

---

# **POKKT SDK v3.1 Integration Guide for Appcelerator (Android)**

---

## Contents:

---

1. Overview
2. Configuration Steps
3. Implementation Steps
4. Functionalities:
  1. Offerwall
  2. Video
5. Debugging and Logging

---

## 1. Overview:

---

Thank you for choosing Pokkt SDK Plugin v3.1 for Appcelerator. Pokkt SDK supports Offerwall as well as Video-Ad campaigns feature. This document contains all the information that is needed by you to setup the SDK with your project. Please follow these steps as per your integration requirement (Video/Offerwall/Both).

Kindly note that these instructions are for Appcelerator with Titanium SDK Version 5.x and above, older versions of 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.

---

## 2. Configuration Steps:

---

The Plugin/Module comes in the file with name “**com.pokkt.titaniumandroid-android-3.1.0.zip**”. In order to add this to your project, go to **Help -> Install Mobile Module**. Now point to this file and press “**Ok**”. This should install the plugin with your project and you should be able to use it inside your project. There is another helper file “**pokkt\_native\_extension.js**” provided, you are supposed to copy this inside your Resource folder.

**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.

---

### 3. Implementation Steps:

---

#### Common

1. For all invocation of Pokkt SDK developer will make use of methods available in **pokkt\_native\_extension.js** file inside **PokktExtension** object.
2. In **PokktExtension** 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 **PokktExtension**. 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 **PokktExtension**.
5. Apart from above mentioned parameters you can assign additional ones based on your integration type. Refer to OfferWall and Video sections below.
6. While in development please call **PokktExtension.setDebug(true)** to see Pokkt debug logs and toast messages. please make sure to change this to **false** for production build.
7. Please call **PokktExtension.sendAppInfo()** to send your application installation information to Pokkt.
8. To use google analytics, please set **setAnalyticsType** and **setAnalyticsID** in **PokktExtension**.
9. To use flurry analytics please set **setAnalyticsType** and **setFlurryApplicationKey** in **PokktExtension**.
10. To use mix panel analytics please set **setAnalyticsType** and **setMixPanelProjectToken** in **PokktExtension**.

#### Session

1. We have option to start session for tracking for which we have **startSession** and **endSession** methods in **PokktExtension**.
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.

---

## Offerwall

1. In **PokktExtension** for Offerwall you can set two additional parameters which are **setOfferWallAssetValue** and **setCloseOnSuccessFlag**. Setting of **setOfferWallAssetValue** is only required if you only want to show campaign of certain value on the Offerwall. **setCloseOnSuccessFlag** is required if you wish to auto-close the Offerwall after user has completed one offer. It's default value is false.
2. Make sure to call **initPokkt** before calling another method for Offerwall in **PokktExtension**.
3. You will need to implement **IOfferwallDelegate** interface as mentioned in Step-4 in configuration steps.
4. To show Offerwall you can call **PokktExtension.getCoins()**.
5. In the screen or activity where you have button to show Offerwall, in that activity **onResume** you should call **PokktExtension.getPendingCoins()** so that you get a callback to award points to the user after he has come back to your game after finishing with Offerwall. You will get a callback for this call in your **IOfferwallDelegate** implementation class in method **EarnedCoins** or **CoinResponseFailed**.
6. You can call **PokktExtension.checkCampaignAvailable()** to check whether the campaigns are available before showing Offerwall button to user. You will get a callback for this call in your **IOfferwallDelegate** implementation class in method **onOfferwallCampaignCheck**.

## Video

1. In **PokktExtension** 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 **PokktExtension.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 **PokktExtension**.

- 
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 disable the back button while video is playing by setting **setBackButtonDisabled** on **PokktExtension**.
  8. You will need to create **IVideoDelegate** implementation class as mentioned in Step-4 in configuration steps.
  9. You can call **PokktExtension.isVideoAvailable()** to check if the campaigns are available before you try to play video.
  10. You can call **PokktExtension.getVideo()** to play video.
  11. You will get different callbacks as given in **IVideoDelegate** implementation class for video playback.
  12. Reward user only from the **onVideoGratified** method in **IVideoDelegate** implementation class.

