifest

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.

Activity Declaration

Add the following activity in your AndroidManifest for Pokkt SDK integration.

```
<activity
       android:theme="@android:style/Theme.Translucent"
       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"** of your choice, the way you want to display the ads.

Implementation Steps

SDK Configuration

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

```
PokktAds.setPokktConfig("<Pokkt Application ID>", "<Pokkt Security Key>",
“<Activity Context>”);
```

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

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

3. Set ***GDPR consent*** in Pokkt SDK. **This must be called before calling any ad related API. Developers/Publishers must get the consent of the user.** For more information on GDPR please refer https://gdpr.eu/ and https://gdpr.eu/faq/. 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.

```
PokktAds.ConsentInfo consentInfo = new PokktAds.ConsentInfo();
consentInfo.setGDPRApplicable(true);
//true if GDPR is applicable.
consentInfo.setGDPRConsentAvailable(false);
//false if the user has given consent to use personal details for ad
targeting.
PokktAds.setDataAccessConsent(consentInfo);
```

Ad Types

FullScreen Ads

- FullScreen Ads are of two types : Video and Interstitial.
- FullScreen ads can be rewarded or non-rewarded.
- FullScreen properties can be configured from the Pokkt dashboard.
- You can either cache the ad in advance or directly call show for it.
- We suggest you 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 FullScreen ad call:

```
PokktAds.cacheAd("<ScreenId>",<PokktAdDelegate>);
```

2. To show FullScreen ad call:

```
PokktAds.showAd("<ScreenId>",<PokktAdDelegate>,null);
```

3. You can check if FullScreen ad is cached or not using

```
PokktAds.isAdCached("<ScreenId>");
```

- Sample Screen Id for Video : 684ab1e66abeb060faa500136c4c6a74
- Sample Screen Id for Interstitial : 5e59028c8332c9583e742c183abbaafb

Banner

- Add **PokktBannerView** to your layout, we use it as a placeholder to populate banner ads 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

```
PokktAds.showAd("<ScreenId>",<PokktAdDelegate>,<PokktBannerView>);
```

- You can remove Banner using:

```
PokktAds.destroyBanner(<pokktBannerView>);
```

- Sample Screen Id for Banner : 129cc53b4666f5ae1ebad6a9bc942764

Native Ads

A native ad may be served in feed or in between the developer content inside the app. Normally, FullScreen ads are delivered on call to action by the user. Native ads eliminate this limitation and show ads without any user request.Native ads are also non intrusive as they will be automatically paused when the ad is out of the view by scrolling. The PokktNativeAdLayout will be part of developer application parent components which may be ListView, Layout in ScrollView or WebView.

Add **PokktNativeAdLayout** to your Layout XML

```
...
<com.pokkt.sdk.pokktnativead.PokktNativeAdLayout
   android:id="@+id/pokkt_native_ad"
   android:layout_width="match_parent"
   android:layout_height="300dp"
   android:background="@android:color/white"
   android:visibility="visible" />
...
```

Request for native Ad.

```
PokktAds.requestNativeAd(“<ScreenId>”, <NativeAdsDelegate>);
```

Once **PokktNativeAd** is received in the AdReady callback of **NativeAdsDelegate** implementation, pokktNativeAdLayout view should be set to pokktNativeAd.

```
pokktNativeAd.setMediaView(findViewById(R.id.pokkt_native_ad),context);
```

Developers must implement **NativeAdsDelegate** and supply POKKT SDK in requestNativeAd().

```
PokktAds.NativeAdsDelegate delegate = new PokktAds.NativeAdsDelegate() {
   @Override
   public void adReady(screenId,pokktNativeAd) {
   }

   @Override
   public void adFailed(screenId, String errorMessage) {
   }

   @Override
   public void adClosed(String screenId, boolean isComplete) {
   }
};
```

Developers will have to do cleanup of Native Ad in onDestroy of activity life cycle.

