

# LevelPlay AdView Adapter Integration Guide (iOS)

Adapter Version: 4.3.9

## Prerequisites

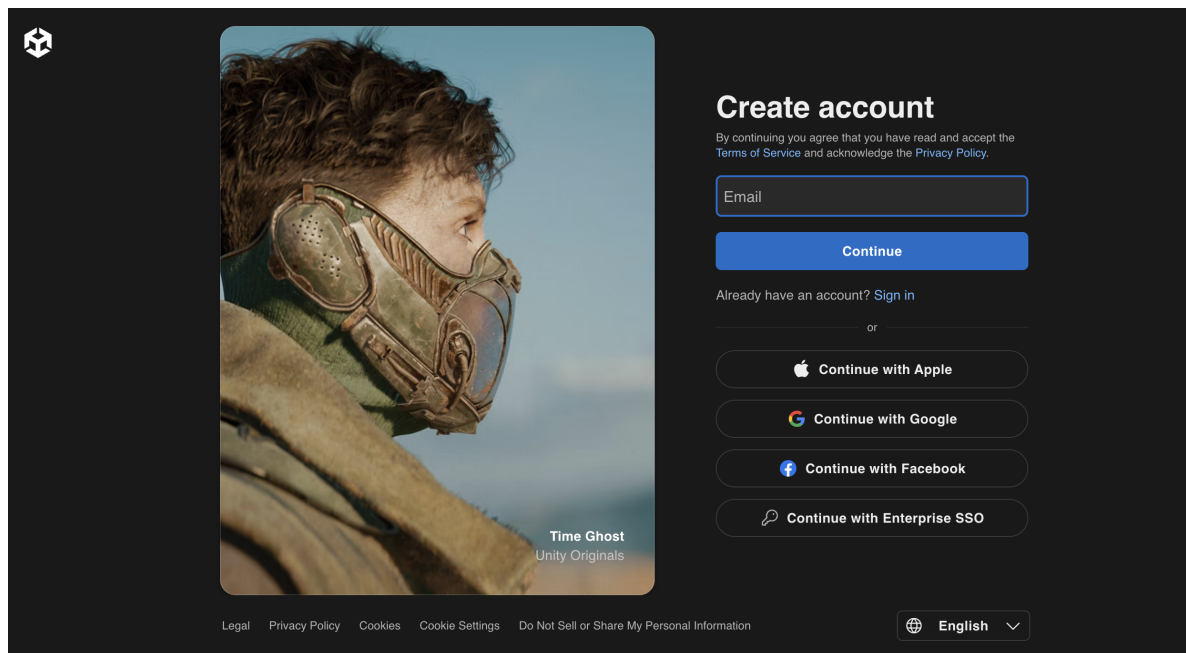
- **iOS Deployment Target:** Defined per network. ironSource (LevelPlay) mediation and AdView support **iOS 13.0 and later**.
- **Xcode:** 16.0 or higher.
- **LevelPlay SDK:** Already integrated in your app with at least one active ad unit.

If you haven't integrated the LevelPlay SDK yet, please follow the official guides first:

- [LevelPlay iOS Integration Guide](#)
- [Custom Network Adapter Overview](#)

## Step 1: Add AdView as a Custom Network in LevelPlay Dashboard

1. Log in to your **LevelPlay (ironSource)** account.



2. Create or select your app.

**Apps** Add app

Visibility: Shown Status Platform

Important! to receive revenue, you must first enter your [payment preferences](#) and [company info](#)

Your account is pending approval. We'll notify you by email when your account is approved.

Displaying 1 out of 1 apps Search by app

App name ↑	Status	Active ad units	S2S reward callback
多进制计算器 24367061d	Temp	4	...

### 3.Navigate to **Monetize** → **Networks** → **+ Add Custom Network.**]

**Ad units** Create ad unit

Visibility: Shown Status Ad format

Your account is pending approval. We'll notify you by email when your account is approved.

To create multiple ad units for interstitial, rewarded and banner ads, use the new SDK APIs. [Learn more](#)

Displaying 3 out of 3 ad units Search by ad unit

Status ↑	Ad unit	Ad format	Networks	Mediation groups
	interstitialforios 0zhwlnzz5afixio0	Interstitial	2	1 ...
	rewardedforios k6g51oth7pqxt303	Rewarded	2	1 ...
	bannerforios kyzvijslty2tu03r	Banner	2	1 ...

**Manage ad networks**

Your account is pending approval. We'll notify you by email when your account is approved.

