
POKKT SDK v4.0 Integration Guide **for Cocos2dx-v2.2.6 & v3.X (Android)**

Contents:

1. Overview
2. Configuration Steps
 - a. AndroidManifest.xml
3. Implementation Steps
4. Functionalities:
 1. Offerwall
 2. Video
5. Debugging and Logging

1. Overview:

Thank you for choosing Pokkt SDK Plugin v4.0 for Cocos2dx. Pokkt SDK supports Offerwall as well as Video-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 (Video/Offerwall/Both). The current plugin supports mediation for various third party ad-networks. These are:

- AdColony
- AppLovon
- Chartboost
- Fyber
- InMobi
- SuperSonic
- UnityAds
- TapJoy
- Vungle
- FusePowered (not available for iOS)
- MoPub (not available for iOS)

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.

Note: There is no need to set **PokktState.setActivity()** from this version onward.

2. Configuration Steps:

The Pokkt SDK v4.0 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. The folder structure should look like following:

```
Classes
| ---<your other classes/folders>
| ---pokktsdk
|   | --- PokktCocosConfig.h
|   | --- PokktCocosConfig.cpp
|   | --- PokktCocosManager.h
|   | --- PokktCocosManager.cpp
|   | --- PokktNativeExtension.h
|   | --- PokktNativeExtension.cpp
|
```

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 **Cocos2dxJavaWrapper.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.

AndroidManifest.xml

Step 1: Configure “AndroidManifest.xml” by adding permissions

Required permissions:

INTERNET
ACCESS_NETWORK_STATE

Recommended Permissions:

READ_PHONE_STATE
WRITE_EXTERNAL_STORAGE
WAKE_LOCK

Optional Permissions:

WRITE_CALENDAR

Add following code snippet within <application...> tag in manifest file.

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

<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WAKE_LOCK" />

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

Step 2: Add Activity definition

Add following Activities in manifest. Add both activities to support both features (offerwall and video).

Offerwall:

```
<activity
    android:name="com.app.pokktsdk.ShowOfferwallActivity"
    android:configChanges="keyboard|keyboardHidden|navigation|
    orientation|screenLayout|uiMode|screenSize"
    android:windowSoftInputMode="adjustPan"/>
```

Video:

```
<activity
    android:name="com.app.pokktsdk.PlayVideoCampaignActivity"
    android:configChanges="keyboard|keyboardHidden|navigation|
    orientation|screenLayout|uiMode|screenSize"
    android:screenOrientation="landscape"
    android:windowSoftInputMode="stateAlwaysHidden|adjustUnspecified"/>
```

Step 3: Add BroadcastReceiver (required only for offerwall)

Add following code snippet within </application> tag in manifest.

Offerwall:

```
<receiver android:name="com.app.pokktsdk.ApplInstallBroadcastReceiver" >
    <intent-filter android:priority="1000" >
        <action android:name="android.intent.action.PACKAGE_INSTALL" />
        <action android:name="android.intent.action.PACKAGE_ADDED" />
        <data android:scheme="package" />
    </intent-filter>
</receiver>
```

Step 4: Add meta-data information

Add the following within </application> tag in manifest.

```
<meta-data
    android:name="offerwallDelegate"
    android:value="com.pokkt.cocos2dx.OfferwallEventsHandler" />

<meta-data
    android:name="videoDelegate"
    android:value="com.pokkt.cocos2dx.VideoEventsHandler" />
```

Step 6: 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>

3. Implementation Steps:

Common

1. For all invocation of Pokkt SDK developer will make use of methods available in **PokktCocosManager** class. This class only have static methods.
2. In **PokktCocosConfig** you can set **setApplicationId**, **setSecurityKey**, **setIntegrationType** and **setAutoCacheVideo**. which are must for all type of integrations
3. Make sure to invoke **initPokkt** method before you invoke any other methods from the **PokktCocosManager**. This does not apply to session related methods namely **startSession** and **endSession**
4. If you are doing server to server integration with Pokkt you can also set **setThirdPartyUserId** in **PokktCocosConfig**.
5. Apart from above mentioned parameters you can assign additional ones based on your integration type. Refer to OfferWall and Video sections below.
6. While in development please call **PokktCocosManager::setDebug(true)** to see Pokkt debug logs and toast messages. please make sure to change this to **false** for production build.
7. Please call **PokktCocosManager::sendAppInfo()** to send your application installation information to Pokkt.
8. Android **MinSDKVersion** should be ≥ 9 .
9. To use google analytics, please set **setAnalyticsType** and **setAnalyticsID** in **PokktCocosConfig**.
10. To use flurry analytics please set **setAnalyticsType** and **setFlurryApplicationKey** in **PokktCocosConfig**.
11. To use mix panel analytics please set **setAnalyticsType** and **setMixPanelProjectToken** in **PokktCocosConfig**.