```
pokktNativeAd.destroy();
```

Please refer to our Pokkt Ads Demo Source code for sample implementation of Native Ads in List, Scroll and WebView

- Sample Screen Id for Video : 684ab1e66abeb060faa500136c4c6a74
- Sample Screen Id for Interstitial : 5e59028c8332c9583e742c183abbaafb

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.

FullScreen Ads

```
PokktAdDelegate pokktAdDelegate = new PokktAds.PokktAdDelegate() {
   @Override
   public void adCachingResult(screenId,isSuccess,reward,errorMessage) {
   }

   @Override
   public void adDisplayedResult(screenId,isSuccess,errorMessage) {
   }

   @Override
   public void adClosed(screenId,isComplete) {
   }

   @Override
   public void adGratified(screenId,reward) {
   }

   @Override
   public void adClicked(screenId) {

   }
};
```

Native Ads

Developers should implement SDK Native Ad delegates and supply POKKT SDK in requestNativeAd().

```
PokktAds.NativeAdsDelegate delegate = new PokktAds.NativeAdsDelegate() {
   @Override
   public void adReady(screenId,pokktNativeAd) {
   }

   @Override
   public void adFailed(screenId,errorMessage) {
   }

   @Override
   public void adClosed(screenId,isComplete) {
   }
};
```

Banner

```
PokktAds.BannerAdDelegate adDelegate = new PokktAds.BannerAdDelegate() {
   @Override
   public void bannerExpanded(screenId) {
   }

   @Override
   public void bannerResized(screenId) {
   }

   @Override
   public void bannerCollapsed(screenId) {
   }

   @Override
   public void adCachingResult(screenId,isSuccess,reward,errorMessage) {
   }

   @Override
   public void adDisplayedResult(screenId,isSuccess,errorMessage) {
   //called when banner is loaded/failed to load
   }

   @Override
   public void adClosed(screenId,isComplete) {
   }

   @Override
   public void adGratified(screenId,reward) {
   }

   @Override
   public void adClicked(screenId) {
   //called when banner is clicked
   }
}
```

Pokkt ad player configuration

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

Application should prefer configuration provided through code by the developer or what's configured for the app in the 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.

```
PokktAdViewConfig adViewConfig = new PokktAdViewConfig ();
// set properties values to adPlayerViewConfig
PokktAds.setAdPlayerViewConfig(adViewConfig );
```

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 the message to be shown before enabling the 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 the 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. **Learn More message**
    Defines message to be shown for video clicks.
    Setter name: setLearnMoreMessage(String learnMoreMessage)
    Values:

    Any String message
    Default value is "Learn more"

12. **Should collect feedback**
    Defines the 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.

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

User Details

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

```
PokktUserDetails pokktUserDetails = new PokktUserDetails();
pokktUserDetails.setName(" ");
pokktUserDetails.setAge(" ");
pokktUserDetails.setSex(" ");
pokktUserDetails.setMobileNumber(" ");
pokktUserDetails.setEmailAddress(" ");
pokktUserDetails.setLocation(" ");
pokktUserDetails.setBirthday(" ");
pokktUserDetails.setMaritalStatus(" ");
pokktUserDetails.setFacebookId(" ");
pokktUserDetails.setTwitterHandle(" ");
pokktUserDetails.setEducation(" ");
pokktUserDetails.setNationality(" ");
pokktUserDetails.setEmployment(" ");
pokktUserDetails.setMaturityRating(" ");

PokktAds.setUserDetails(pokktUserDetails);
```

Pokkt Server Callback Params

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

```
Map<String, String> params = new HashMap<>();
params.put("testdata","{\"adnetwork\": \"pokkt\"}");

PokktAds.setCallbackExtraParams(params);
```

Debugging

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

```
PokktAds.Debugging.shouldDebug(“<Context Context>”,<true>);
```

2. Export log
   Export your log to your desired location, we generally have it in the root directory of the SD card, if permission for external storage is provided and in the cache folder otherwise.

```
PokktAds.Debugging.exportLog(getActivity());
```

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

```
PokktAds.Debugging.exportLog(getActivity());
```

Proguard

If you are using proguard in your app, add the following rules to your proguard file.

```
               # Pokkt SDK
-keep class com.pokkt.** { public *; }
-dontwarn com.pokkt.**
# moat
-keep class com.moat.** { *; }
-dontwarn com.moat.**
# OM
-keep class com.iab.omid.library.pokkt.**{*;}
-dontwarn com.iab.omid.**
# 360 if Pokktsdk360ext.jar or Pokktsdk360ext.aar  is added
-keep class com.pokkt.sdk360ext.** { *; }
# For communication with Pokkt WebView
-keepclassmembers class * {
   @android.webkit.JavascriptInterface <methods>;
}
```