POKKT SDK v3.0.0 Integration Guide for ShiVa (iOS)

Contents:

1. Introduction:

Thank you for choosing Pokkt SDK for ShiVa. This document contains all the information which is needed to implement Pokkt SDK.

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 v1.0.0 for ShiVa comes in two file. One is ".ste" files and one ".zip" file. Here is the details

Step 1:

1. **com.pokkt.extension.ste:** This is the plugin which is need to import in project from ShiVa editor (**Data Explorer->Import->Archive)**.

2. **PokktPluginSample.ste:** This is the Sample Project and it contains few AIModel and HUD. This sample project will show the video. One of the AI_Model (which is **AI_POKKT**) from this sample is very important to implement PokktSDK.

3. **PokktSDKResource.bundle:** This is the resource which is needed so add this in iOS project under resource folder.

Step 2:

If you are ready to build the XCode project using ShiVa Editor then please add below four header file under **Additional file** option. You will get these files in SDK folder.

1. VideoResponse.h
2. PokktManager.h
3. PokktController.h
4. PokktConfig.h

Once your project is built and the XCode project is exported, open this exported project using XCode(It may open itself once you build the project from ShiVa editor). Ensure the followings frameworks are present(linked) inside project setting's "**Build-Phases -> Link Binaries With Libraries**", if not then add them manually:

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

Step 3:

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

Step 4.

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 (S3DEngine_AppDelegate.m). You will get this class once xcode project gets imported.

```
[application setMinimumBackgroundFetchInterval:
UIApplicationBackgroundFetchIntervalMinimum];
```

Step 5:

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 6.

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 7.

