Introduction

This guide is intended for developers who are integrating the AdView iOS SDK. After you have integrated the SDK, you must choose the relevant ad format and follow the steps for implementing that format.

Prerequisites

- iOS 11.0+

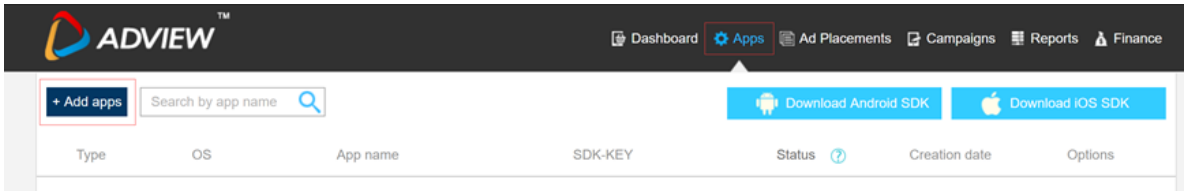
Step 1: Sign up and acquire AppId

Before integration, you need to sign up on **AdView Mobile Marketplace** first and acquire a specific AdView App id **SDK-KEY** for each app you would like to integrate with AdView SDK.

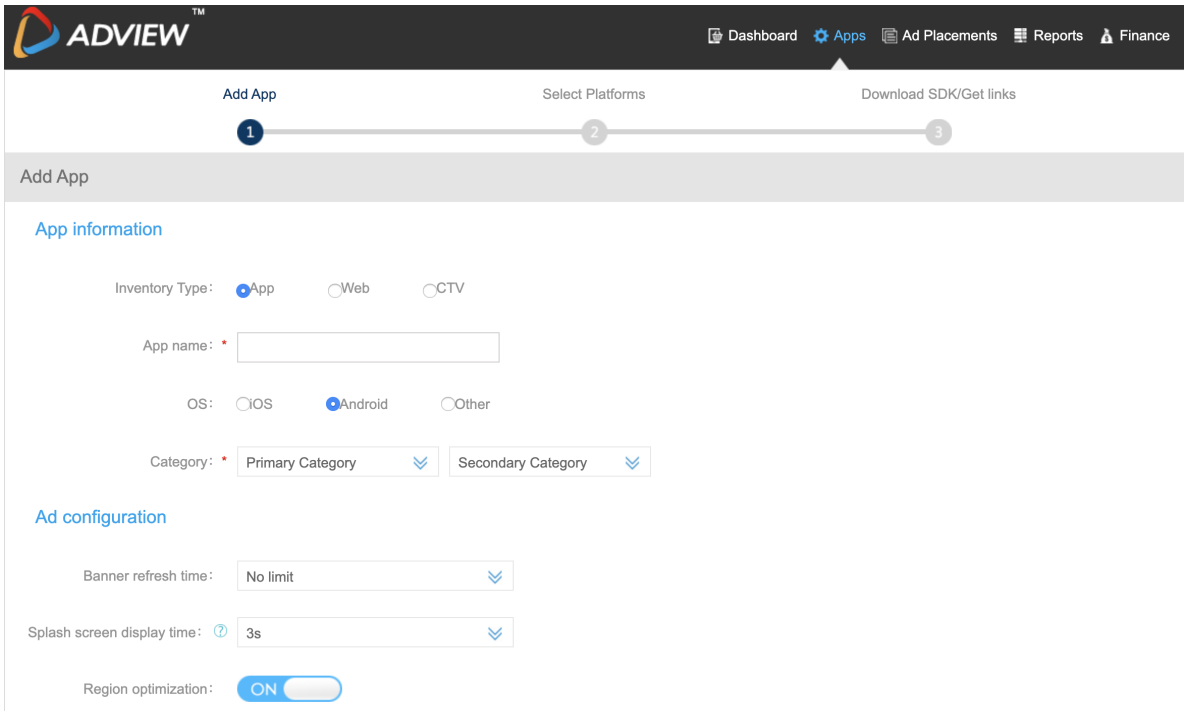
1. Sign up your publisher account at AdView Official Website:

https://www.adview.com/web/overseas/register

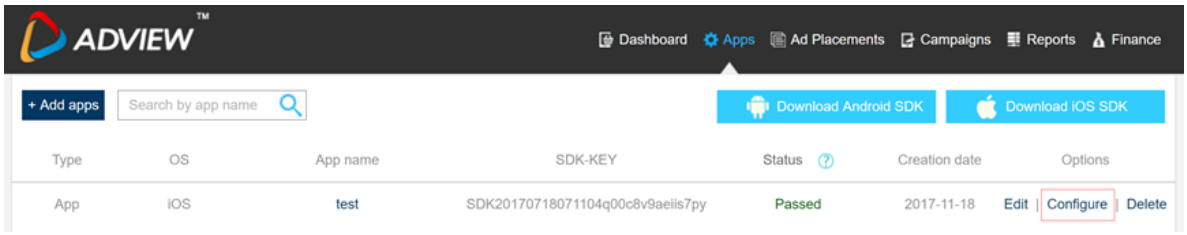
2. Sign in with your user name and password, AdView Publisher Dashboard opens
3. On the above menu bar, click **Apps**
4. On the left-up corner, click **+Add apps**



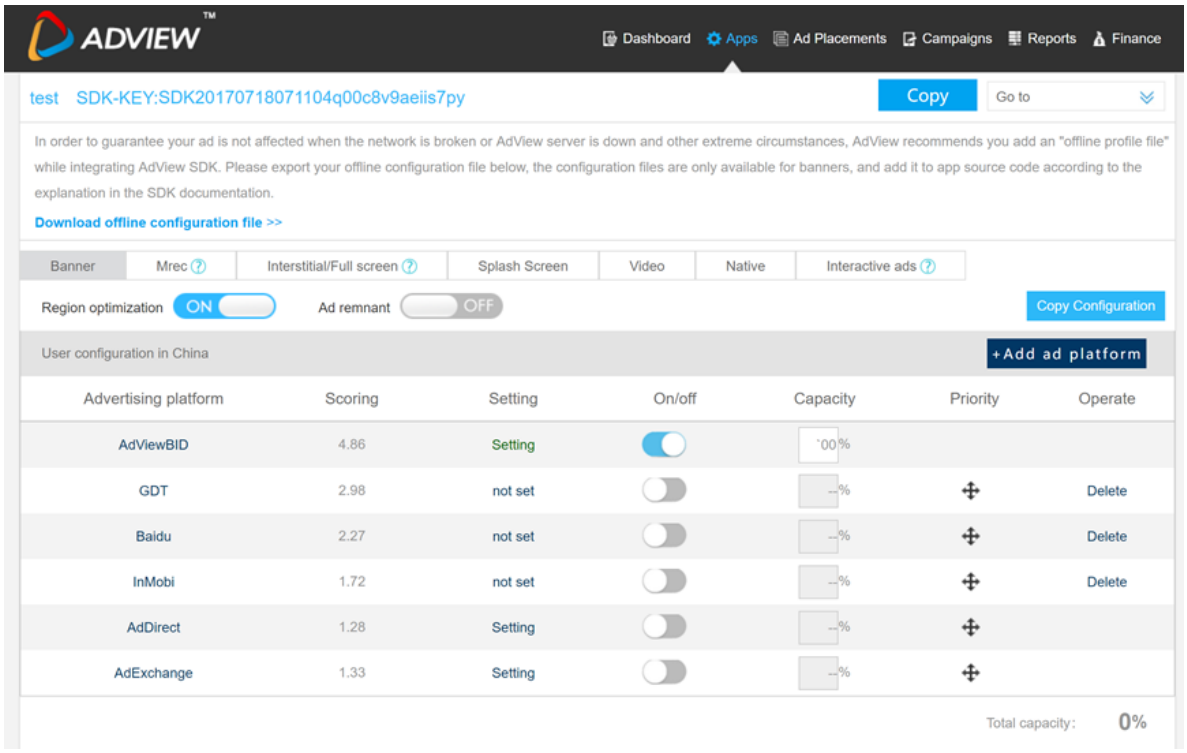
5. Fill in the app information step by step, please select the correct ad format, otherwise you might not get any ad fill. After filling all the necessary app information, please submit the app information for review. The review process generally takes one business day, please contact your account manager if any questions.