Session

1. We have option to start session for tracking for which we have **startSession** and **endSession** methods in **PokktCocosManager**.
2. You should call **startSession** at the start of his application and once only. You will need to call this method after setting required details like **setApplicationId**, **setSecurityKey** and **setIntegrationType**.

-
3. You should call **endSession** at the end of his application and once only.

Offerwall

1. In **PokktCocosConfig** for Offerwall you can set two additional parameters which are **setOfferWallAssetValue** and **setCloseOnSuccessFlag**. Setting of **setOfferWallAssetValue** is only required if you only want to show campaign of certain value on the Offerwall. **setCloseOnSuccessFlag** 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 **PokktCocosManager**.
3. You will need to implement **IOfferwallDelegate** interface as mentioned in Step-4 in configuration steps.
4. To show Offerwall you can call **PokktCocosManager::getCoins()**.
5. In the screen or activity where you have button to show Offerwall, in that activity **onResume** you should call **PokktCocosManager::getPendingCoins()** so that you get a callback to award points to the user after he has come back to your game after finishing with Offerwall. You will get a callback for this call in your **IOfferwallDelegate** implementation class in method **EarnedCoins** or **CoinResponseFailed**.
6. You can call **PokktCocosManager::checkCampaignAvailable()** to check whether the campaigns are available before showing Offerwall button to user. You will get a callback for this call in your **IOfferwallDelegate** implementation class in method **onOfferwallCampaignCheck**.

Video

1. In **PokktCocosConfig** for Video you can set five additional parameters which are **setAutoCacheVideo**, **setSkipEnabled**, **setDefaultSkipTime**, **setScreenName** and **setIncentivised**.
2. **setAutoCacheVideo** is required if you want to automatically cache video on user device. It has default value as true. if you set it as false then video will not be automatically cached and you will have to call **PokktCocosManager::CacheVideoCampaign()** to start caching videos on device.
3. If you want to enable/disable the skip button on video screen please set **setSkipEnabled** to **true/false**. The default value for **setSkipEnabled** is false.
4. If you have enabled skipped button by setting **setSkipEnabled** to **true**, then you can control after how many seconds the skip button will be visible in video by setting **setDefaultSkipTime** to appropriate value. Since most videos will be 30 sec or less

please set **setDefaultSkipTime** as 10 or less. You can also give your own skip message by setting **setCustomSkipMessage** on **PokktCocosConfig**.

5. **setScreenName** has default value as 'default' and can be used by you to give different screen-name for different places in your app where you are showing video-ads. You will control ad targeting based on these screen names which should match exactly with screen names defined in dashboard. **setScreenName** can not contain white spaces and only special characters allowed are hyphen and underscore.
6. You can choose to show video with or without incentive to user by setting **setIncentivised** to **true/false**. Video gratification will only happen for incentivised playback.
7. You can disable the back button while video is playing by setting **setBackButtonDisabled** on **PokktCocosConfig**.
8. You will need to create **IVideoDelegate** implementation class as mentioned in Step-4 in configuration steps.
9. You can call **PokktCocosManager::isVideoAvailable()** to check if the campaigns are available before you try to play video.
10. You can call **PokktCocosManager::getVideo()** to play video.
11. You will get different callbacks as given in **IVideoDelegate** implementation class for video playback.
12. Reward user only from the **onVideoGratified** method in **IVideoDelegate** implementation class.

Export Logs

1. Developer should call **PokktCocosManager::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

PokktCocosConfig also has provision for developers to provide extra user data available with them to Pokkt. We currently support following data points: **setName**, **setAge**, **setSex**, **setMobileNo**, **setEmailAddress**, **setLocation**, **setBirthday**, **setMaritalStatus**, **setFacebookId**, **setTwitterHandle**, **setEducation**, **setNationality**, **setEmployment** and **setMaturityRating**.

In-App Notifications

Developer can add In-App notifications in their dashboard.

Add Notification

Basic

Name

App

Platform

☐ iOS ☐ Android ☒ All

Filters

Countries

App Version

Last Seen

Min

Max

Schedule

Repeat

Repeat

Dates

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31					

Time

O'clock

Message

Message

Title

Add Image



Cancel

Save

Repeat schedule can be daily, weekly monthly.

Daily Repeat can have options like frequency of repeat and time in hours of notification.