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];
}
```

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.

3. Code Integration:

Implementation Steps

Import the POKKT extension (com.pokkt.extension.ste) and follow the below steps:

- Import com.pokkt.extension.ste in ShiVa Editor from Data Explorer. After importing, this can be seen in project under plugin folder as PokktNativeExtension. Now drag this plugin from Data Explorer into Game Editor under plugin section.

- Now import PokktPluginSample.ste where you will see multiple AIModel and HUD. This sample project contains one important AIModel which is AI_POKKT. **This is mandatory to use in your project**. This sample project contains few more AIModel which listen event and those gets trigger from AI_POKKT handler. So update the AI_POKKT AIModel handler according to your AIModel.

  Here is the example how to do that:

  This is one of the event ("onDownloadCompleted Handle") from AI_POKKT.

  user.sendEvent (application.getCurrentUser ( ), "AI_PlayVideo", "onDownloadCompleted Handle", values)

  Now I want to dispatch event from this model to other model where we actually need it so Here "AI_PlayVideo" is the AIModel name where we want to listen "onDownloadCompleted" event. So this you need to change according to your AIModel.

- Now export the game from Data Explore. On clicking export button will show one popup where select "Runtime Package (.stk)" and select iOS as option and then export it. Then it will generate .stk file.

- Once project has been exported then open ShiVa Editor (UAT 2.0.0 beta) in OSX and do the all required setting for iPhone or iPad in Settings option. There you need to assign path for Xcode developer.

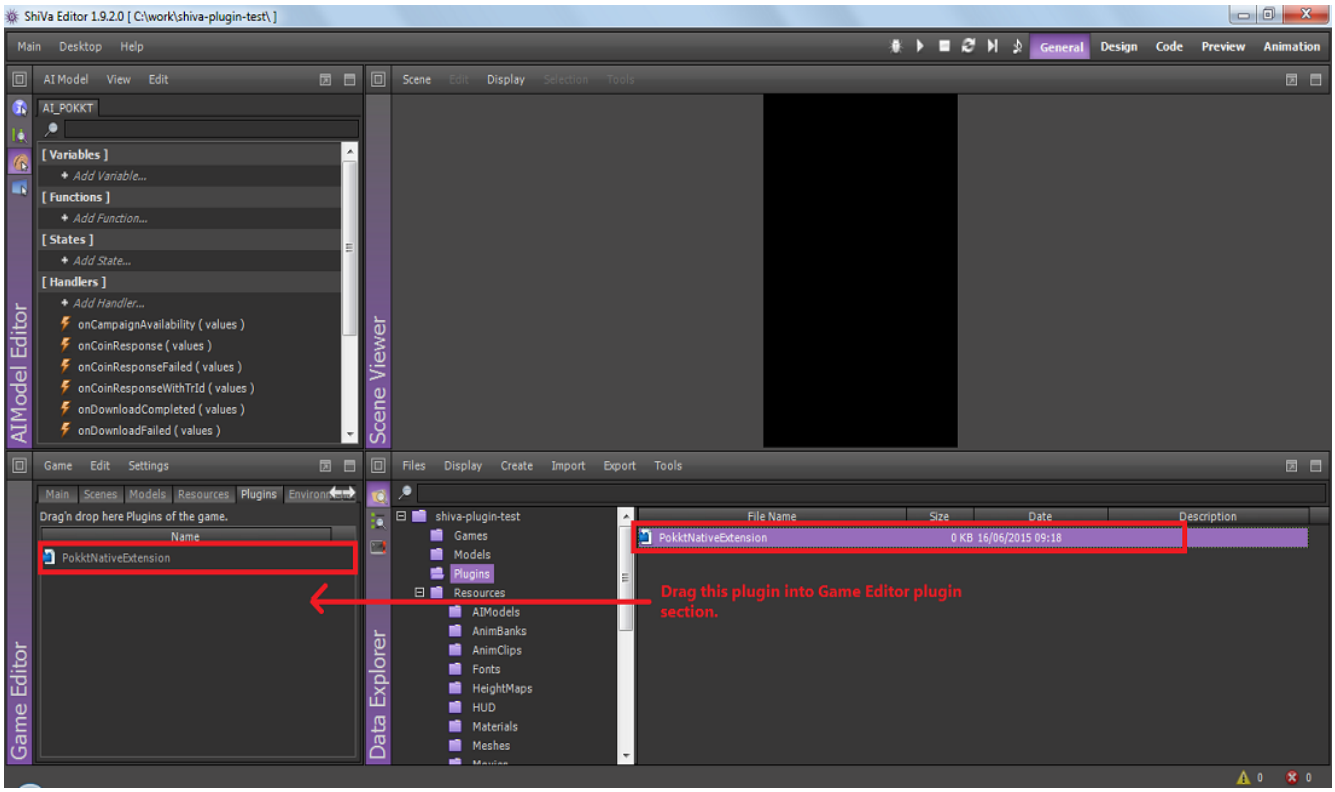
  Requirement for making iOS build as suggested by ShiVa team:

  - ShiVa UAT 2.0.0 beta

  Once it is done then follow the below steps:

  1. Select Content tab and give the path of ".stk" file.
  2. Select Authoring tab and choose Authoring type as "Project". Also give the Bundle identifier and all required data.

  Once all these two steps are done then click on Build button which will export the iOS Project and will open Xcode. And here add the .bundle file which was given with downloaded package. Add this under resource folder.



Screen shot: How to add Pokkt plugin in your project.

- Common

1. First we need to set SET_APPLICATION_ID, SET_SECURITY_KEY and SET_AUTO_CACHING with calling performOperation method like below:
   PokktNativeExtension.performOperation (PokktNativeExtension.SET_SECURITY_KEY, "iJ02lJss0M").
   So set all three properties. Please follow the sample code(AI_MainMenu onTestVideoClicked method).
   Once this is done then calling INIT_POKKT is mandatory which will initialize Pokkt SDK. Nothing to pass in parameter for this so pass empty string("").

2. If you are doing server to server integration then please perform operation:
   PokktNativeExtension.SET_THIRD_PARTY_USER_ID.

3. While in development please call
   PokktNativeExtension.performOperation (PokktNativeExtension.SET_DEBUG, "true")
   to see Pokkt debug.
   please make sure to change this for production build.
   PokktNativeExtension.performOperation (PokktNativeExtension .SET_DEBUG, "false").

4. Please call *PokktNativeExtension.SEND_APP_INFO* to send your application installation information to Pokkt.

5. To use google analytics, please set *SET_SELECTED_ANALYTICS_TYPE* as *PokktNativeExtension.GOOGLE_ANALYTICS* and set *SET_GOOGLE_ANALYTICS_ID* with passing Google analytics ID.

6. To use flurry analytics, please set *SET_SELECTED_ANALYTICS_TYPE* as *PokktNativeExtension.FLURRY* and set *SET_FLURRY_APPLICATION_KEY* with passing Flurry app key.

7. To use MixPanel analytics, please set *SET_SELECTED_ANALYTICS_TYPE* as *PokktNativeExtension.MIXPANEL* and set *SET_MIX_PANEL_PROJECT_ID* with passing MixPanel project ID.

- Session

    1. Pokkt SDK is adding session tracking for which we have *PokktNativeExtension. START_SESSION .*

    2. *START_SESSION* should be called at the start of the application and once only also will be need to provide *applicationId*, *securityKey* and *IntegrationType* so for this please call setParam before calling *START_SESSION*.

    3. We should call *PokktNativeExtension.END_SESSION* at the end of his application and once only

- Video

    1. For video you can set five additional parameter which are autoCacheVideo, skipEnabled, defaultSkipTime, screenName and incentivised.

    2. autoCacheVideo is required if you want to automatically cache video on user device. You need to set this parameter in SET_PARAMS as true. If you set it as false then video will not be automatically cached and will have to call *PokktNativeExtension.performOperation (PokktNativeExtension .CACHE_VIDEO_CAMPAIGN, "")* to start caching videos on device.

    3. If you want to enable/disable the skip button on video screen please set *SET_SKIP_ENABLED* as "true"/"false". The default value for *SET_SKIP_ENABLED* is false. You can call like this:
    PokktNativeExtension.performOperation (PokktNativeExtension. SET_SKIP_ENABLED, "true")

- Optional Parameters

  - There are also 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.* For reference, please check *AI_MainMenu* AIModel.

- Export Logs

  1. Invoke *PokktNativeExtension.EXPORT_LOG* 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

4. Functionalities:

5.1 Video:

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

- onDownloadCompleted
- onDownloadFailed
- onVideoClosed
- onVideoCompleted
- onVideoDisplayed
- onVideoGratified
- onVideoSkipped

Reference on how to consume them:

```
function AI_POKKT.onDownloadCompleted ( values )
---------------------------------------------------------------------------
user.sendEvent ( application.getCurrentUser ( ), "AI_PlayVideo", "onDownloadCompleted
Handle", values )
---------------------------------------------------------------------------
end
```

This event will trigger on video download completed and values contains total earned points.

```
function AI_POKKT. onDownloadFailed ( values )
---------------------------------------------------------------------------
user.sendEvent ( application.getCurrentUser ( ), "AI_PlayVideo", " onDownloadFailedHandle",
values )
---------------------------------------------------------------------------
end
```

This event will trigger on video download failed. Here values doesn't have any data. It is empty and it is not useful.

```
function AI_POKKT. onVideoClosed ( values )
---------------------------------------------------------------------------
user.sendEvent ( application.getCurrentUser ( ), "AI_PlayVideo", " onVideoClosedHandle", values
)
---------------------------------------------------------------------------
end
```

This event will trigger on video close. Here values doesn't have any data. It is empty and it is not useful.

```
function AI_POKKT. onVideoCompleted ( values )
---------------------------------------------------------------------------
user.sendEvent ( application.getCurrentUser ( ), "AI_PlayVideo", " onVideoCompletedHandle",
values )
---------------------------------------------------------------------------
end
```

This event will trigger on video completed. Here values doesn't have any data. It is empty and it is not useful.

function AI_POKKT. **onVideoDisplayed** (values)
---
user.sendEvent (application.getCurrentUser ( ), "AI_PlayVideo", " onVideoDisplayedHandle", values)
---
end
This event will trigger on video displayed. Here values doesn't have any data. It is empty and it is not useful.

function AI_POKKT. **onVideoGratified** (values)
---
user.sendEvent (application.getCurrentUser ( ), "AI_PlayVideo", " onVideoGratifiedHandle", values)
---
end
This event will trigger on video gratified. So here values will have earned points in string format.

function AI_POKKT. **onVideoSkipped** (values)
---
user.sendEvent (application.getCurrentUser ( ), "AI_PlayVideo", " onVideoSkippedHandle", values)
---
end
This event will trigger on video skipped. Here values doesn't have any data. It is empty and it is not useful.

A video file is cached on user's device. You can set the auto-caching option in the beginning. In case of manual caching, call the following method to start video caching:

```
PokktNativeExtension.performOperation ( PokktNativeExtension.CACHE_VIDEO_CAMPAIGN, "" );
```

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

```
PokktNativeExtension.isVideoAvailable ( )
```

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:

```
PokktNativeExtension.performOperation ( PokktNativeExtension.GET_VIDEO, "screen_name" )
PokktNativeExtension.performOperation ( PokktNativeExtension.GET_VIDEO_NON_INCENT, " screen_name " )
```

Next, AI_POKKT.onVideoGratified callback will call 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.

```
PokktNativeExtension.performOperation ( PokktNativeExtension.SET_DEBUG, "true" )
```

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

```
PokktNativeExtension.performOperation ( PokktNativeExtension.SHOW_LOG, "showing log" )
```

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.