



POKKT SDK Integration Guide (v 7.2.0)

Xamarin

Overview	2
Implementation Steps	3
SDK Configuration	3
Ad Types	4
Video	4
Rewarded	4
Non Rewarded	4
Check if Ad is already cached	4
Interstitial	4
Rewarded	4
Non Rewarded	5
Check if Ad is already cached	5
Banner	5
Ad Actions	6
Video	6
Interstitial	6
Banner	6
Pokkt ad player configuration	7
User Details	9
Pokkt Server Callback Params	9
Debugging	10
Analytics	11
Google Analytics	11
Flurry Analytics	11
MixPanel Analytics	11
Fabric Analytics	11
IAP(In App Purchase)	12
Project Configuration	13
Android	13
Dependencies	13
Manifest	13
Permissions Declarations	13
Activity Declaration	14
Service Declaration	14
iOS	14
Dependencies	14

Framework	15
Info.plist	15



Overview

Thank you for choosing **Pokkt SDK** for **Xamarin**. This document contains all the information required to set up the SDK with your project. We also support mediation for various third party networks. To know the supported third party networks and their integration process go to mediation section.

Before implementing plugins it is mandatory to go through [project configuration](#) and [implementation steps](#), as these sections contain mandatory steps for basic SDK integration and are followed by every plugin.

You can download our SDK from pokkt.com.

ScreenName: 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 **PokktAds Demo** app provided with SDK . We will be referencing this app during the course of explanation in this document. We suggest you go through the sample app for better understanding.



Implementation Steps

SDK Configuration

1. You need to set extension before calling any method like below. This should be the first call.

```
PokktAds.SetNativeExtentions(new AndroidExtension(this));  
PokktAds.SetNativeExtentions(new IosExtension(this));
```

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

```
PokktAds.SetPokktConfig("<Pokkt Application ID>", "<Pokkt Security Key>");
```

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

```
PokktAds.SetThirdPartyUserId("<Third party user Id>");
```

4. 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-fags.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.

```
PokktAds.ConsentInfo consentInfo = new PokktAds.ConsentInfo();  
consentInfo.GDPRApplicable = true;  
//true if GDPR is applicable.  
consentInfo.GDPRConsentAvailable = false;  
//false if user has given consent to use personal details for ad targeting.  
PokktAds.SetDataAccessConsent(consentInfo);
```

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

```
PokktAds.Debugging.ShouldDebug(<true>);
```



Ad Types

Video

- Video ad can be rewarded or non-rewarded. You can either cache the ad in advance or directly call show for it.
- We suggest you to 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.

Rewarded

1. To cache rewarded ad call:

```
PokktAds.VideoAd.CacheRewarded("<ScreenName>");
```

2. To show rewarded ad call:

```
PokktAds.VideoAd.ShowRewarded("<ScreenName>");
```

Non Rewarded

1. To cache non-rewarded ad call:

```
PokktAds.VideoAd.CacheNonRewarded("<ScreenName>");
```

2. To show non-rewarded ad call:

```
PokktAds.VideoAd.ShowNonRewarded("<ScreenName>");
```

Check if Ad is already cached

```
PokktAds.VideoAd.isAdCached("<ScreenName>", <true / false>);
```

Interstitial

Rewarded

1. To cache rewarded ad call:

```
PokktAds.Interstitial.CacheRewarded("<ScreenName>");
```

2. To show rewarded ad call:

```
PokktAds.Interstitial.ShowRewarded("<ScreenName>");
```

Non Rewarded

1. To cache non-rewarded ad call:

```
PokktAds.Interstitial.CacheNonRewarded("<ScreenName>");
```

2. To show non-rewarded ad call:

```
PokktAds.Interstitial.ShowNonRewarded("<ScreenName>");
```

Check if Ad is already cached

```
PokktAds.Interstitial.isAdCached("<ScreenName>", <true / false>);
```

Banner

There are two ways to load banners:

1. Load banner

```
PokktManager.LoadBanner(<ScreenName>, <BannerPosition>);
```

There are predefined positions for banner positioning inside **BannerPosition** (Com.

Pokkt.Plugin.Common.BannerPosition).

- TOP_LEFT.
- TOP_CENTER.
- TOP_RIGHT.
- MIDDLE_LEFT.
- MIDDLE_CENTER.
- MIDDLE_RIGHT.
- BOTTOM_LEFT.
- BOTTOM_CENTER.
- BOTTOM_RIGHT.

2. Load banner with rect

```
PokktAds.Banner.LoadBannerWithRect(<ScreenName>,<Height>, Width,<br><x>,<y>);
```

Height : Custom height for banner.



