POKKT SDK v2.0.0 Integration Guide for Marmalade (iOS)

Contents:

1. Introduction
2. Installation
3. Code Integration
4. Functionalities:
    1. Video
5. Debugging and Logging
6. Important Points

1. Introduction:

Thank you for choosing Pokkt SDK for m Marmalade. This document contains all the information that is needed by you to setup the SDK with your project

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:

All we need is the file provided: *PokktNativeExtension.zip*. And also will provide you one sample project which is *PokktSample.zip* file. Use this as a reference. *VideoScreen.cpp* class is useful for you. Please check how it is implemented. And also here we are giving you complete implementation details.

Export this package and make this as subproject of your project. Please keep as it is this subproject and don't do any changes in this subproject.

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.

4. Code Integration:

Implementation Steps

- Common

1. For all invocation of Pokkt SDK developer will make use of methods available in *PokktNativeManager* class.This class only have static methods.
2. Before calling any other methods from the *PokktNativeManager* 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 *PokktMarmaladeConfig* instance is required. *PokktMarmaladeConfig* is plain object which will hold all the values required by the SDK which you need to assign.
4. In *PokktMarmaladeConfig* you can assign *applicationId*, *securityKey* and *IntegrationType* which are must for all type of integrations.
5. If you are doing server to server integration with pokkt you can also mention *thirdPartyUserId* in *PokktMarmaladeConfig*.
6. Apart from above mentioned parameters you can assign additional ones based on your integration type.(please refer to Video sections below.)
7. While in development please call *PokktNativeManager.setDebug(true);* to see pokkt debug logs and toast messages. please make sure to change this to *PokktNativeManager.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 *PokktNativeManager.*
2. You should call *startSession* at the start of his application and once only. You will need to provide PokktMarmaladeConfig instance for this method with *applicationId*, *securityKey* and *IntegrationType* assigned.
3. You should call *endSession* at the end of his application and once only.

- Video

1. In *PokktMarmaladeConfig* 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 *PokktNativeManager.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 *PokktMarmaladeConfig*
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 can disable the back button while video is playing by setting *backButtonDisabled* on *PokktMarmaladeConfig*.
8. You will need to inherit *PokktDelegate.cpp* class in your class where you want to listen **Video** related events handler and you will have to write all **Video** related method handler. Refer to the *VideoScreen.cpp* class to get the better ideas.
9. You can call *PokktNativeManager::isVideoAvailable()* to check if the campaign is available before you try to play video.

10. You can call *PokktNativeManager::getVideo(PokktMarmaladeConfig);* to play video with incentives and *PokktNativeManager:: getVideoNonIncent(PokktMarmaladeConfig);* to play video without incentives.

11. Please reward user only from the *PokktNativeManager::VideoGratifiedEvent* implementation.

12. Before calling ***getVideo*** please change the orientation. There is some issue with Marmalade for iOS. If your app is running in portrait mode and video plays in Landscape mode then it creates some issue on during video play and also after completed video but if your app is already in landscape mode then no need to do anything. So this is for such app which is running in portrait mode.

    Use below code when you call **getVideo**:

```
if(s3eDeviceGetInt(S3E_DEVICE_OS) == S3E_OS_ID_IPHONE)
        s3eSurfaceSetInt( S3E_SURFACE_DEVICE_ORIENTATION_LOCK,
S3E_SURFACE_LANDSCAPE_FIXED );
```

    And when you receive **videoClosed** event then use below code to change the orientation:

```
if(s3eDeviceGetInt(S3E_DEVICE_OS) == S3E_OS_ID_IPHONE)
    s3eSurfaceSetInt( S3E_SURFACE_DEVICE_ORIENTATION_LOCK,
S3E_SURFACE_PORTRAIT_FIXED );
```

- Optional Parameters

  - *PokktMarmaladeConfig* 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.*

5. Functionalities:

5.1 Video:

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

1. *VideoClosedEvent*
2. *VideoDisplayedEvent*
3. *VideoSippedEvent*
4. *VideoCompletedEvent*
5. *VideoGratifiedEvent*
6. *DownloadCompletedEvent*
7. *DownloadFailedEvent*

Add those handler where you want to listen Video campaign related events. So please check *VideoScreen.cpp* for reference.:

Reference on how to consume them:

```
// handle pokkt video ad events

PokktNativeManager::setListener(PokktNativeManager::VideoClosedEvent,
pokkt_event_selector(VideoScreen::handleVideoClosed), this);

void VideoScreen::handleVideoClosed(std::string message)
{
        // This is important for iOS because Marmalade has some issue with iOS orientation.
        if(s3eDeviceGetInt(S3E_DEVICE_OS) == S3E_OS_ID_IPHONE)
            s3eSurfaceSetInt( S3E_SURFACE_DEVICE_ORIENTATION_LOCK,
        S3E_SURFACE_PORTRAIT_FIXED );
}


PokktNativeManager::setListener(PokktNativeManager::VideoDisplayedEvent,
pokkt_event_selector(VideoScreen::handleVideoDisplayed), this);

void VideoScreen::handleVideoDisplayed(std::string message)
{
}

PokktNativeManager::setListener(PokktNativeManager::VideoSkippedEvent,
pokkt_event_selector(VideoScreen::handleVideoSkipped), this);

void VideoScreen::handleVideoSkipped(std::string message)
{
}
```

```
PokktNativeManager::setListener(PokktNativeManager::VideoCompletedEvent,
pokkt_event_selector(VideoScreen::handleVideoCompleted), this);

void VideoScreen::handleVideoCompleted(std::string message)
{
}


PokktNativeManager::setListener(PokktNativeManager::VideoGratifiedEvent,
pokkt_event_selector(VideoScreen::handleVideoGratified), this);

void VideoScreen::handleVideoGratified(std::string coins)
{
    float coinsFloat = 0;
    std::stringstream(coins) >> coinsFloat;
    _coinsEarned += coinsFloat;
    char buf[100] = "";
    sprintf(buf, "%f", _coinsEarned);
    pointsEarnedLbl->SetText(buf);
}


PokktNativeManager::setListener(PokktNativeManager::DownloadCompletedEvent,
pokkt_event_selector(VideoScreen::handleDownloadCompleted), this);

void VideoScreen::handleDownloadCompleted(std::string coinsToEarn)
{
}


PokktNativeManager::setListener(PokktNativeManager::DownloadFailedEvent,
pokkt_event_selector(VideoScreen::handleDownloadFailed), this);

void VideoScreen::handleDownloadFailed(std::string message)
{
}
```

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:

```
PokktNativeManager::cacheVideoCampaign();
```

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

```
PokktNativeManager::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:

```
PokktMarmaladeConfig::setScreenName(“screen_name”);
PokktNativeManager::getVideo(PokktMarmaladeConfig);
PokktNativeManager::getVideoNonIncent(Pokkt
MarmaladeConfig);
```

Next, you can listen to *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.

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

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

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

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

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

7. 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.