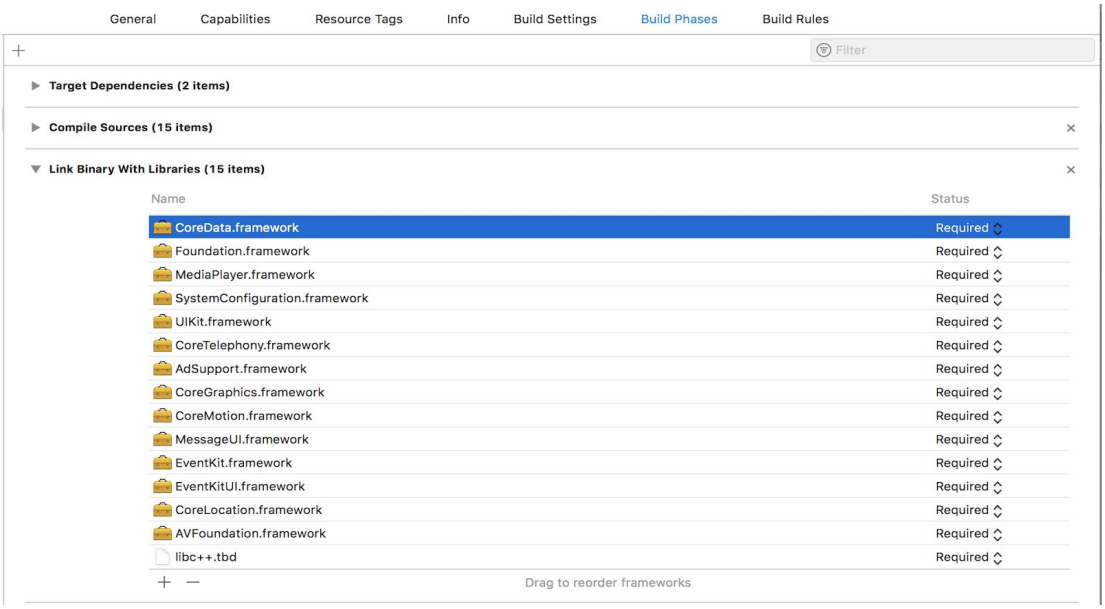
PokktSDK Integration Guide (iOS, v8.0.1)

Contents

- SDK Integration

- Download the latest SDK. [LINK]
- Unzip the downloaded file and drag the PokktSDK.framework directory into Xcode under Framework.
- Add the required compiler flags to 'Other Linker Flags' field in your project's Build Settings:
    - **-ObjC**
    - **-lxml2**

- Make sure that these frameworks are included to respect the library dependencies:



- SDK Configuration

1. Set **Application Id** and **Security key** in Pokkt SDK. You can get it from the Pokkt dashboard from your account. We generally assign unique application-id and security-key.

```
[PokktAds setPokktConfigWithAppId:appId securityKey:securityKey];
```

2. When your application is under development and if you want to see sdk logs and other informatory messages, you can enable it by setting **setDebug** to **true**. Make sure to disable debugging before release.

```
[PokktDebugger setDebug:<YES/NO>];
```

***For More SDK Configuration. See Section More Settings**

- Full-Screen Ads

1. Full-screen ads can be rewarded or non-rewarded and video or interstitial. You can either cache the ad in advance or directly call show for it.
2. We suggest you cache the ad in advance so as to give seamless play behaviour, In other case it will stream the video which may lead to unnecessary buffering delays depending on the network connection.

3. **Screen-Id:** This one parameter is accepted by almost all API's of Pokkt SDK. This controls the placement of ads and can be created on Pokkt Dashboard. We will be referencing the **PokktAdsDemo** app provided with SDK during the course of explanation in this document. We suggest you go through the sample app for better understanding.