Width: Custom width for banner.
x: Point x on screen to show banner.
y: Point y on screen to show banner.

You can remove Banner using:

```
PokktAds.Banner.RemoveBanner();
```

Ad Actions

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.

Video

```
PokktAds.VideoAd.AdAvailabilityEvent += AdAvailabilityStatus;  
PokktAds.VideoAd.AdCachingCompletedEvent += AdCachingCompleted;  
PokktAds.VideoAd.AdCachingFailedEvent += AdCachingFailed;  
PokktAds.VideoAd.AdDisplayedEvent += AdDisplayed;  
PokktAds.VideoAd.AdFailedToShowEvent += AdFailedToShow;  
PokktAds.VideoAd.AdSkippedEvent += AdSkipped;  
PokktAds.VideoAd.AdCompletedEvent += AdCompleted;  
PokktAds.VideoAd.AdClosedEvent += AdClosed;  
PokktAds.VideoAd.AdGratifiedEvent += AdGratified;
```

Interstitial

```
PokktAds.Interstitial.AdAvailabilityEvent += AdAvailabilityStatus;  
PokktAds.Interstitial.AdCachingCompletedEvent += AdCachingCompleted;  
PokktAds.Interstitial.AdCachingFailedEvent += AdCachingFailed;  
PokktAds.Interstitial.AdDisplayedEvent += AdDisplayed;  
PokktAds.Interstitial.AdFailedToShowEvent += AdFailedToShow;  
PokktAds.Interstitial.AdSkippedEvent += AdSkipped;  
PokktAds.Interstitial.AdCompletedEvent += AdCompleted;  
PokktAds.Interstitial.AdClosedEvent += AdClosed;  
PokktAds.Interstitial.AdGratifiedEvent += AdGratified;
```

Banner

```
PokktAds.Banner.BannerLoadedEvent += bannerLoaded;  
PokktAds.Banner.BannerLoadFailedEvent += bannerLoadFailed;
```



Pokkt ad player configuration

Pokkt Ad player works the way App is configured at Pokkt dashboard, but we provide a way to override those settings using **PokktAdViewConfig**.

Application should prefer configuration provided through code by developer or what's configured for the app in 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 admin console.

```
PokktAdViewConfig adViewConfig = new PokktAdViewConfig ();  
// set properties values to adPlayerViewConfig  
PokktAds.SetAdPlayerViewConfig(adViewConfig );
```

Various properties that can be managed through this are:

1. Back button

Defines if user is allowed to close the Advertisement by clicking on back button or not.

Property name : BackButtonDisabled

Values:

True = Back button is disabled and user cannot close the Ad.

False = Back button is not disabled and user can close the Ad.

2. Default skip time

Defines the time after which user can skip the Ad.

Property name: DefaultSkipTime

Values:

Any Integer value.

Default value is 10 seconds .

3. Should allow skip

Defines if user is allowed to skip the Ad or not.

Property name: ShouldAllowSkip

Values:

True = User can skip Ad.

False = User can't skip Ad.

4. Should allow mute

Defines if user is allowed to mute the Video Ad or not.

Property name: ShouldAllowMute

Values:

True = User can mute video Ad.

False = User can't mute video Ad.

5. Should confirm skip

Defines if confirmation dialog is to be shown before skipping the Ad.

Property name: ShouldConfirmSkip

Values:

True = Confirmation dialog will be shown before skipping the video.

False = Confirmation dialog will not be shown before skipping the video.

6. 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?".

7. Affirmative label for skip dialog

Defines what should be the label for affirmative button in skip dialog.

Property name: SkipConfirmYesLabel

Values:

Any String message.

Default value is "Yes".

8. Negative label for skip dialog

Defines what should be the label for affirmative button in skip dialog.

Property name: SkipConfirmNoLabel

Values:

Any String message.

Default value is "No".

9. Skip timer message

Defines message to be shown before enabling 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"

10. Incentive message

Defines message to be shown during video progress, that after what time user will be incentivised.

Property name: IncentiveMessage

Values:

Any String message

Default value is "more seconds only for your reward !"

11. Should collect feedback

Defines message to be shown during video progress, that after what time user will be incentivised.

Property name: ShouldCollectFeedback

Values:

True = If you want to collect feedback from the user for the Ad.

False = If you don't want to collect feedback from the user for the Ad.

12. Audio Enabled

Provides a medium to disable audio for video ad without user interaction.

Property name: IsAudioEnabled

Values:

True = If you want to play audio for video ad.