### Optional Parameters

**PokktExtension** 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. Functionalities:

---

### 4.1 Offerwall

There are five events to listen too. These are:

- PokktCoinResponse
- PokktCoinResponseWithTransId
- PokktCoinResponseFailed
- PokktCampaignAvailability
- PokktOfferwallClosed

(Ideally)Add handlers to these events in the Awake() method of your MonoBehaviour class.  
Below are the references on how to use them:

Reference on how to consume them:

```
Ti.App.addEventListener('PokktCoinResponse', function (e) {  
    var points = 'Points Earned: ' + e.params;  
});
```

```
Ti.App.addEventListener('PokktCoinResponseWithTransId', function (e) {  
    var points = 'Points Earned: ' + e.params;  
});
```

```
Ti.App.addEventListener('PokktCoinResponseFailed', function (e) {  
    Ti.API.info('coin response failed!');  
});
```

```
Ti.App.addEventListener('PokktCampaignAvailability', function (e) {  
    Ti.API.info('campaign availablity: ' + e.params);  
});
```

```
Ti.App.addEventListener('PokktOfferwallClosed', function (e) {  
    Ti.API.info('offerwall is closed!');  
});
```

### There are two ways to invoke offerwall.

**Open Asset Value:** In this case, POKKT platform provides all offers with any asset value.  
Sample code snippet:

```
var pe = require('pokkt_native_extension');  
pe.getCoins(0);
```



---

**Fixed Asset Value:** In this case, POKKT platform provides all offers with fixed asset value.

Sample code snippet:

```
var pe = require('pokkt_native_extension');  
pe.getCoins(<asset_value>);
```

**Pending Coins:** In case after completing activity, if status of transaction is pending, then call getPendingCoins() method.

```
var pe = require('pokkt_native_extension');  
pe.getPendingCoins();
```

**Check for campaign availability:** If you are using optional meta-tag as mentioned in Step 5, you can call the following to check for campaign availability:

```
var pe = require('pokkt_native_extension');  
pe.checkOfferWallCampaign();
```

The result will be noticed with the event:

PokktCampaignAvailability

Moreover, you can listen to the following event to know whether offerwall has been closed or not:

PokktOfferwallClosed

---

## 4.2 Video:

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

- PokktVideoClosed
- PokktVideoDisplayed
- PokktVideoSipped
- PokktVideoCompleted
- PokktVideoGratified
- PokktDownloadCompleted
- PokktDownloadFailed

(Ideally) Add handlers to these events in the Awake() method of your MonoBehaviour class. Below are the references on how to use them:

Reference on how to consume them:

```
Ti.App.addEventListener('PokktDownloadCompleted', function (e) {  
    Ti.API.info('video is downloaded!');  
});
```

```
Ti.App.addEventListener('PokktDownloadFailed', function (e) {  
    Ti.API.info('video download failed!');  
});
```

```
Ti.App.addEventListener('PokktVideoDisplayed', function (e) {  
    Ti.API.info('video is displayed!');  
});
```

```
Ti.App.addEventListener('PokktVideoClosed', function (e) {  
    Ti.API.info('video is closed!');  
});
```

```
Ti.App.addEventListener('PokktVideoSkipped', function (e) {  
    Ti.API.info('video is skipped!');  
});
```

```
Ti.App.addEventListener('PokktVideoCompleted', function (e) {  
    Ti.API.info('video is completed!');  
});
```

```
Ti.App.addEventListener('PokktVideoGratified', function (e) {  
    Ti.API.info('video is gratified!');  
});
```

---

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:

```
var pe = require('pokkt_native_extension');  
pe.cacheVideoCampaign();
```

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

```
var pe = require('pokkt_native_extension');  
var isAvailable = pe.isVideoAvailable();
```

You should listen to **PokktDownloadCompleted** 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 be made:

```
var pe = require('pokkt_native_extension');  
if (incentivised)  
    pe.getVideo('sample_screen');  
else  
    pe.getVideoNonIncent('sample_screen');
```

Next, you can listen to **PokktVideoGratified** 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.:

```
pe.setDebug(true);
```

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

```
pe.showLog('sampleApp');
```

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

```
pe.showToast('sampleApp');
```

---

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.