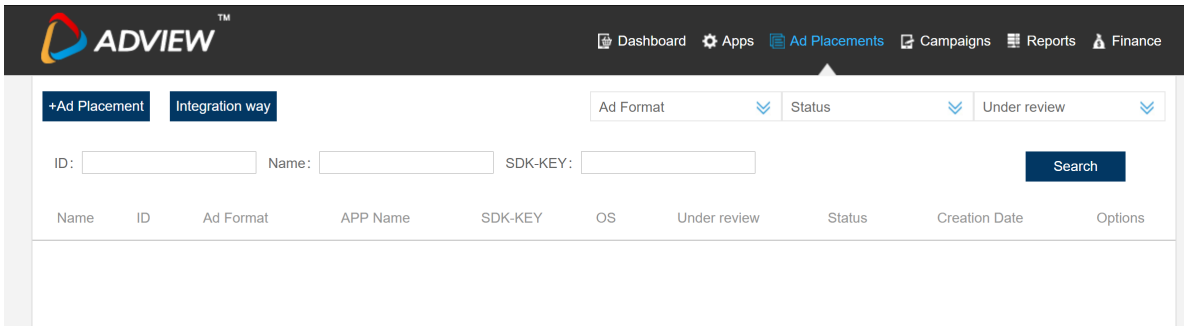
6. After the app has passed the review, the app **Status** will become **Passed** as below, please click **Configure** to config the ad network mediation configuration.



7. If you are using AdView's ad network mediation feature, please click **Add ad platform** to add the ad networks you would like to support. If you are already using other ad mediation solutions, please simply turn the Advertising platform **AdViewBID** on and set the capacity to 100%. You can also find the specific AdView app id **SDK-KEY** here, which you will need for the SDK integration.



8. If your app's ad format is Video or Native, you need to create **ad placement** for each ad place in your app. Please select **Ad Placement** on the above menu bar and click **+Ad Placement**, then add the necessary ad placement information accordingly.



Step 2: Integration

Import the AdViewSDK

Currently, we support **Manual download** and **CocoaPods** for SDK import. You can select the import mode as your need.

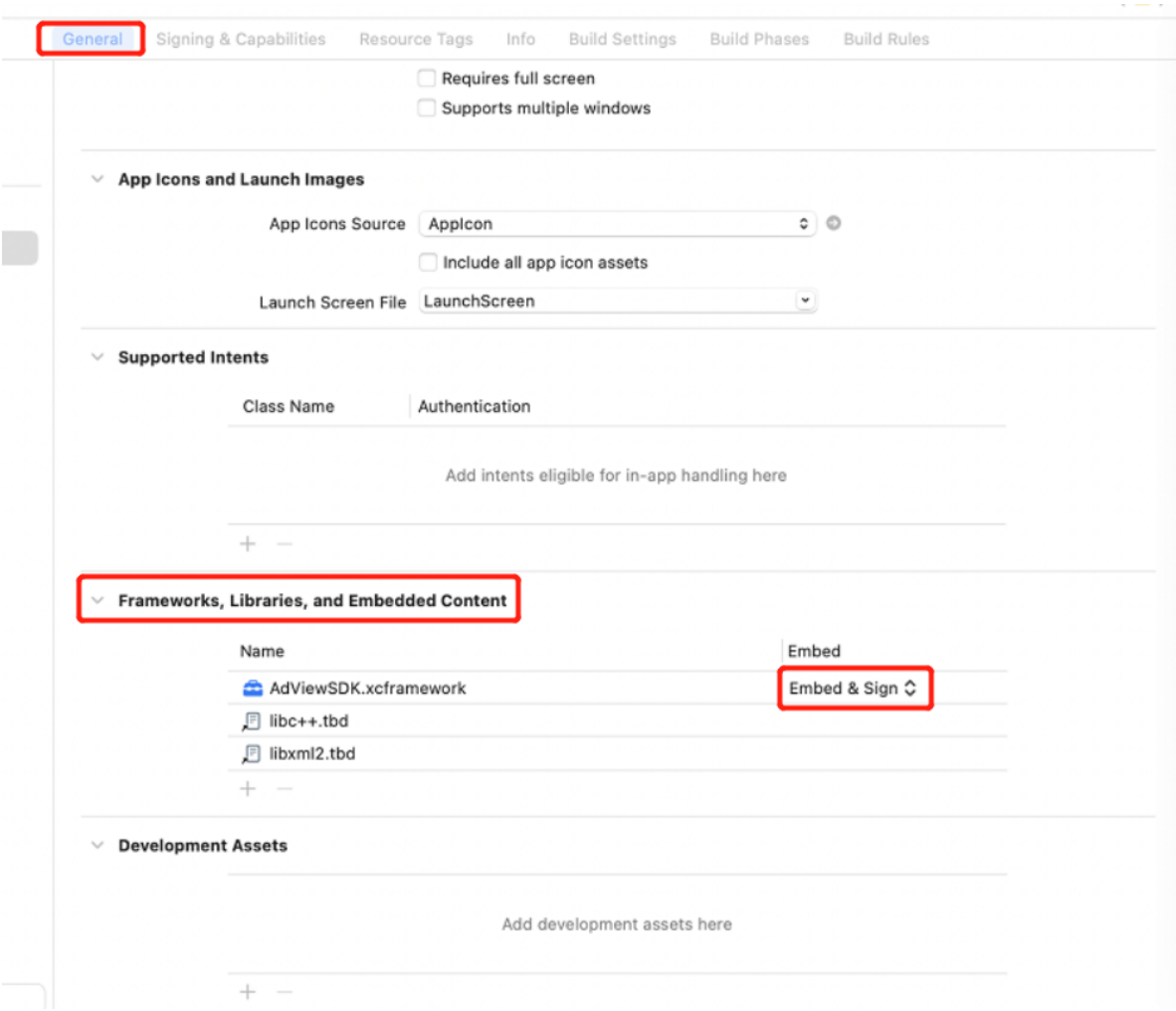
Manual SDK Download

1. Download the latest Version SDK here:

https://www.adview.com/web/productAndTech/sdk#download-sdk

2. Extract the zip file(Include: framework package, integration document, example)
3. Import the framework into your project