The screenshot shows a configuration window with two main sections: 'Schedule' and 'Message'.
In the 'Schedule' section, the 'Repeat' dropdown is set to 'Daily'. Below it, the 'Every' field is set to '1' with a unit of 'Day(s)'. The 'Time' field is set to '12' with a unit of 'O'clock'.
In the 'Message' section, there is a 'Title' input field and a larger text area for the message body. Below the text area is an 'Add Image' button. To the right of the text area is a preview of a smartphone screen displaying the notification.
At the bottom right of the window are 'Cancel' and 'Save' buttons.

Weekly repeat can have options like frequency of repeat in weeks, days of repeat and time in hours of notification.

The screenshot shows a configuration window similar to the one above, but for a weekly notification.
In the 'Schedule' section, the 'Repeat' dropdown is set to 'Weekly'. The 'Every' field is set to '1' with a unit of 'Week(s)'. Below this is a grid of days: 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', and 'Sun'. The 'Time' field is set to '12' with a unit of 'O'clock'.
The 'Message' section is identical to the previous screenshot, with a 'Title' field, a message text area, an 'Add Image' button, and a smartphone preview.
At the bottom right are 'Cancel' and 'Save' buttons.

Monthly repeat can have options like frequency of repeat in months dates of repeat and time in hours for notification .

Repeat: Monthly

Every 1 Month(s)

Dates:

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31					

Time: 12 O'clock

Message

Message: Title

Add Image

Cancel Save

For don't repeat case, there are options like dates and time in hour for notification.

The notifications are listed and can be edited. Notifications can also be deactivated/activated.

List Notifications

Id	Name	App Id	Header	Status	Action	
					Edit	Deactivate
1	push	1000125	Hi There	ACTIVE	Edit	Deactivate
2	in app	1000125	Hi There	ACTIVE	Edit	Deactivate

4. Functionalities:

4.1 Offerwall

There are five events to listen too. These are:

- CoinResponseEvent
- CoinResponseWithTransIdEvent
- CoinResponseFailedEvent
- CampaignAvailabilityEvent
- OfferwallClosedEvent

Add listeners to these events in the init() method of your Node or similar class. These are all of EventCustom type. Below are the references on how to use them:

Add listeners:

```
// listen for pokkt events
PokktCocosManager::getInstance()->setListener(
    PokktCocosManager::CoinResponseEvent,
    pokkt_event_selector(OfferwallView::handleCoinResponse),
    this);

PokktCocosManager::getInstance()->setListener(
    PokktCocosManager::CoinResponseWithTransIdEvent,
    pokkt_event_selector(OfferwallView::handleCoinResponseWithTrId),
    this);

PokktCocosManager::getInstance()->setListener(
    PokktCocosManager::CoinResponseFailedEvent,
    pokkt_event_selector(OfferwallView::handleCoinResponseFailed),
    this);

PokktCocosManager::getInstance()->setListener(
    PokktCocosManager::CampaignAvailabilityEvent,
    pokkt_event_selector(OfferwallView::handleCampaignAvailibity),
    this);

PokktCocosManager::getInstance()->setListener(
    PokktCocosManager::OfferwallClosedEvent,
    pokkt_event_selector(OfferwallView::handleOfferwallclosed),
    this);
```

Reference on how to consume them:

```
void OfferwallView::handleCoinResponse(std::string coins)
{
    if (coins == "-1")
    {
        // no coins earned
    }
    else
    {
        // coins earned
    }
}

void OfferwallView::handleCoinResponseWithTrId(std::string coinsWithTrId)
{
    // extract comma separated values
    std::stringstream stream(coinsWithTrId);
    std::string value;
    std::vector<std::string> values;
    while (getline(stream, value, ','))
    {
        values.push_back(value);
        if (stream.peek() == ',')
            stream.ignore();
    }

    if (values.size() < 2)
        return; // the values received are not proper

    std::string points = values[0];
    std::string transId = values[1];

    if (points == "-1")
    {
        // no points earned
    }
    else
    {
        // points earned equal to coins with transaction id
    }
}

void OfferwallView::handleCoinResponseFailed(std::string message)
{
    // pending coins request fails
}

void OfferwallView::handleCampaignAvailibity(std::string message)
{
    // campaign availability status
    bool available = message == "true";
}

void OfferwallView::handleOfferwallclosed(std::string message)
{
    // offerwall is closed
}
```

There are two ways to invoke offerwall.

Open Asset Value: In this case, POKKT platform provides all offers with any asset value.

Sample code snippet:

```
PokktCocosManager:: getFreeCoins();
```

Fixed Asset Value: In this case, POKKT platform provides all offers with fixed asset value.

