



POKKT SDK Integration Guide (v 7.2.0)

	Cocos2D-3x
Overview	2
Implementation Steps	3
SDK Configuration	3
Ad Types	4
Video	4
Rewarded	4
Non Rewarded	4
Interstitial	4
Rewarded	4
Non Rewarded	5
Banner	5
Ad Listeners	7
Video	7
Interstitial	7
Banner	7
Pokkt ad player configuration	8
User Details	11
Pokkt Server Callback Params	11
Debugging	12
Analytics	13
Google Analytics	13
Flurry Analytics	13
MixPanel Analytics	13
Fabric Analytics	13
IAP(In App Purchase)	13
Project Configuration	14
Android	14
Dependencies	14
Manifest	14
Permissions Declarations	14
Activity Declaration	15
Service Declaration	15
iOS	16
Dependencies	16
Framework	17
Info.plist	18



Overview

Thank you for choosing **Pokkt SDK** for **Cocos2d-3x**. 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 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.

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.

Implementation Steps

SDK Configuration

1. 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.

```
PCPokktAds::setPokktConfig("<Pokkt Application ID>", "<Pokkt Security Key>");
```

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

```
PCPokktAds::setThirdPartyUserId("<Third party user Id>");
```

3. 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.

```
PCPokktAds::Debugging.shouldDebug(<true>);
```

4. 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.

```
pokkt::PCPokktConsentInfo info;  
info.GDPRApplicable = value;  
info.GDPRConsentAvailable = value;  
PCPokktAds::setPokktConsentInfo(info);
```

Ad Types

Video

- 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.

Rewarded

1. To cache rewarded ad call:

```
PCPokktAds::cacheRewardedVideoAd("<ScreenName>");
```

2. To show rewarded ad call:

```
PCPokktAds::showRewardedVideoAd("<ScreenName>");
```

Non Rewarded

1. To cache non-rewarded ad call:

```
PCPokktAds::cacheNonRewardedVideoAd("<ScreenName>");
```

2. To show non-rewarded ad call:

```
PCPokktAds::showNonRewardedVideoAd("<ScreenName>");
```

You can check if ad is cached or not using

```
PCPokktAds::isVideoAdCached("screen_name", isRewarded);
```

Interstitial

Rewarded

1. To cache rewarded ad call:

```
PCPokktAds::cacheRewardedInterstitialAd("<ScreenName>");
```

2. To show rewarded ad call:

```
PCPokktAds::showRewardedInterstitialAd("<ScreenName>");
```

Non Rewarded

1. To cache non-rewarded ad call:

```
PCPokktAds::cacheNonRewardedInterstitialAd("<ScreenName>");
```

2. To show non-rewarded ad call:

```
PCPokktAds::showNonRewardedInterstitialAd("<ScreenName>");
```

You can check if ad is cached or not using

```
PCPokktAds::isInterstitialCached("screen_name", isRewarded);
```

Banner

There are two ways to load banners:

1. Load banner

```
PCPokktAds::loadBanner(<ScreenName>, <BannerPosition>);
```

There are predefined positions for banner positioning inside **BannerPosition** (Com. Pokkt.Plugin.Common.BannerPosition).

- TOP_LEFT.
- TOP_CENTER.
- TOP_RIGHT.
- MIDDLE_LEFT.
- MIDDLE_CENTER.
- MIDDLE_RIGHT.
- BOTTOM_LEFT.
- BOTTOM_CENTER.
- BOTTOM_RIGHT.

2. Load banner with rect

```
PCPokktAds::initWithBannerAdSize(<ScreenName>,<Height>, Width>, <x>,<y>);
```

Height : Custom height for banner.

Width: Custom width for banner.

x: Point x on screen to show banner.

y: Point y on screen to show banner.