False = If you don't want to play audio for video ad.



User Details

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

```
PokktUserDetails pokktUserDetails = new PokktUserDetails();
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 Server Callback Params

Developer can set some values in POKKT SDK that they need to be sent to their server via POKKT Server callbacks.

```
Dictionary<string, string> extraParam = new Dictionary<string, string>();
extraParam.put("testdata", "{ \"adnetwork\": \"pokkt\" }");

PokktAds.SetCallbackExtraParams(extraParam);
```

Debugging

Other than enabling debugging for Pokkt SDK, it can also be used to:

1. Export log

Export your log to your desired location, we generally have it in root directory of SD card, if permission for external storage is provided and in cache folder otherwise.

```
PokktAds.Debugging.ExportLog();
```

2. Export log to cloud

You can also export log to cloud.

```
PokktAds.Debugging.ExportLogToCloud();
```



Analytics

We support various analytics in Pokkt SDK.

Below is mentioned how to enable various analytics with Pokkt SDK.

Google Analytics

Google analytics Id can be obtained from Google dashboard.

```
AnalyticsDetails analyticsDetail = new AnalyticsDetails();
analyticsDetail.SelectedAnalyticsType = AnalyticsType.GOOGLE_ANALYTICS;
analyticsDetail.GoogleAnalyticsId = "Google Analytics Id";
PokktAds.Analytics.SetAnalyticsDetails(analyticsDetail);
```

Flurry Analytics

Flurry application key can be obtained from Flurry dashboard.

```
AnalyticsDetails analyticsDetail = new AnalyticsDetails();
analyticsDetail.SelectedAnalyticsType = AnalyticsType.FLURRY;
analyticsDetail.FlurryApplicationKey = "Flurry Application Key";
PokktAds.Analytics.SetAnalyticsDetails(analyticsDetail);
```

MixPanel Analytics

MixPanel project token can be obtained from MixPanel dashboard.

```
AnalyticsDetails analyticsDetail = new AnalyticsDetails();
analyticsDetail.SelectedAnalyticsType = AnalyticsType.MIXPANEL;
analyticsDetail.MixPanelProjectToken = "MixPanel Project Token";
PokktAds.Analytics.SetAnalyticsDetails(analyticsDetail);
```

Fabric Analytics

Analytics Id is not required in case of Fabric.

```
AnalyticsDetails analyticsDetail = new AnalyticsDetails();
analyticsDetail.SelectedAnalyticsType = AnalyticsType.FABRIC;
analyticsDetail.FabricAnalyticsId = "Fabric Analytics Id";
PokktAds.Analytics.SetAnalyticsDetails(analyticsDetail);
```

IAP(In App Purchase)

Call trackIAP to send any In App purchase information to Pokkt.

```
InAppPurchaseDetails inAppPurchaseDetails = new InAppPurchaseDetails();
inAppPurchaseDetails.ProductId = "<productId>";
inAppPurchaseDetails.PurchaseData = "<purchaseData>";
inAppPurchaseDetails.PurchaseSignature = "<purchaseSignature>";
inAppPurchaseDetails.PurchaseStore = IAPStoreType.GOOGLE;
inAppPurchaseDetails.Price = <100.00>;

PokktAds.Analytics.trackIAP(inAppPurchaseDetails);
```



Project Configuration

In the package downloaded above you will find:

1. Docs:
Contains documentations for step wise step integration for SDK.
2. PokktAdsDemo:
Source code for *PokktAds Demo*(Sample app) which showcase implementation of Pokkt SDK through code for better understanding.
3. Extensions:
PokktExtension.dll, *PokktExtension.Droid.dll* and *PokktExtension.iOS.dll* both of these are required by Pokkt SDK and should be added to project

Android

minSdkVersion supported is **11**.

Dependencies

- Add PokktExtension.dll and PokktExtension.Droid.dll with analytics to your project as “Reference”.
- Add *support-v4.jar*
- We expect **Google play services** integrated in project, although it;s optional but we recommend you to integrate it, as it’s required to fetch **AdvertisingID** for device,which is useful to deliver targeted advertising to Android users.

Manifest

Permissions Declarations

Add the following permissions to your project manifest

1. Mandatory permissions.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

- android.permission.INTERNET = Required for SDK communication with server.
- android.permission.ACCESS_NETWORK_STATE = Required to detect changes in network, like if WIFI is available or not.

2. Optional permissions.

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.SEND_SMS" />
```



```
<uses-permission android:name="android.permission.VIBRATE" />
```

