
POKKT SDK v1.2 Integration Guide **for Cocos2dx-v3.x (iOS)**

Contents:

1. Introduction
2. Installation
3. PokktManagerSecurity.plist Setup
4. SDK Setup on Cocos2dx-v3.x
5. Video Ad Functionalities
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. 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
|   | --- 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. The contents of "libpokkt.zip" should look like following:

```
libpokkt
| ---lib
|   | --- libPokktSDK.a
|
| ---include
|   | --- PokktManager.h
|   | --- PokktController.h
|   | --- VideoResponse.h
|   | --- PokktManagerSecurity.plist
|
| --- PokktResource.bundle
```

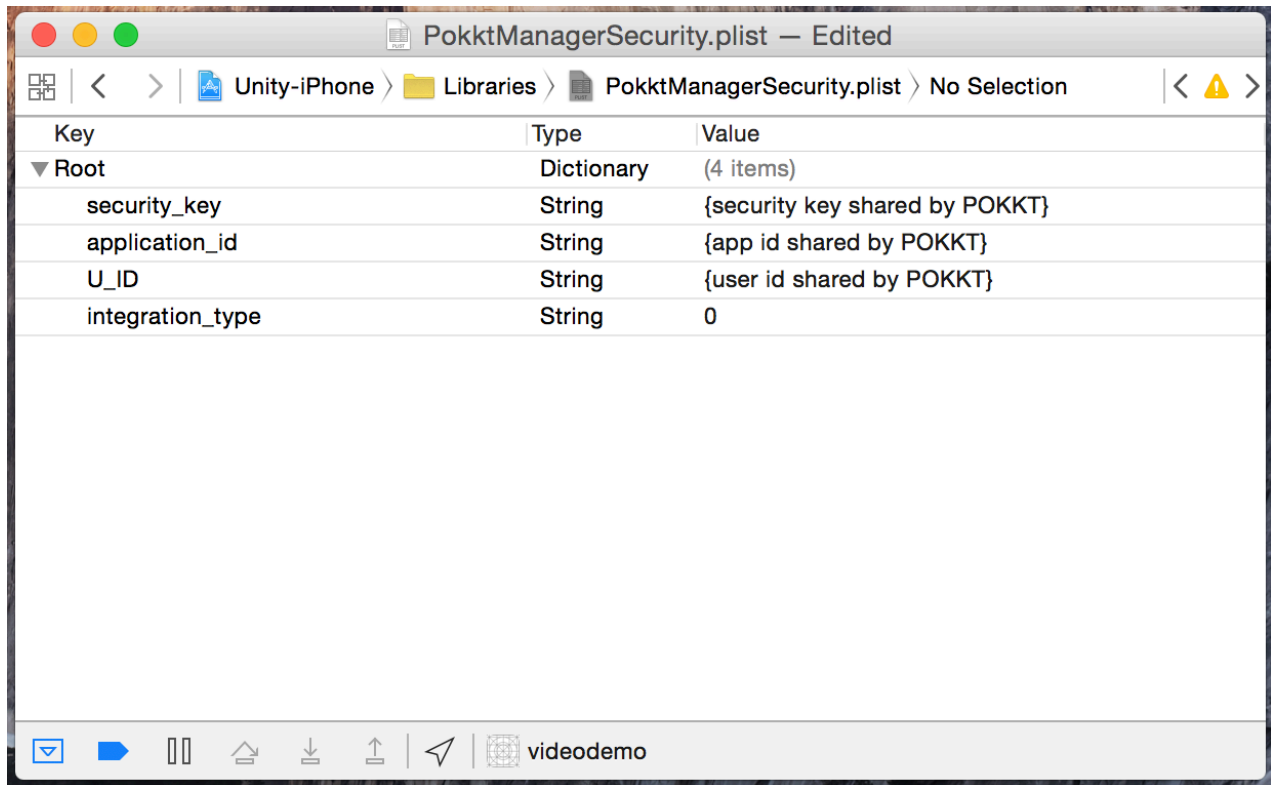
PokktResource.bundle

Make sure to add the "PokktResource.bundle" to the your project.

2. PokktManagerSecurity.plist Setup:

Step 4: Add meta-data information

Refer the following image for reference:



Note:

Leave the “integration_type” value to “0” for now.

4. SDK Setup on Cocos2dx-v3.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 “PokktManagerSecurity.plist”. In order to disable auto-caching of videos, you are required to enter these values in your code and NOT in PokktManagerSecurity.plist.

```
// 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 ONLY AFTER you set the params. Ref.:

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

Adding Frameworks

Make sure to add the following frameworks to your project:

- > SystemConfiguration.framework
- > MediaPlayer.framework

5. 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 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.

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.