You can remove Banner using:

```
PCPokktAds::destroyBanner( )
```


Ad Listeners

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

```
PCPokktAds::setAdEventListener(AD_CACHING_COMPLETED_EVENT, PC_ADS_EVENT_SELECTOR(
VideoAdView::handleAdCachingCompleted), this);
PCPokktAds::setAdEventListener(AD_CACHING_FAILED_EVENT,
PC_ADS_EVENT_SELECTOR(VideoAdView::handleAdCachingFailed), this);
PCPokktAds::setAdEventListener(AD_DISPLAYED_EVENT,
PC_ADS_EVENT_SELECTOR(VideoAdView::handleAdDisplayed), this);
PCPokktAds::setAdEventListener(AD_SKIPPED_EVENT,
PC_ADS_EVENT_SELECTOR(VideoAdView::handleAdSkipped), this);
PCPokktAds::setAdEventListener(AD_COMPLETED_EVENT,
PC_ADS_EVENT_SELECTOR(VideoAdView::handleAdCompleted), this);
PCPokktAds::setAdEventListener(AD_CLOSED_EVENT,
PC_ADS_EVENT_SELECTOR(VideoAdView::handleAdClosed), this);
PCPokktAds::setAdEventListener(AD_GRATIFIED_EVENT,
PC_ADS_EVENT_SELECTOR(VideoAdView::handleAdGratified), this);
```

Interstitial

```
PCPokktAds::setAdEventListener(AD_CACHING_COMPLETED_EVENT, PC_ADS_EVENT_SELECTOR(
InterstitialAdView::handleAdCachingCompleted), this);
PCPokktAds::setAdEventListener(AD_CACHING_FAILED_EVENT, PC_ADS_EVENT_SELECTOR(Int
erstitialAdView::handleAdCachingFailed), this);
PCPokktAds::setAdEventListener(AD_DISPLAYED_EVENT,
PC_ADS_EVENT_SELECTOR(InterstitialAdView::handleAdDisplayed), this);
PCPokktAds::setAdEventListener(AD_SKIPPED_EVENT,
PC_ADS_EVENT_SELECTOR(InterstitialAdView::handleAdSkipped), this);
PCPokktAds::setAdEventListener(AD_COMPLETED_EVENT,
PC_ADS_EVENT_SELECTOR(InterstitialAdView::handleAdCompleted), this);
PCPokktAds::setAdEventListener(AD_CLOSED_EVENT,
PC_ADS_EVENT_SELECTOR(InterstitialAdView::handleAdClosed), this);
PCPokktAds::setAdEventListener(AD_GRATIFIED_EVENT,
PC_ADS_EVENT_SELECTOR(InterstitialAdView::handleAdGratified), this);
```

Banner

```
PCPokktAds::setAdEventListener(BANNER_LOADED_OP,
PC_ADS_EVENT_SELECTOR(BannerAdView::handleBannerLoadSuccessfully), this);

PCPokktAds::setAdEventListener(BANNER_LOAD_FAILED_OP,
PC_ADS_EVENT_SELECTOR(BannerAdView::handleBannerLoadFailed), this);
```

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.

```
PokktAdPlayerViewConfig adPlayerViewConfig = new PokktAdPlayerViewConfig ();  
// set properties values to adPlayerViewConfig  
PCPokktAds::setAdPlayerViewConfig(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. **isAudioEnabled**

Defines if user is allowed to mute the Video Ad or not.

Setter name: setIsAudioEnabled(boolean isAudioEnabled)

Values:

True = Video Ad will be mute.

False = Video Ad will not be mute.

User Details

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

```
PCPokktUserDetails* pokktUserDetails = new PokktUserDetails();
pokktUserDetails -> name = " ";
pokktUserDetails -> age = " ";
pokktUserDetails -> sex = " ";
pokktUserDetails -> mobileNumber = " ";
pokktUserDetails -> emailAddress = " ";
pokktUserDetails -> location = " ";
pokktUserDetails -> birthday = " ";
pokktUserDetails -> maritalStatus = " ";
pokktUserDetails -> facebookId = " ";
pokktUserDetails -> twitterHandle = " ";
pokktUserDetails -> education = " ";
pokktUserDetails -> nationality = " ";
pokktUserDetails -> employment = " ";
pokktUserDetails -> maturityRating = " ";

PokktAds::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.

```
String params = "{\"adnetwork\":\"pokkt\"}";

PCPokktAds::setCallBackExtraParam("<String>");

String format should be same as like:
"{\"Key 1\":\"value 1\", \"Key 2\":\"value 2\", \"Key 3\":\"value 3\", \"Key 4\":\"value 4\"}"
```

Debugging

Other than enabling debugging for Pokkt SDK, it can also be used to:

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.

```
PCPokktAds : exportLog();
```

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.

```
PCPokktAnalyticsDetails* analyticsDetail = new AnalyticsDetails();
analyticsDetail->eventType = "GOOGLE_ANALYTICS";
analyticsDetail->googleAnalyticsID( "<Google Analytics Id>");
PCPokktAds::setAnalyticDetail(analyticsDetail);
```

Flurry Analytics

Flurry application key can be obtained from Flurry dashboard.

```
PCPokktAnalyticsDetails* analyticsDetail = new AnalyticsDetails();
analyticsDetail->eventType = "FLURRY";
analyticsDetail->flurryApplicationKey("<Flurry Application Key>");
PCPokktAds::setAnalyticDetail(analyticsDetail);
```

MixPanel Analytics

MixPanel project token can be obtained from MixPanel dashboard.

```
PCPokktAnalyticsDetails* analyticsDetail = new AnalyticsDetails();
analyticsDetail->eventType = "MIXPANEL";
analyticsDetail->mixPanelProjectToken( "<MixPanel Project Token>");
PCPokktAds::setAnalyticDetail(analyticsDetail);
```

Fabric Analytics

Analytics Id is not required in case of Fabric.

```
PCPokktAnalyticsDetails* analyticsDetail = new AnalyticsDetails();
analyticsDetail->eventType = "FABRIC";
PCPokktAds::setAnalyticDetail(analyticsDetail);
```

IAP(In App Purchase)

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

```
PCIAPDetails* pciAPDetails = new PCIAPDetails();
pciAPDetails -> productId("<productId>");
pciAPDetails -> purchaseData("<purchaseData>");
pciAPDetails -> purchaseSignature("<purchaseSignature>");
pciAPDetails -> purchaseStore(IAPStoreType.GOOGLE);
pciAPDetails -> price(<100.00>);
PCPokktAds::setTrackIAP(pciAPDetails );
```

Project Configuration

In the package downloaded above you will find:

1. Docs:
 - Contains documentations for step wise step integration for SDK..
2. android_plugin
 - AndroidManifest.xml: mentions
 - libs
 - android-support-v4.jar
 - google-play-services.jar
 - PAPCocos2dx.jar
 - PokktSDK_v7.0.jar
 - pokktsdk360ext.jar
3. ios_plugin
 - PokktSDK.framework.
4. pokkt_demo_app
 - Source code for *PokktAds Demo*(Sample app) which showcase implementation of Pokkt SDK through code for better understanding.
5. pokktsdk :
 - Extension for pokkt SDK.

Android

minSdkVersion supported is **11**.

Dependencies

- Extract the *PokktSDK_Native_Extension.zip* file and put the content inside your C++ project, preferably directly inside the *Classes* folder.
- Add *PokktSDK_v7.0.jar*, *android-support-v4.jar* and *PAPCocos2dx.jar* to your project.
- We expect **Google play services** integrated in project, although it;s optional but we recommend you to integrate it, as it's required to fetch **AdvertisingID** for device,which is useful to deliver targeted advertising to Android users.

Manifest

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.

- Optional permissions.

```
<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 internally 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

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

```
<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

Add the following service in your AndroidManifest for receiving InApp notifications. [How to set up InApp notifications.](#)

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

ios

In the package downloaded above you will find:

Dependencies

- Extract the *pokktsdk.zip* file and put the content inside your C++ project, preferably directly inside the *Classes* folder.
- Extract the *libpokkt.zip* and add its content to your project. And add the *PokktSDK.framework*. Just to make sure, go to your project's settings "*Build Phases -> Link Binary with Libraries*". If *PokktSDK.framework* is not present then make an entry there and you are good to go.
- Ensure that "*-ObjC*" and "*-lxml2*" flag is set inside project setting's "*Build Settings -> Other Linker Flags*".

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];
```

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
{
    [PokktAds callBackgroundTaskCompletedHandler:
        ^(UIBackgroundFetchResult result)
        {
            completionHandler(result);
        }
    ];
}
```

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];
```

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
    {
        [PokktAds inAppNotificationEvent:notification];
    }
```

Framework

Add Following frameworks to your project:

```
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
-lsqlite3.tbd
```

Info.plist

Add the below exceptions to your application info.plist.

```
<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>

```