Sample code snippet:

```
PokktCocosManager:: getFreeCoins(fixedAssetValue);
```

Pending Coins: In case after completing activity, if status of transaction is pending, then call `getPendingCoins()` method. You should always call this method in your calling activity's `onResume` method so that you can check for the pending coins for user.

Sample code snippet:

```
PokktCocosManager:: getPendingCoins();
```

Check for campaign availability: If you are using optional meta-tag as mentioned in Step 5, you can call the following to check for campaign availability:

```
PokktCocosManager: checkOfferwallCampaign();
```

The result will be noticed with the event:

```
PokktCocosManager::CampaignAvailabilityEvent
```

Moreover, you can listen to the following event to know whether offerwall has been closed or not:

```
PokktCocosManager::OfferwallClosedEvent
```

4.2 Video:

There are 7 events to manage the video caching and its playback, these are:

- VideoClosedEvent
- VideoDisplayedEvent
- VideoSkippedEvent
- VideoCompletedEvent
- VideoGratifiedEvent
- DownloadCompletedEvent
- DownloadFailedEvent

Add listeners to these events in the init() method of your Node or similar class. These are all of EventCustom type. Below are the references on how to use them:

Add listeners:

```
// listen for pokkt events
PokktCocosManager::setListener(
    PokktCocosManager::VideoClosedEvent,
    pokkt_event_selector(VideoView::handleVideoClosed),
    this);

PokktCocosManager::setListener(
    PokktCocosManager::VideoDisplayedEvent,
    pokkt_event_selector(VideoView::handleVideoDisplayed),
    this);

PokktCocosManager::setListener(
    PokktCocosManager::VideoSkippedEvent,
    pokkt_event_selector(VideoView::handleVideoSkipped),
    this);

PokktCocosManager::setListener(
    PokktCocosManager::VideoCompletedEvent,
    pokkt_event_selector(VideoView::handleVideoCompleted),
    this);

PokktCocosManager::setListener(
    PokktCocosManager::VideoGratifiedEvent,
    pokkt_event_selector(VideoView::handleVideoGratified),
    this);

PokktCocosManager::setListener(
    PokktCocosManager::DownloadCompletedEvent,
    pokkt_event_selector(VideoView::handleDownloadCompleted),
    this);

PokktCocosManager::setListener(
    PokktCocosManager::DownloadFailedEvent,
    pokkt_event_selector(VideoView::handleDownloadFailed),
    this);
```

Reference on how to consume them:

```
void VideoView::handleVideoClosed(std::string message)
{
    // video is closed
}

void VideoView::handleVideoDisplayed(std::string message)
{
    // video is displayed
}

void VideoView::handleVideoSkipped(std::string message)
{
    // video was skipped
}

void VideoView::handleVideoCompleted(std::string message)
{
    // video viewing is completed
}

void VideoView::handleVideoGratified(std::string coins)
{
    if (coins == "-1")
    {
        // no points earned
    }
    else
    {
        // points earned equals to coins
    }
}

void VideoView::handleDownloadCompleted(std::string coinsToEarn)
{
    // video is downloaded
}

void VideoView::handleDownloadFailed(std::string message)
{
    // pending coins request fails
}
```

A video file is cached on user's device. You can set the auto-caching option in the beginning, as mentioned earlier in this document. In case of manual caching, call the following to start video caching:

```
PokktCocosManager::getInstance()->startVideoCaching();
```

Before playing video or showing button to play video, Application should check whether video is cached or not by calling the following:

```
PokktCocosManager::getInstance()->isVideoAvailable()
```

You should listen to `PokktCocosManager::DownloadCompletedEvent` to check whether download is completed or not, you can show the play buttons once you receive this event.

Furthermore, Application can decide to play video as incent (user will be gratified after watching complete video) or non-incent (user will not be gratified after watching complete video). You must provide the screen-name parameter for it. Followings are the method calls to me made:

```
PokktCocosManager::getVideo("screen_name");  
PokktCocosManager::getVideoNonIncent("screen_name");
```

Next, you can listen to `PokktCocosManager::VideoGratifiedEvent` to get the coins earned, if at all, by watching the last video.

5. Debugging and Logging

You can enable the SDK logs by setting the debugging option to true anytime. Ref.:

```
PokktCocosManager::setDebug(<true/false>);
```

This can help you debug basic issues related to Pokkt SDK.

You can use the following command to log some debug messages:

```
PokktCocosManager::showLog("pokkt init...");
```

You can use the following command to display a message as Toast on android device:

```
PokktCocosManager::showToast("pokkt init...");
```

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.