
POKKT SDK v2.0.6 Integration Guide for Cocos2dx-v3.x (Android)

Contents:

1. Introduction
2. Installation
3. SDK Setup on Android
4. SDK Setup on Cocos2dx-v3.x
5. Functionalities:
 1. Offerwall
 2. Video
6. Debugging and Logging

1. Introduction:

Thank you for choosing Pokkt SDK for Cocos2dx-v3.x. This document contains all the information that is needed by you to setup the SDK with your project. Kindly note that these instructions are for Cocos2dx-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.

2. Installation:

The Pokkt SDK v2.0.6 for Cocos2dx-v3.x 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
|   | --- PokktManager.h
|   | --- PokktManager.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.

3. SDK Setup on Android:

Step 1: Add the POKKT libraries to your project

Steps to follow:

- > Copy **PokktSDK.jar** and **Cocos2dxJavaWrapper.jar** to your project's **libs** folder.
- > 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.

Step 2: Configure “AndroidManifest.xml” by adding permissions

Required permissions:

Common:

```
READ_PHONE_STATE
INTERNET
ACCESS_NETWORK_STATE
```

Video specific:

```
WRITE_EXTERNAL_STORAGE
WAKE_LOCK
```

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

Common:

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

Video specific:

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

Step 3: 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.PokktManager"
    android:configChanges="keyboard|keyboardHidden|navigation|
    orientation|screenLayout|uiMode|screenSize"
    android:windowSoftInputMode="adjustPan">
</activity>
```

Video:

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

Step 4: 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 5: Add meta-data information

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

```
<meta-data
    android:name="security_key"
    android:value="{security key shared by POKKT}">
</meta-data>
<meta-data
    android:name="application_id"
    android:value="{app id shared by POKKT}">
</meta-data>
<meta-data
    android:name="U_ID"
    android:value="{user id shared by POKKT}">
</meta-data>
<meta-data
    android:name="integration_type"
    android:value=" integration_type_value">
</meta-data>
```

Optionally you can also add the following meta data tag if you want to listen for additional events for the offerwall

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

Note:

Value for integration is as follows

(SDK supports both features By default, i.e., default value is "0"):

"1": Only offer wall

"2": Only video

For the time being, when your app is in approval process and you have not received security key, app-id, user id, etc. from POKKT, you can use POKKT sample app provided to you for demo-purpose.

You can skip adding meta-tag and pass this information to PokktManager as one of the params-value also as explained in Invoke SDK into Application.

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 recognising the devices. It no longer allows the developer to read the Android_ID. Instead a new Advertisers ID is needed to be used.

Please find more details below

<https://developer.android.com/google/play-services/id.html>

Step 7: Assign the main activity to the Pokkt Handler

See the following for the reference:

```
public class AppActivity extends Cocos2dxActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);

        // set the activity
        PokktManagerHandler.setActivity(this);
    }
}
```

Step 8: Make Entries in jni/Android.mk

Make entries for PokktManager.cpp and PokktNativeExtension.cpp inside jni/Android.mk. See the next snippet for reference:

```
LOCAL_SRC_FILES := hellocpp/main.cpp \  
                  ../../Classes/AppDelegate.cpp \  
                  :  
                  :  
                  ../../Classes/pokktsdk/PokktNativeExtension.cpp \  
                  ../../Classes/pokktsdk/PokktCocosManager.cpp  
                  :  
                  :
```

4. SDK Setup on Cocos2d-x:

Initialize PokktManager

You can start with setting up the following values.

- Security Key
- Application Id
- User Id
- Integration Type
- Auto Cache Video

These values can also be set in AndroidManifest.xml. In order to disable auto-caching of videos, you are required to enter these values in your code and NOT in AndroidManifest.xml.

```
// SET THESE UP AS PER VALUES PROVIDED TO YOU
PokktManager::SecurityKey = "<your security key>";
PokktManager::ApplicationId = "<your application id>";
PokktManager::UserId = "<your user id>";
PokktManager::IntegrationType = "0";
```

After setting these values, make sure to set the auto caching option in the SDK. Ref.:

```
// make sure to set auto caching disabled once the params are set
PokktManager::getInstance()->setAutoCaching(false);
```

5. Functionalities:

5.1 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
PokktManager::getInstance()->setListener(
    PokktManager::CoinResponseEvent,
    pokkt_event_selector(OfferwallView::handleCoinResponse),
    this);

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

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

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

PokktManager::getInstance()->setListener(
    PokktManager::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:

```
PokktManager::getInstance()->getFreeCoins();
```

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

Sample code snippet:

```
PokktManager::getInstance()->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:

```
PokktManager::getInstance()->getPendingCoins();
```

[Optional] 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:

```
PokktManager::getInstance()->checkOfferwallCampaign();
```

The result will be noticed with the event:

```
PokktManager::CampaignAvailabilityEvent
```

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

```
OfferwallView::handleOfferwallclosed
```

5.2 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 reference on how to use them:

Add listeners:

```
// listen for pokkt events
PokktManager::getInstance()->setListener(
    PokktManager::VideoClosedEvent,
    pokkt_event_selector(VideoView::handleVideoClosed),
    this);

PokktManager::getInstance()->setListener(
    PokktManager::VideoDisplayedEvent,
    pokkt_event_selector(VideoView::handleVideoDisplayed),
    this);

PokktManager::getInstance()->setListener(
    PokktManager::VideoSkippedEvent,
    pokkt_event_selector(VideoView::handleVideoSkipped),
    this);

PokktManager::getInstance()->setListener(
    PokktManager::VideoCompletedEvent,
    pokkt_event_selector(VideoView::handleVideoCompleted),
    this);

PokktManager::getInstance()->setListener(
    PokktManager::VideoGratifiedEvent,
    pokkt_event_selector(VideoView::handleVideoGratified),
    this);

PokktManager::getInstance()->setListener(
    PokktManager::DownloadCompletedEvent,
    pokkt_event_selector(VideoView::handleDownloadCompleted),
    this);

PokktManager::getInstance()->setListener(
    PokktManager::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:

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

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

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

You should listen to PokktManager::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:

```
PokktManager::getInstance()->getVideo("screen_name");  
PokktManager::getInstance()->getVideoNonIncent("screen_name");
```

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

6. Debugging and Logging

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

```
PokktManager::getInstance()->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:

```
PokktManager::getInstance()->showLog("pokkt init...");
```

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

```
PokktManager::getInstance()->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.