- Add the dependency in YourProjectTarget's `Build Phases` - `Link Binary With Libraries`.
- Add `AdViewSDK.xcframework` in `Build Phases` - `Embed Frameworks`.
- Set Embed type as Embed & Sign.




```
dependencies:
    libc++.tbd
    libxml2.tbd
```

4. Update Your Project Settings

- Declare the following permissions in your `Info.plist`:

```
App Transport Security Settings - Allow Arbitrary Loads - YES
```

- Declare the following Flags in your `Build Settings`:

```
Build Settings - Other Linkr Flags - "-Objc"
```

CocoaPods

The simplest way to import the SDK into an iOS project is to use CocoaPods. Open your project's Podfile and add this line to your app's target:

```
pod 'AdViewSDK'
```

Then from the command line run:

```
pod install --repo-update
```

If you're new to CocoaPods, see their official documentation for info on how to create and use Podfiles.

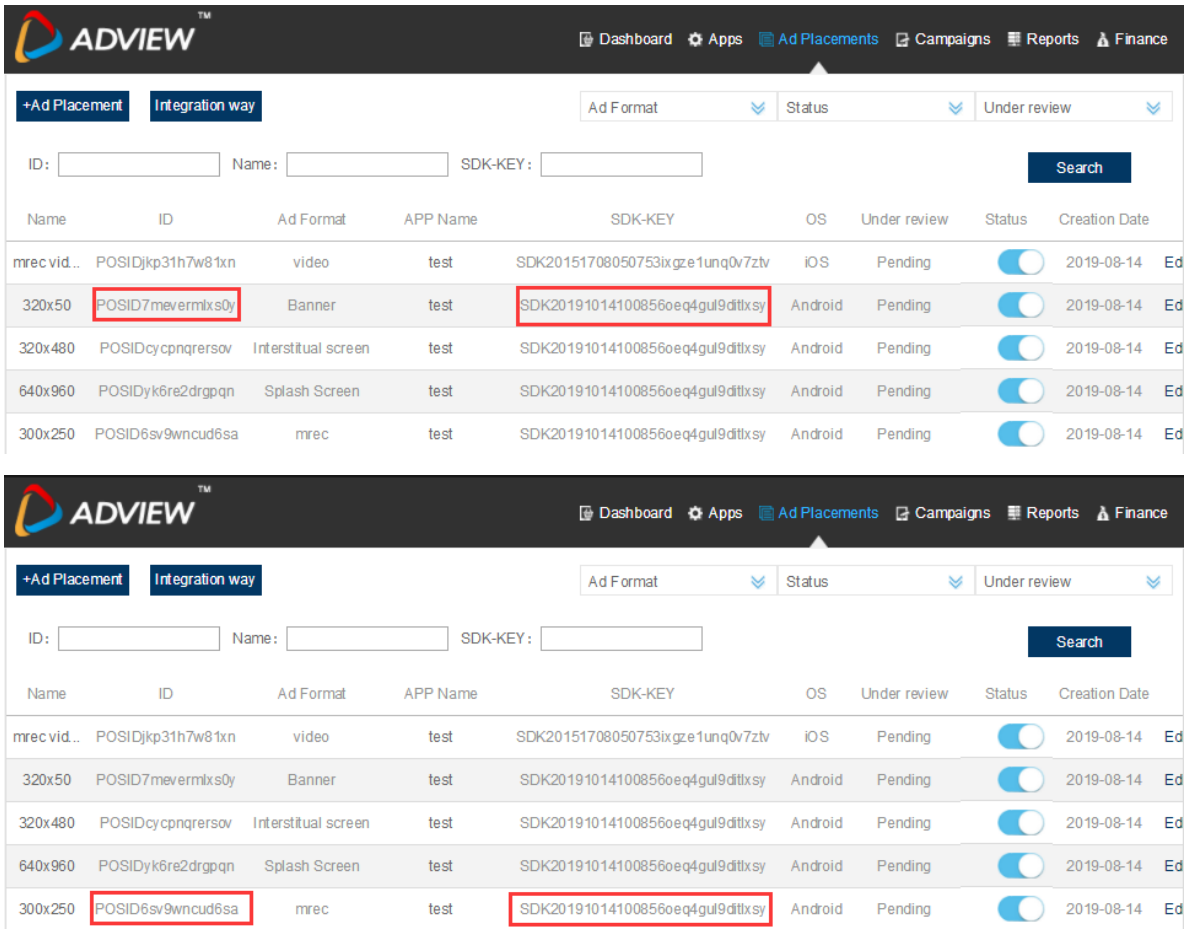
Step 3: Configure Ad Units in Your App

Banner

Banner ads usually appear at the top or bottom of your app's screen. Adding one to your app takes just a few lines of code.

Create an APP in the Publisher UI for instructions. You will need that Ad Placement ID and SDK-KEY.

Usually when you add an app, you have the option of automatically ad banner or mrec ad placement.



