POKKT SDK v5.0 Integration Guide for Cocos2dx-v2.2.6 & v3.X (Android)

Contents:

1. Overview:

Thank you for choosing Pokkt SDK Plugin v5.0 for Cocos2dx. Pokkt SDK supports Offerwall as well as Video/Interstitial-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 (Ad/Offerwall/Both). The current plugin supports mediation for various third party ad-networks. These are:

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

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

Kindly note that these instructions are for Cocos2dx v2.2.6 & v3.X, older versions are not supported at this moment.

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:

The Pokkt SDK for Cocos2dx comes in two zip files:

a. **pokktsdk.zip**
b. **pokktjars.zip**

Extract the pokktsdk.zip file and put the content inside your C++ project, preferably directly inside the **Classes** folder.

These files will help you to communicate with the native SDK. Please remember that the same SDK can be used for iOS, there is a separate document explaining the procedure. You can ignore the files related to iOS deployment and focus on the items mentioned in this document for deploying this on Android.

Add the POKKT libraries to your project:

Steps to follow:

> Extract the contents of pokktjars.zip.

> Copy **PokktSDK.jar and PAPCocos2dx.jar** to your project's **libs** folder.

> Merge the items of "**res**" directory with your project's "res" directory **appropriately**.

> Right click your project name and select **Properties**.

> Select Java Build Path → Add External JARs.

> Select these jars. The POKKT library is now added to your project.

Minimum android SDK version is 14.

Include 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

Include "android support v4" library

In order to support Pokkt In-App Notifications, you will need to add "android support v4" library to your project.

Android-Manifest Changes

Merge the content of “**MergeToManifest.xml**” with your manifest.

3. Implementation Steps:

Common

1. For all invocation of Pokkt SDK developer will make use of methods available in **PCPokktManager** class. This class only have static methods.

2. Setup your **PCPokktConfig,** provide **applicationId** and **securityKey,** these are must for initializing Pokkt.

3. Once you have you **PCPokktConfig** ready, invoke **initPokkt** method before you invoke any other methods from the **PCPokktManager.** 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 **POKKT_INITIALISED_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 **PCPokktConfig**.

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 **PCPokktManager::setDebug(true)** to see Pokkt debug logs and toast messages. Make sure to change this to **false** for production build.

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

9. Call **PCPokktManager::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 **PCPokktConfig**.

11. To use flurry analytics please set **selectedAnalyticsType** to “FLURRY” and **flurryApplicationKey** in **PCPokktConfig**.

12. To use mix panel analytics please set **selectedAnalyticsType** to “MIXPANEL” and **mixPanelProjectToken** in **PCPokktConfig**.

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

Session

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

Offerwall

1. for Offerwall, you can set two additional parameters in **PCPokktConfig**, which are **offerWallAssetValue** and **closeOnSuccessFlag**. Setting of **offerWallAssetValue** is only required if you only want to show campaign of certain value on the Offerwall. **CloseOnSuccessFlag** 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 **PCPokktManager**.

3. To show Offerwall you can call **PCPokktManager::getCoins()**.

4. Call **PCPokktManager::.getPendingCoins()** whenever your application resumes from background, so that you get a callback to award points to the user after he has come back to your game or app after finishing with Offerwall.

5. You can call **PCPokktManager::checkCampaignAvailable()** to check whether the campaigns are available before showing Offerwall button to user.

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

    - COIN_RESPONSE_EVENT
    - COIN_RESPONSE_WITH_TRANS_ID_EVENT
    - COIN_RESPONSE_FAILED_EVENT
    - CAMPAIGN_AVAILABILITY_EVENT
    - OFFERWALL_CLOSED_EVENT

Rewarded/Non-Rewarded Ads

1. You will have to configure an **PCAdConfig** object to request for any ad to be displayed. It also provides options for customizing your ad-screen. It is recommended to have different **PCAdConfig** objects for each screens. Followings are the values that you can set with **PCAdConfig**:
    - **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 **PCPokktManager::cacheAd(adConfig)** to cache the ad on device. Cached-ads provide better user experience than streaming-ads
3. You can call **PCPokktManager::checkAdAvailability(adConfig)** to check if the ad-campaigns are available or not. The result will be notified via **AD_AVAILABILITY_EVENT**.
4. Invoke **PCPokktManager::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:

    - AD_CACHING_COMPLETED_EVENT
    - AD_CACHING_FAILED_EVENT
    - AD_CLOSED_EVENT
    - AD_COMPLETED_EVENT
    - AD_DISPLAYED_EVENT
    - AD_SKIPPED_EVENT
    - AD_GRATIFIED_EVENT
    - AD_AVAILABILITY_EVENT

6. Reward user ONLY from the **AdGratifiedEvent.**

Banner

Pokkt SDK allows to show banner ad on your screen. You can set any custom size or any position for banner. There are few fixed position already given in BannerPosition class. But you can customise that also.

1.Load Banner :Use loadBanner to show banner ad like below:

```
PokktManager.loadBanner(ScreenName, BannerPosition.TOP_CENTER);
```

2.Remove Banner: Use removeBanner to remove banner ad from screen like below:

```
PokktManager.removeBanner(ScreenName);
```

3.Auto Refresh Bannerr: Use setBannerAutoRefresh method to disable or enable auto-refresh. Default it is true and it will refresh automatically based on given time on Pokkt dashboard for particular screen name.

```
PokktManager.setBannerAutoRefresh(false/true);
```

4.Banner Position: Use BannerPosition class properties for banner position.

5.Custom Banner: There are also option is given to custom banner size and set custom position by using below method:

```
PokktManager.loadBannerWithRect(ScreenName, width, height, x, y);
```

6.Banner Event: Register event to get banner related callback like below:

```
   //Register event
PokktManager.dispatcher.addEventListener(PokktEvent.BANNER_LOADED,
onBannerLoaded);
PokktManager.dispatcher.addEventListener(PokktEvent.BANNER_LOAD_FAILED,
onBannerLoadFailed);

// Handler method
protected function onBannerLoaded(event:PokktEvent):void {
     //event.screenName: same scree name for what banner ad request was
done
}
protected function onBannerLoadFailed(event:PokktEvent):void {
     //event.screenName: same scree name for what banner ad request was
```

```
done
    //event.message: Reason why banner load failed
```

Export Logs

1. Developer should call **PCPokktManager::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.