- android.permission.WAKE_LOCK = Required to prevent device from going into the sleep mode during video play.
- android.permission.WRITE_EXTERNAL_STORAGE = Required to store media files related to ads in external SD card, if not provided we will use app cache folder to store media files, which will result in unnecessary increase in application's size. It is recommended to ask for this permission as low end devices generally have less internal memory available.
- android.permission.WRITE_CALENDAR = Some Ads create events in calendar.
- android.permission.ACCESS_FINE_LOCATION = Some Ads show content based on user's location
- android.permission.CALL_PHONE = Some Ads are interactive and they provide you a way to call directly from the content.
- android.permission.SEND_SMS = Some Ads are interactive and they provide you a way to send message.
- android.permission.VIBRATE = Some Ads provide haptic feedback, so as to maintain their behavior we need this permission

Activity Declaration

Add the following activity in your AndroidManifest for Pokkt SDK integration.

```
<activity
    android:name="com.pokkt.sdk.PokktAdActivity"
    android:configChanges="keyboard|keyboardHidden|navigation|
                           orientation|screenLayout|uiMode|
                           screenSize|smallestScreenSize"
    android:hardwareAccelerated="true"
    android:label="Pokkt"
    android:screenOrientation="landscape"
    android:windowSoftInputMode="stateAlwaysHidden|adjustUnspecified" />
```

You can change the **android:screenOrientation="landscape"** to of your choice, the way you want to display the ads.

Service Declaration

Add the following service in your AndroidManifest for receiving InApp notifications. How to set up InApp notifications see "Pokkt Dashboard" document.

```
<service
    android:name="com.pokkt.sdk.notification.NotificationService"
    android:exported="false"
    android:label="PokktNotificationService" />
```

iOS

Dependencies

- Add PokktExtension.dll and PokktExtension.iOS.dll with analytics to your project as "Reference".
- If you are using **Visual Studio Code**
Navigate to "Select iOS project -> Project Options -> Build -> iOS Build" and then enter following arguments in "Additional mtouch arguments" text box:



```
-gcc_flags "-ObjC -lxml2 -framework CoreMotion -framework CoreData -framework Foundation  
-framework MediaPlayer -framework SystemConfiguration -framework UIKit -framework  
CoreTelephony -framework EventKit -framework AdSupport -framework CoreGraphics -framework  
CoreMotion -framework MessageUI -framework EventKitUI -framework CoreLocation -framework  
AVFoundation" to Additional mtouch arguments
```

Framework

```
CoreData.framework  
Foundation.framework  
MediaPlayer.framework  
SystemConfiguration.framework  
UIKit.framework  
CoreTelephony.framework  
EventKit.framework  
AdSupport.framework  
CoreGraphics.framework  
CoreMotion.framework  
MessageUI.framework  
EventKitUI.framework  
CoreLocation.framework  
AVFoundation.framework  
Libc++.tbd  
libsqlite3.tbd
```

Info.plist

Add the below exceptions to your application info.plist.

```
<key>NSAppTransportSecurity</key>  
  <dict>  
    <key>NSExceptionDomains</key>  
    <dict>  
      <key>pokkt.com</key>  
      <dict>  
        <key>NSIncludesSubdomains</key>  
        <true/>  
        <key>NSExceptionAllowsInsecureHTTPLoads</key>  
        <true/>  
        <key>NSExceptionRequiresForwardSecrecy</key>  
        <false/>  
        <key>NSExceptionMinimumTLSVersion</key>  
        <string>TLSv1.2</string>  
        <key>NSThirdPartyExceptionAllowsInsecureHTTPLoads</key>  
        <false/>  
        <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>  
        <true/>  
        <key>NSThirdPartyExceptionMinimumTLSVersion</key>  
        <string>TLSv1.2</string>
```



```

        <key>NSRequiresCertificateTransparency</key>
        <false/>
    </dict>
    <key>cloudfront.net</key>
    <dict>
        <key>NSIncludesSubdomains</key>
        <true/>
        <key>NSExceptionAllowsInsecureHTTPLoads</key>
        <true/>
        <key>NSExceptionRequiresForwardSecrecy</key>
        <false/>
        <key>NSExceptionMinimumTLSVersion</key>
        <string>TLSv1.2</string>
        <key>NSThirdPartyExceptionAllowsInsecureHTTPLoads</key>
        <false/>
        <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>
        <true/>
        <key>NSThirdPartyExceptionMinimumTLSVersion</key>
        <string>TLSv1.2</string>
        <key>NSRequiresCertificateTransparency</key>
        <false/>
    </dict>
</dict>
</dict>

```