1. Init the Banner AdView

The AdView SDK provides a UIView subclass `AdViewView`, which handles requesting and loading ads. We provide the following methods for you to set up this subclass:

```
+ (AdViewView *)requestBannerSize:(AdViewBannerSize)size
                       bannerType:(AdViewBannerType)type
                       positionId:(NSString *)positionId
                         delegate:(id<AdViewViewDelegate>)delegate;
- (NSString *)appId;
```

```
self.banner = [AdViewView requestBannerSize:AdViewBannerSize_320x50
                                 bannerType:AdViewBannerType_normal
                                 positionId:@"POSID6uooxjk893j0"
                                   delegate:self];
[self.view addSubview:self.banner];
```

Only MREC mode banner need `Position ID`.

```
self.MREC = [AdViewView requestBannerSize:AdViewBannerSize_300x250
                                bannerType:AdViewBannerType_video
                                positionId:@"POSID4zt29dh4zbp4"
                                  delegate:self];
[self.view addSubview:self.MREC];

- (NSString *)appId {
    return @"SDK20191028101142elckebuvuy5lczp";
}
```

return the `SDK-KEY` in protocol function `- (NSString *)appId;`

Choose the ad size you need, we recommend 728x90 for tablet traffic and 320x50 for mobile traffic. The default size of MREC is 300x250, more define as following, all definitions are in `AdViewDefinesPublic.h` note the corresponding value. In addition, MREC currently supports the display of video advertisements, which will be described in detail below.

```
typedef NS_ENUM(NSInteger, AdViewBannerSize) {
    AdViewBannerSize_320x50,
    AdViewBannerSize_480x44,
    AdViewBannerSize_300x250,   //MREC
    AdViewBannerSize_480x60,
    AdViewBannerSize_728x90,
};
```

And the Banner type:

```
typedef NS_ENUM(NSInteger, AdViewBannerType) {
    AdViewBannerType_normal,    //Html5、Image
    AdViewBannerType_video      //Video
};
```

2. Should not change the Ad's bouns,because it may cause some error.But you can change the origin in the visible area.
3. Using the Delegate *AdViewViewDelegate.h*

which includes a variety of optional callbacks you can use to be notified of events.For example, when an ad has successfully loaded, or when the ad is about to present or dismiss a modal view. `AdViewDelegate` is a interface includes the following notifications:

```
@protocol AdViewViewDelegate <AdViewGDPRProcotol>
@required
- (NSString *)appId;
- (UIViewController *)viewControllerForShowModal;
@optional
- (void)didReceivedAd:(AdViewView *)adView;
- (void)didFailToReceiveAd:(AdViewView *)adView Error:(NSError*)error;
- (void)adViewDidDisplayAd:(AdViewView *)adView ;
- (void)adViewdidFailToDisplayAd:(AdViewView *)adView Error:(NSError*)error;
- (void)adViewWillPresentScreen:(AdViewView *)adView;
- (void)adViewDidDismissScreen:(AdViewView *)adView;
- (void)adViewDidFinishShow:(AdViewView *)adview;
- (int)autoRefreshInterval;
- (BOOL)logMode;
...
@end
```

4. Refresh time set

You can use `- (int)autoRefreshInterval` to return the automatic refresh time of banner. The minimum refresh time allowed by AdView is 30s, so when you set the refresh time between 1 and 30 seconds, the default refresh time is 30 seconds.

```
- (int)autoRefreshInterval {
    return 30;
}
```

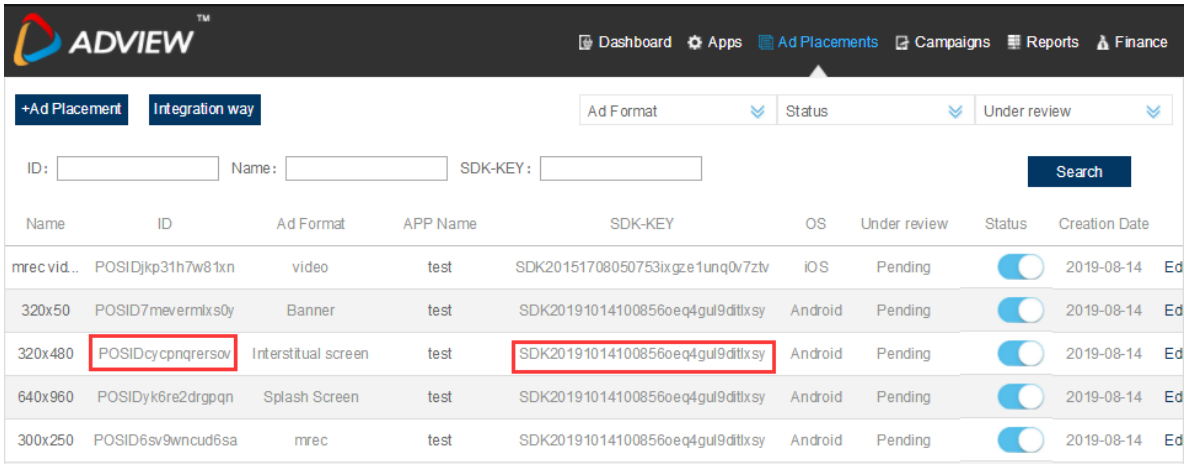
If you set the refresh time to 0, which means the banner will not automatically refresh. The ads will **expire in 10 minutes**. So please notice the ad's life cycle,release old ad then reinitialize the new banner.

Interstitial

Interstitial ads provide full-screen experiences, commonly incorporating rich media to offer a higher level of interactivity compared to banner ads. Interstitials are typically shown during natural transitions in your app; for example, after completing a game level, or while waiting for a new view to load.

Create an APP in the Publisher UI for instructions. You will need that SDK-KEY.

Usually when you add an app, you have the option of automatically interstitial ad placement.



1. Create an Interstitial Ad

Income the `delegate`、`SDK-KEY` and `viewControllerForShowModal` complete interstitial initialization:

```
+ (AdViewView *)requestAdInterstitialWithPositionId:(NSString *)positionId
                                           delegate:(id<AdViewViewDelegate>)delegate;
- (NSString *)appId;
```

```
self.interstitial = [AdViewView
requestAdInterstitialWithPositionId:@"POSIDajpz3chwo1q2"
                                                                  delegate:self];

- (NSString *)appId {
    return @"SDK20191028101142elckebuvuy5lczp";
}

- (UIViewController *)viewControllerForShowModal {
    return ViewController;
}
```

2. Display an Interstitial Ad

When the advertisement is ready, you will receive the following callbacks:

```
- (void)didReceivedAd:(AdViewView *)adView
```

Next，Call the following methods to serve interstitial ads to your users:

```
- (BOOL)showInterstitialWithRootViewController:(UIViewController
*)rootViewController
```

Of course, you can also become a step , in *-(void)didReceivedAd:*, it means ad is ready and can be shown as you need, you can call the show method, then ad will pop up to screen:

```
- (void)didReceivedAd:(AdViewView *)adView {
    [adView showInterstitialWithRootViewController:[UIApplication
sharedApplication].keyWindow.rootViewController];
}
```

