POKKT SDK v2.0 Integration Guide for Cocos2dx 3.x (iOS)

Contents:

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 Version 3.x and above, older versions of Cocos2dx 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 plugin for Cocos2dx-v3.x comes in two zip files:

A. pokktsdk.zip.
B. libpokkt.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
    |       |—— ios
    |       |       |—— IOSExtension.h
    |       |       |—— IOSExtension.mm
    |       |—— PokktCocosConfig.h
    |       |—— PokktCocosConfig.cpp
    |       |—— PokktCocosManager.h
    |       |—— PokktCocosManager.cpp
    |       |—— PokktNativeExtension.h
    |       |—— PokktNativeExtension.cpp
```

Extract and *libpokkt.zip* and add its content to your project. It should automatically add the *libPokktSDK.a* and other header files to the project. Just to make sure, go to your project's *Settings -> Build Phases -> Link Binary with Libraries*. If *libPokktSDK.a* is not present then make an entry there and you are good to go. Make sure that you have all the contents of "*include*" folder in your project too.

Make sure to add the "*PokktSDKResource.bundle*" to the your project.

Your Project needs to have following frameworks to use PokktSDK

- CoreData.framework
- Foundation.framework
- MediaPlayer.framework
- SystemConfiguration.framework
- UIKit.framework

3. Code Integration:

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. Before calling any other methods from the *PokktCocosManager* please make sure that you have called the *initPokkt* already.(This does not apply to session related methods namely *startSession* and *endSession*)
3. For almost all methods call *PokktCocosConfig* instance is required. *PokktCocosConfig* is plain object which will hold all the values required by the SDK which you need to assign.
4. In *PokktCocosConfig* you can assign *applicationId* and *securityKey* which are must for all type of integrations.Value for *integrationType* for iOS will always be PokktIntegrationType.INTEGRATION_TYPE_VIDEO
5. If you are doing server to server integration with pokkt you can also mention *thirdPartyUserId* in *PokktCocosConfig*.
6. While in development please call *PokktCocosManager.SetDebug(true);* to see pokkt debug logs and toast messages. please make sure to change this to *PokktCocosManager.SetDebug(false);* for production build.

- Session

1. Starting with this version Pokkt SDK is adding session 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 provide pokktConfig instance for this method with *applicationId* and *securityKey* assigned.
3. You should call *endSession* at the end of his application and once only.

- Video

1. In *PokktCocosConfig* for Video you can set five additional parameters which are *autoCacheVideo,skipEnabled,defaultSkipTime,screenName* and *incentivised*.
2. *autoCacheVideo* 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 *skipEnabled* as true/false.The default value for *skipEnabled* is false.
4. If you have enabled skipped button by setting *skipEnabled* as true then you can control after how many seconds the skip button will be visible in video by setting *defaultSkipTime* to appropriate value.Since most videos will be 30 sec or less please set *defaultSkipTime* as 10 or less.You can also give your own skip message by setting *customSkipMessage* on *PokktCocosConfig*
5. *screenName* 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. ScreenName 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 *incentivised* as true or false. Video gratification will only happen for incentivised playback.
7. You will need to implement all video related events in *PokktCocosManager.*. Please refer to *PokktCocosManager.h* for all events.
8. You can call *PokktCocosManager.isVideoAvailable()* to check if the campaign are available before you try to play video.
9. You can call *PokktCocosManager.GetVideo(pokktCocosConfig);* to play video with incentives and *PokktCocosManager.GetVideoNonIncent(pokktCocosConfig);* to play video without incentives.

10. Please reward user only from the *PokktCocosManager::VideoGratifiedEvent* implementation.

- Optional Parameters
  - *PokktCocosConfig* 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.*

4. Video Ad Functionalities:

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

1. VideoClosedEvent
2. VideoDisplayedEvent
3. VideoSkippedEvent
4. VideoCompletedEvent
5. VideoGratifiedEvent
6. DownloadCompletedEvent
7. 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
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.cacheVideoCampaign();
```

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

```
PokktCocosManager.IsVideoAvailable();
```

You should listen to *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 incentivised (user will be gratified after watching complete video) or non-incentivised (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:

```
pokktCocosConfig.setScreenName(“screen_name”);
PokktCocosManager.GetVideo(pokktConfig);
PokktCocosManager.GetVideoNonIncent(pokktConfig);
```

Next, you can listen to *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.

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

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

6. Important Points

- Please do not copy the code points from this pdf as it may introduce unwanted characters and space in your code.instead please refer to sample app source code in pokkt bundle.
- Please also refer to sample app source code for better understanding of implementation.