4. To cache Full-screen ad call:

```
[PokktAds cacheAd:screenId withDelegate:delegate];
```

5. To Show Video Ad:

```
[PokktAds showAd:screenId withDelegate:delegate presentingVC:viewcontroller];
```

You can check if an ad is available, before making a show request.

```
[PokktAds isAdCached:screenId];
```

6. Ad actions are optional, but we suggest to implement them as it will help you to keep track of the status of your ad request.

```
- (void)adCachingResult:(NSString *)screenId
              isSuccess:(BOOL)success
             withReward:(double)reward
           errorMessage:(NSString *)errorMessage;

- (void)adDisplayResult:(NSString *)screenId
              isSuccess:(BOOL)success
           errorMessage:(NSString *)errorMessage;

- (void)adClosed:(NSString *)screenId
    adCompleted:(BOOL)adCompleted;

- (void)adClicked:(NSString *)screenId;

- (void)adGratified:(NSString *)screenId
         withReward:(double)reward;

- (void)adReady:(NSString *)screenId
   withNativeAd:(PokktNativeAd *)pokktNativeAd;

- (void)adFailed:(NSString *)screenId
           error:(NSString *)errorMessage;
```

- ## Banner Ads

1. Load banner Ad call:

```
UIView  *banner = [UIView initWithFrame:CGRectMake(0, 0, [UIScreen
mainScreen].bounds.size.width, 50)];
[self.view addSubview:banner];

[PokktAds showAd:screenTF.text withDelegate:self
     inContainer:banner];
```

2. Refresh banner:

   You can set a banner refresh rate on the Pokkt dashboard. Refresh rate should be in range of 30 -100

3. Destroy banner

```
[PokktAds dismissAd: screenId;
```

4. Ad actions are optional, but we suggest to implement them as it will help you to keep track of the status of your ad request.

```
- (void)adReady:(NSString *)screenId
   withNativeAd:(PokktNativeAd *)pokktNativeAd;

- (void)adFailed:(NSString *)screenId
           error:(NSString *)errorMessage;

- (void)adClicked:(NSString *)screenId;
```

- ## Native Ads

1. A native ad may be served in feed or in between the developer content inside the app. Normally, FullScreen ads are delivered on call to action by the user. Native ads eliminate this limitation and show ads without any user request.Native ads are also non intrusive as they will be automatically paused when the ad is out of the view by scrolling. The PokktNativeAdLayout will be part of developer application parent components which may be ListView, Layout in ScrollView or WebView.

   Load Native Ad:

```
[PokktAds requestNativeAd:(NSString *)screenId withDelegate:self];
```

2. Ad actions are optional, but we suggest to implement them as it will help you to keep track of the status of your ad request.

```
- (void)adReady:(NSString *)screenId
   withNativeAd:(PokktNativeAd *)pokktNativeAd;

- (void)adFailed:(NSString *)screenId
           error:(NSString *)errorMessage;

- (void)adClicked:(NSString *)screenId;

// inside adReady:withNativeAd: method you will receive the native ad, you
can access it by following code ->
UIView *adView = [pokktNativeAd getMediaView];
```

Additional SDK Configuration

GDPR

As of May 25th, the General Data Protection Regulation (GDPR) will be enforced in the European Union.
Set ***GDPR consent*** in Pokkt SDK. **This must be called before calling any ad related API. Developers/Publishers must get the consent from user.** For more information on GDPR please refer https://www.eugdpr.org/ and https://www.eugdpr.org/gdpr-faqs.html. This API can again be used by publishers to revoke the consent. If this API is not called or invalid data provided then SDK will access the users personal data for ad targeting

```
PokktConsentInfo *consentInfo = [[PokktConsentInfo alloc] init];
consentInfo.isGDPRApplicable = true;
consentInfo.isGDPRConsentAvailable = false;
[PokktAds setPokktConsentInfo:consentInfo];
[PokktAds getPokktConsentInfo];
```

Extra Parameter

You can set extra parameters to POKKT SDK, to be passed back to your server via POKKT server callback. These Extra parameters will be in key-value pair.The key must be alphanumeric value. See the below example

```
NSMutableDictionary *dict = [[NSMutableDictionary alloc] initWithCapacity:0];
[dict setValue:@"value1" forKey:@"key1"];
[dict setValue:@"value2" forKey:@"key2"];
[dict setValue:@"value3" forKey:@"key3"];
[PokktAds setCallbackExtraParam:dict];
```