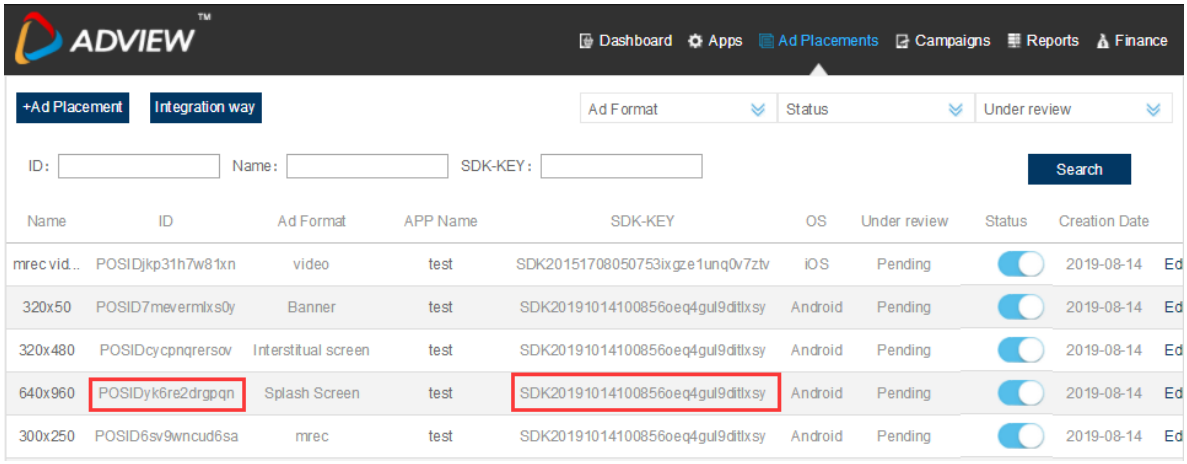
3. Using the Delegate

`AdViewViewDelegate.h` provides a listener interface, which includes a variety of optional callbacks you can use to be notified of events; for example, when an ad has successfully loaded, or when the ad is about to present or dismiss a modal view. The same as Banner.

Spread

Spread is usually an advertisement displayed at the start of app and usually has higher visibility than banner and insterstitial.

Create an APP in the Publisher UI for instructions. You will need SDK-KEY.



1. Create an spread Ad

Income the following parameters and complete Spread initialization:

```
+ (AdViewView *)requestSpreadActivityWithPositionId:(NSString *)positionId
                                           delegate:
(id<AdViewViewDelegate>)delegate;
- (NSString *)appId;
- (UIViewController *)viewControllerForShowModal;
```

You should write these code in the `AppDelegate.m`

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions {
    if (@available(iOS 13.0, *)) {

    } else {
        self.window = [[UIWindow alloc] initWithFrame:[UIScreen
mainScreen].bounds];
        [self.window setBackgroundColor:[UIColor whiteColor]];

        ViewController * vc = [[ViewController alloc] init];
        UINavigationController * navc = [[UINavigationController alloc]
initWithRootViewController:vc];
        [self.window setRootViewController:navc];
        [self.window makeKeyAndVisible];
    }
    [AdViewView requestSpreadActivityWithDelegate:self];
    return YES;
}
```

2. Display spread Ad

Usually spread will show automatically.

3. Using the Delegate

also, `AdViewViewDelegate.h` provides a listener interface, which includes a variety of optional callbacks you can use to be notified of events.

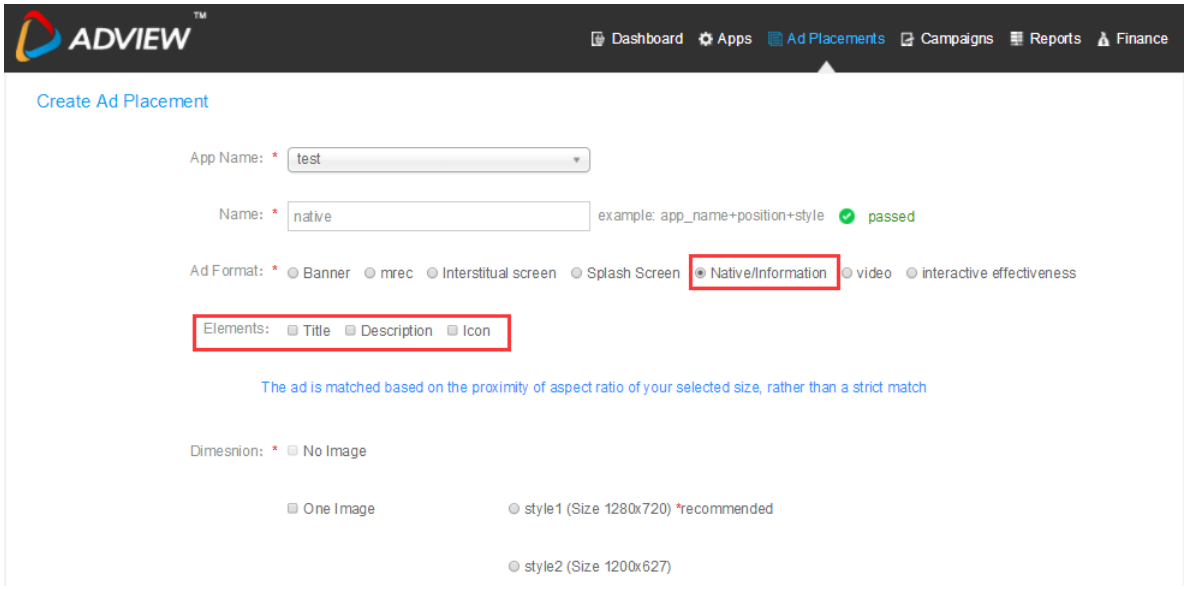
```
- (NSString *)appId;
- (UIViewController *)viewControllerForShowModal;
- (NSString *)logoImgName;
```

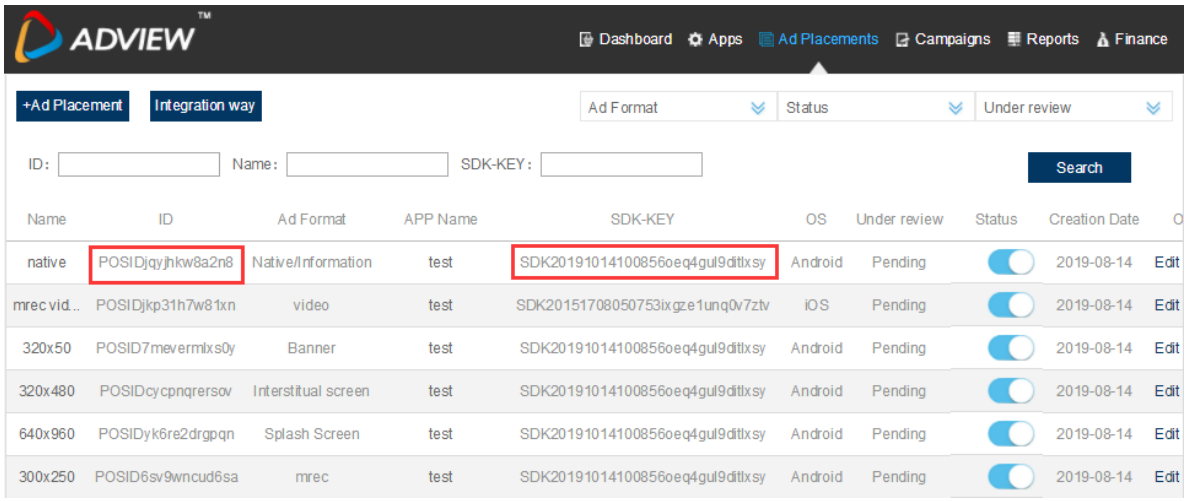
Native

Native ads make it easy for you to monetize your app in a way that's consistent with its existing design. The AdView SDK gives you access to an ad's individual assets so you can design the ad layout to be consistent with the look and feel of your app.

Init the Native Ad You will need that Ad Position ID and SDK-KEY.

You need to choose the right creative and image size when creating native ad placement.





1. Create an AdView Native Ad:

```
- (instancetype)initWithAppKey:(NSString *)appkey positionID:(NSString
*)positionID;
```

```
    self.native = [[AdViewNativeAd alloc] initWithAppKey:ADVIEW_APPID
positionID:ADVIEW_NATIVE_POSITION_ID];
    self.native.delegate = self;
    self.native.controller = self;
```

2. Call `-(void)loadNativeAdWithCount:` to request an ad from the server and download associated images asynchronously.

```
    [self.native loadNativeAdWithCount:1];
```

3. Display native Ad, after `- (void)adViewNativeAdSuccessToLoadAd:NativeData:` callback is called when native ad is downloaded from server.
   Upon failure `- (void)adViewNativeAdFailToLoadAd:WithError:` callback is called with a corresponding error code message.

```
- (void)adViewNativeAdSuccessToLoadAd:(AdViewNativeAd *)adViewNativeAd
NativeData:(NSArray <AdViewNativeData *>*)nativeDataArray {
    //this is contentView, see demo
    self.nativeAdView = [[UIView alloc] init];

    //consuct the custom ad view

    //must use this function to send display
    [self.native showNativeAdWithData:nativeDataArray.firstObject
             friendlyObstructionArray:friendlyObstructionArray
                               onView:self.nativeAdView];
}
```

according to **AdViewNativeData.adProperties** is a list contains items as following:

Normal native mode:

Figure native4.1

Name	Value Type	Description
adIcon	String	icon url for adview icon
adLogo	String	logo url for adview logo
desc	String	description
title	String	ad title string
imageList	Array	large images for ad resource
iconUrlString	String	icon for ad resource
pimage	String	opt-out icon
pclick	String	opt-out click url
ompara、omurl、omvendor	String	OMSDK

5. Using the Delegate

```
@protocol AdViewNativeAdDelegate <AdViewGDPRProcotol>
- (void)adViewNativeAdSuccessToLoadAd:(AdViewNativeAd *)adViewNativeAd
NativeData:(NSArray <AdViewNativeData *>*)nativeDataArray;
- (void)adViewNativeAdFailToLoadAd:(AdViewNativeAd *)adViewNativeAd WithError:
(NSError *)error;
- (void)adViewNativeAdWillShowPresent;
- (void)adViewNativeAdClosed;
- (void)adViewNativeAdResignActive;
@end
```

Opt-out

In line with new industry self-regulatory guidance provided by the DAA, AdView now requires all native ads to display the privacy information icon and click region. The icon must be at least 20x20px and can be placed in any one of the four corners of the ad.
You can use your own icon and privacy information page, or use the corresponding icons and privacy information pages provided by AdView, You can get them with **pimage** and **pclick**, see Figure native4.1.

demo codes for got privacy image :

```
    NSString *optOutIconURL = data.adProperties[@"pimage"];
    UIImageView *optOutIcon = [[UIImageView alloc] initWithImage:[UIImage
imageWithData:[NSData dataWithContentsOfURL:[NSURL
URLWithString:optOutIconURL]]]];
    optOutIcon.userInteractionEnabled = YES;
    UITapGestureRecognizer *optOutIconTap = [[UITapGestureRecognizer alloc]
initWithTarget:self action:@selector(optOutIconClick:)];
    [optOutIcon addGestureRecognizer:optOutIconTap];
```

demo codes for privacy click URL, you can got click url with `NSString *optOutCilickURL = data.adProperties[@"pclick"];` and show it by yourself.

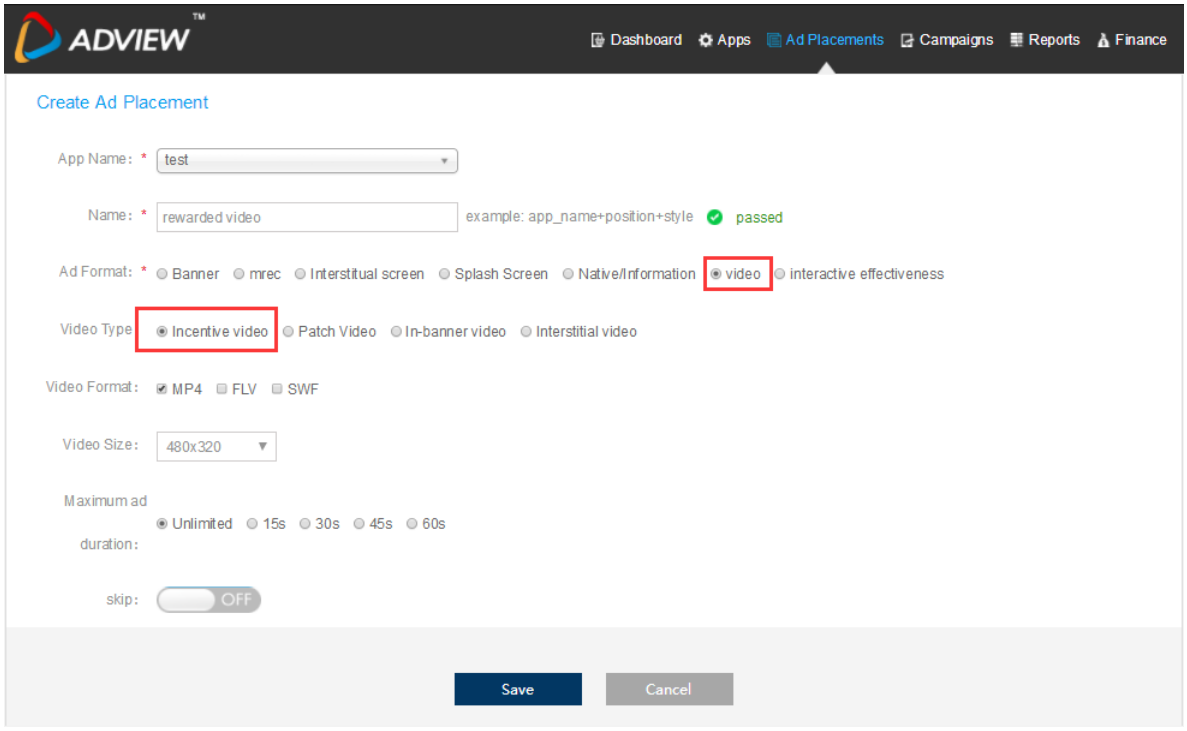
```
//display privacy information
- (void)optOutIconClick:(id)sender {
    [[UIApplication sharedApplication] openURL:[NSURL
URLWithString:optOutCilickURL]];
}
```

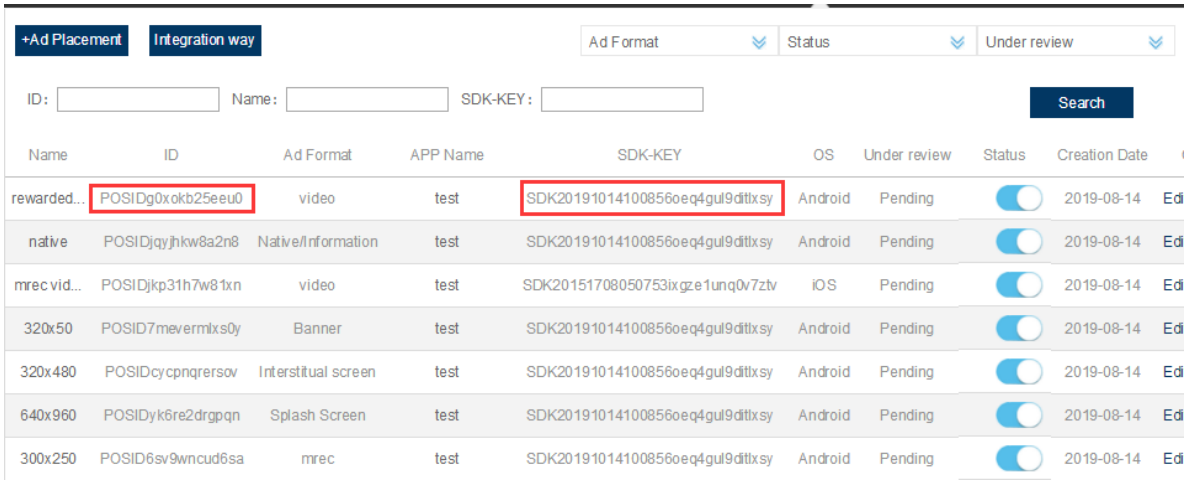
Video

Video Ads are shown in fullscreen mode and can be placed before, in-between, or after the app content.

Rewarded & Interstitial video

Create an APP in the Publisher UI for instructions. You will need that Ad Position ID and SDK-KEY.





1. Create an Rewarded video Ad:

Income the following parameters and complete initialization:

```
    self.video = [AdViewVideo playVideoWithAppId:ADVIEW_APPID
                                      positionId:ADVIEW_VIDEO_POSITION_ID
                                       videoType:AdViewVideoTypeInstl
                                        delegate:self];
    [self.video setInterfaceOrientations:[UIApplication
sharedApplication].statusBarOrientation];
    [self.video getVideoAD];
```

callback **AdViewVideoDelegate** is call capture events from video SDK process.

```
@protocol AdViewVideoDelegate <AdViewGDPRProcotol>
- (void)adViewVideoIsReadyToPlay:(AdViewVideo *)video;
- (void)adViewVideoPlayStarted;
- (void)adViewVideoPlayEnded;
- (void)adViewVideoClosed;
- (void)adViewVideoClicked;
- (void)adViewVideoDidReceiveAd:(NSString *)vastString;
- (void)adViewVideoFailReceiveDataWithError:(NSError*)error;
```

AdView provides a response function `- (void)adViewVideoIsReadyToPlay:` to let you know if the advertisement was successfully requested:

This function is called when the ad request is successful:

```
- (void)adViewVideoIsReadyToPlay:(AdViewVideo *)video
```

This function is called when an ad request fails:

```
- (void)adViewVideoFailReceiveDataWithError:(NSError *)error {
    NSLog(@"%@",error);
}
```

3. Called when the video ad is ready, after which video shows can be made:

```
- (void)adViewVideoIsReadyToPlay:(AdViewVideo *)video
```

Next， Call the following methods to serve reward ads to your users

```
- (BOOL)showVideoWithController:(UIViewController *)controller
```

Of course, you can also become a step , in *- (void)adViewVideoIsReadyToPlay:* , it means ad is ready and can be shown as you need, you can call the show method, then ad will pop up to screen

```
- (void)adViewVideoIsReadyToPlay:(AdViewVideo *)video {
  [video showVideoWithController:self];
}
```

4. You can also monitor the following play status to determine rewards for users:

Called when video ads start playing

```
- (void)adViewVideoPlayStarted;
```

Called when video ads have ended

```
- (void)adViewVideoPlayEnded;
```

Called when the video ad is closed:

```
- (void)adViewVideoClosed;
```

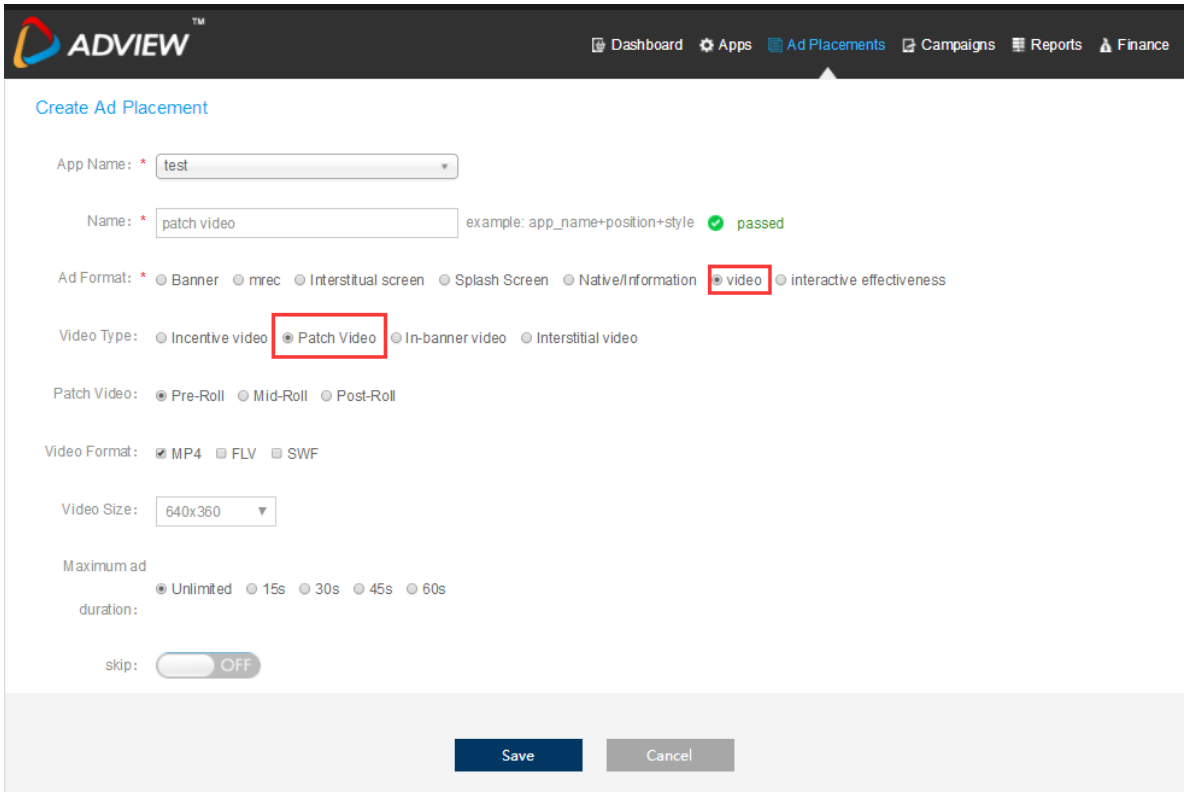
Called when the video ad is playing incorrectly:

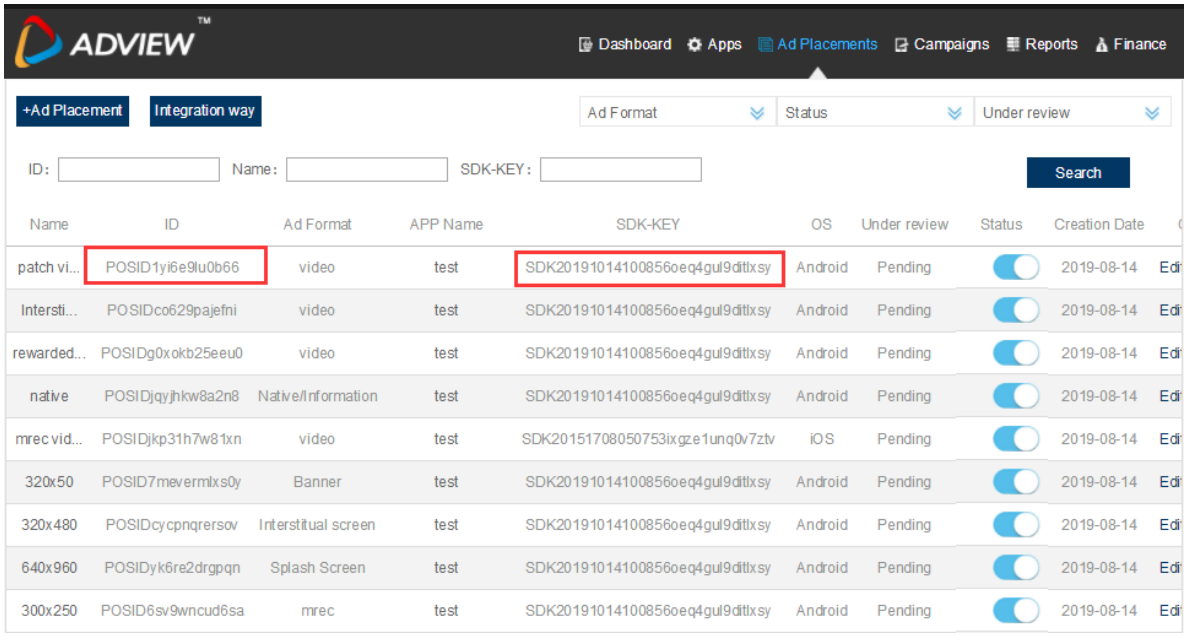
```
- (void)adViewVideoFailReceiveDataWithError:(NSError*)error;
```

In-stream Video

In-stream Video can provide video source (means VAST xml content) to app , and SDK will not show it . though app can parse vast itself and play it as it wish.

Create an APP in the Publisher UI for instructions. You will need that Ad Position ID and SDK-KEY.





1. Create an paste video ad:

```
    self.video = [AdViewVideo playVideoWithAppId:ADVIEW_APPID
                                      positionId:ADVIEW_VIDEO_POSITION_ID
                                       videoType:AdViewVideoTypePreMovie
                                        delegate:self];
```

2. Response for video ad received:
   with paste video mode, after ad video is received, the video contents will be pass through to app with callback *onReceivedVideo()*, in parameter string:

```
- (void)adViewVideoDidReceiveAd:(NSString *)vastString {
    NSLog(@"%@",vastString);
}
```

3. Play & other process:

Can follow the same steps as the reward video.

GDPR

As a publisher, you should integrate a Consent Management Platform (CMP) and request for vendor and purpose consents as outlined in IAB Europe's Mobile In-App CMP API v1.0: Transparency & Consent Framework. You can find a reference implementation of a web-based CMP and the corresponding native wrappers here in the IAB's GDPR Transparency and Consent Framework.

if you already got IABConsent_ConsentString, you just need pass through to SDK construction method as the last parameter.See `AdViewViewDelegate.h` - `AdViewGDPRProcotol` :

```
@protocol AdViewGDPRProcotol <NSObject>
- (BOOL)CMPPresent;
- (BOOL)subjectToGDPR;
- (NSString *)userConsentString;
- (NSString *)parsedPurposeConsents;
- (NSString *)parsedVendorConsents;
@end
```

or can also save examples directly in NSUserDefaults

if **String gdpr** is empty, it means not support GDPR.

CCPA

The California Consumer Privacy Act (CCPA) was created to provide California consumers with greater transparency and control over their personal information. In many ways, the CCPA is a first of its kind regulation in the United States that seeks to create broad privacy and data protection rules that apply to all industries doing business in the jurisdiction of California, rather than focusing on a single sector or specific data collection and use practices.

For Publishers with California-Based Users

As a publisher, you need to make sure to request consent from California-based users (to give or refuse consent / to opt-out or opt-in) about private data transfer. This answer should be saved in `NSUserDefaults` with key `IABUSPrivacy_String` in the US Privacy String format (CCPA Opt-Out Storage Format).

Sample of US Privacy String Saving in NSUserDefaults

```
- (void)saveCCPAUsPrivacyString:(NSString *)usString {
    if (usString.length > 0) {
        [NSUserDefaults.standardUserDefaults setObject:usString
                                                forKey:@"IABUSPrivacy_String"];
    }
}
```

AdView SDK reads that value by key "IABUSPrivacy_String" if it exists and uses it as an optional parameter for all ad requests.