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

Contents:

1. Overview:

Thank you for choosing Pokkt SDK Plugin v4.0 for Cocos2dx. Pokkt SDK supports Video-Ad campaigns feature. This document contains all the information that is needed by you to setup the SDK with your project. The current plugin supports mediation for various third party ad-networks. These are:

- AdColony
- AppLovon
- Chartboost
- Fyber
- InMobi
- SuperSonic
- UnityAds
- TapJoy
- Vungle

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 and above, older versions are not supported.

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 Doc/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 v4.0 for Cocos2dx comes in two zip files:

a. **pokktsdk.zip**
b. **libpokkt.zip**

Step 1. 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
    |
```

Step 2. 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's "**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. The contents of "**libpokkt.zip**" should look like following:

```
libpokkt
    | ——lib
    |       | —— libPokktSDK.a
    |
    | ——include
    |       | —— AdNetwork.h
    |       | —— InAppPurchaseRequest.h
    |       | —— NetworkModel.h
    |       | —— POKKMoatBootstrap.h
    |       | —— POKKMoatMobileAppKit.h
    |       | —— POKKMoatTracker.h
    |       | —— POKKMoatVideoTracker.h
    |       | —— PokktConfig.h
    |       | —— PokktController.h
    |       | —— PokktManager.h
    |       | —— PokktNetworkDelegate.h
    |       | —— VideoResponse.h
```

```
|      | —— PokktController.h
|      | —— VideoResponse.h
|      | —— PokktManagerSecurity.plist
|
|—— PokktResource.bundle
```

Step 3. Make sure to add the above "**PokktResource.bundle**" to the your project.

Step 4. Ensure the followings frameworks are present(linked) inside project setting's "**Build-Phases -> Link Binaries With Libraries**", if not than add them manually:

- **CoreData**.framework
- **Foundation**.framework
- **MediaPlayer**.framework
- **SystemConfiguration**.framework
- **UIKit**.framework
- **CoreTelephony**.framework
- **EventKit**.framework
- **AdSupport**.framework

Step 5. Ensure that "**-ObjC**" flag is set inside project setting's "**Build Settings -> Other Linker Flags**".

Step 6. 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];
```

Step 7. 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
{
  [PokktManager callBackgroundTaskCompletionHandler:
        ^(UIBackgroundFetchResult result)
        {
              completionHandler(result);
  }];
}
```

Step 8. 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];
```

Step 9. 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
{
  [PokktManager inAppNotificationEvent:notification];
}
```

Step 10. Open project's **Info.plist** as Source Code. Make an entry of the following:

```
<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>
```

This should be inside the main “dict” node.

Step 11. Ensure that **Enable Bitcode** is set to **false** in the generated project’s **Build Settings**.

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. Refer to Video sections below for information on additional parameters.

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. To use google analytics, please set **setAnalyticsType** and **setGoogleAnalyticsID** in **PokktCocosConfig**. Analytic-type here will be "**GOOGLE_ANALYTICS**".

9. To use flurry analytics please set **setAnalyticsType** and **setFlurryApplicationKey** in **PokktCocosConfig**. Analytic-type here will be "**MIXPANNEL**".

10. To use mix panel analytics please set **setAnalyticsType** and **setMixPanelProjectToken** in **PokktCocosConfig**. Analytic-type here will be "**MIXPANNEL**".

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.

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 call **PokktCocosManager::isVideoAvailable()** to check if the campaigns are available before you try to play video.

8. You can call **PokktCocosManager::getVideo()** to play video.

9. Reward user only from the **onVideoGratified** event.

Export Logs

1. Invoke **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**.

4. Video-Ad Functionalities:

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::startVideoCaching();
```

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 `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 concludes the integration documentation. It is highly suggested that you should check the sample app that is provided to you to understand it better.