- ## ThirdPartyUser Id

  If you are using server to server integration with Pokkt, you can also set **Third Party UserId** in PokktAds.

```
PokktAds.setThirdPartyUserId:(NSString*) userId;
```

- ## User Detail

  For better targeting of ads you can also provide user details to our SDK using

```
PokktUserInfo *pokktUserDetails = [PokktUserInfo alloc] init];
pokktUserDetails.Name = "";
pokktUserDetails.Age = "";
pokktUserDetails.Sex = "";
pokktUserDetails.MobileNo = "";
pokktUserDetails.EmailAddress = "";
pokktUserDetails.Location = "";
pokktUserDetails.Birthday = "";
pokktUserDetails.MaritalStatus = "";
pokktUserDetails.FacebookId = "";
pokktUserDetails.TwitterHandle = "";
pokktUserDetails.Education = "";
pokktUserDetails.Nationality = "";
pokktUserDetails.Employment = "";
pokktUserDetails.MaturityRating = "";

[PokktAds setUserDetails: pokktUserDetails];
```

● Pokkt AdPlayer Configuration

Pokkt Ad player works the way the App is configured at Pokkt dashboard, but we provide a way to override those settings using **PokktAdPlayerViewConfig**.
Application should prefer configuration provided through code by the developer or what's configured for the app in the dashboard, can be controlled any time through the dashboard itself. If you want to make changes to this configuration after your app distribution, you can contact **Pokkt Team** to do the same for your app through the admin console.

```
PokktAdPlayerViewConfig *adPlayerViewConfig =
    [[PokktAdPlayerViewConfig alloc] init];
[PokktAds setPokktAdPlayerViewConfig:adPlayerViewConfig];
```

Various properties that can be managed through this are:

1. **Default skip time**

Defines the time after which the user can skip the Ad.
Property name: DefaultSkipTime

Values: Any Integer value. Default value is 10 seconds .

2. Should allow skip

Defines if a user is allowed to skip the Ad or not.
Property name**:** ShouldAllowSkip
Values**:** YES/NO.

3. Should allow mute

Defines if a user is allowed to mute the Video Ad or not.
Property name**:** ShouldAllowMute
Values**:** YES/NO.

4. Should confirm skip

Defines if confirmation dialog is to be shown before skipping the Ad.
Property name**:** ShouldConfirmSkip
Values**:** YES/NO.

5. Skip confirmation message

Defines what confirmation message to be shown in skip dialog.
Property name**:** SkipConfirmMessage
Values**:** Any String message. Default value is “Skipping this video will earn you NO rewards. Are you sure?”.

6. Affirmative label for skip dialog

Defines what should be the label for the affirmative button in skip dialog.
Property name**:** SkipConfirmYesLabel
Values**:** Any String message. Default value is a “Yes”.

7. Negative label for skip dialog

Defines what should be the label for the affirmative button in skip dialog.
Property name**:** SkipConfirmNoLabel
Values**:** Any String message. Default value is “No”.

8. Skip timer message

Defines the message to be shown before enabling the skip button. Don’t forget to add placeholder “**##**” in your custom message.
This placeholder is replaced by property “Default skip time” assigned above.
Property name**:** SkipTimerMessage
Values**:** Any String message. Default value is “You can skip video in ## seconds”

9. Incentive message

Defines the message to be shown during video progress, that after what time the user will be incentivised.
Property name**:** IncentiveMessage
Values**:** Any String message Default value is “more seconds only for your reward!”

10. AudioEnabled

AudioEnabled set YES or NO to mute ad,if 'YES' ad will be mute. , default is NO**.**
Property name**:** isAudioEnabled
Values**:** YES/NO.

11. Should collect feedback

Defines the message to be shown during video progress, that after what time the user will be incentivised.
***Property name*:** setShouldCollectFeedback
***Values*:** YES/NO.