Displaying 29 out of 29 ad networks Search by network

Network	Supported ad formats	Setup
smaato		<a href="#">Setup</a>
SUPER		<a href="#">Setup</a>
Tapjoy		<a href="#">Setup</a>
Tencent		<a href="#">Setup</a>
verve		<a href="#">Setup</a>
VK Ad Network		<a href="#">Setup</a>
Yandex		<a href="#">Setup</a>
<a href="#">+ Add custom network</a>		

### 4.Enter the following **Network Key** for AdView:

15bed2985

## ← Ad network setup

Custom network settings

Network key \*

Network key

Confirm key

Cancel

Save

5.Fill in the required credentials (provided by your AdView Account Manager):

- **userId**
- **accessKey**

## ← Ad network setup

AdView account settings

userId \*

userId

accesskey \*

accesskey

Reported Revenue \*

☒ Rate based revenue

Revenue will be reported based on the rate you set

☐ Reporting API

Revenue will be reported based on the network's reporting API

Cancel

Save

6.Revenue reporting: Currently supports **Rate-based revenue only**.

7.Save the network.

Unity LevelPlay

management

Mediation

Segments

A/B

Setup

Ad units

Instances

Networks

Placements

Direct deals

Test devices

Offerwall

Ad Quality

Review

Analysis

Creatives

User journey

Notifications

Manage

Rules

多进制计算器

Instances

Import

Export

Your account is pending approval. We'll notify you by email when your account is approved.

Displaying 2 out of 2 ad networks

Search by network

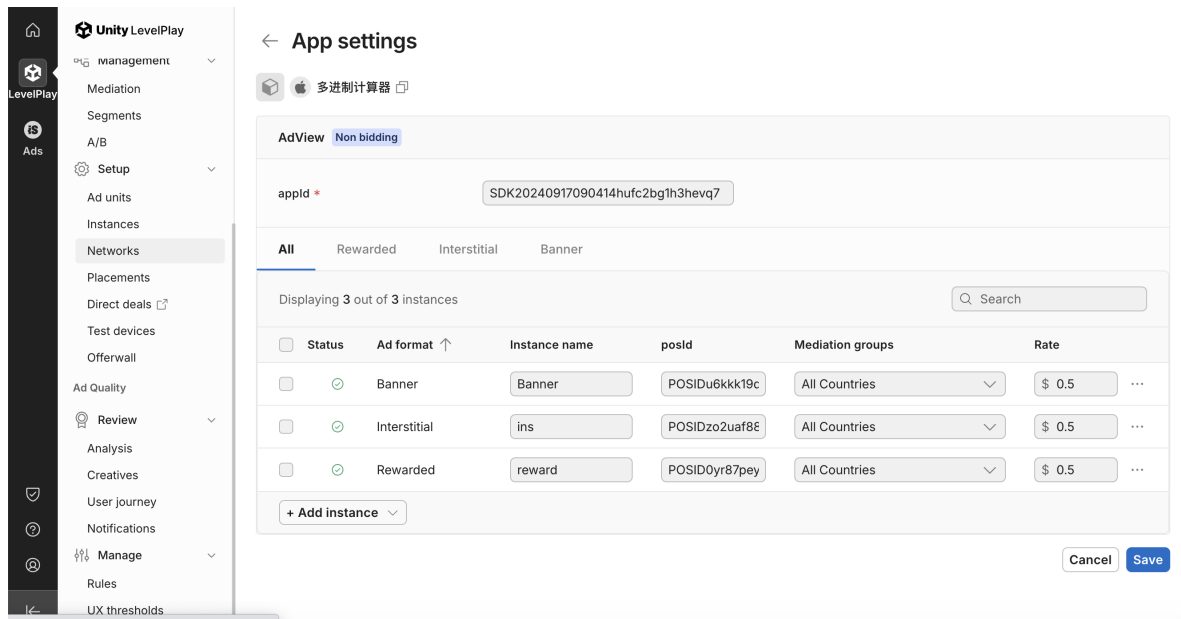
Bidding (1)	Rewarded	Interstitial	Banner	Native
ironSource	✓	✓	✓	
Custom (1)	Rewarded	Interstitial	Banner	Native
AdView	✓	✓	✓	N/A

## Instance Configuration

After adding the network:

- Go to **Networks**, select AdView.
- Enable the toggle.

- Enter your **AdView APPID (SDK-Key)** and **Placement ID (posId)** for each ad unit.
- Set the desired **CPM floor** and click **Save**.



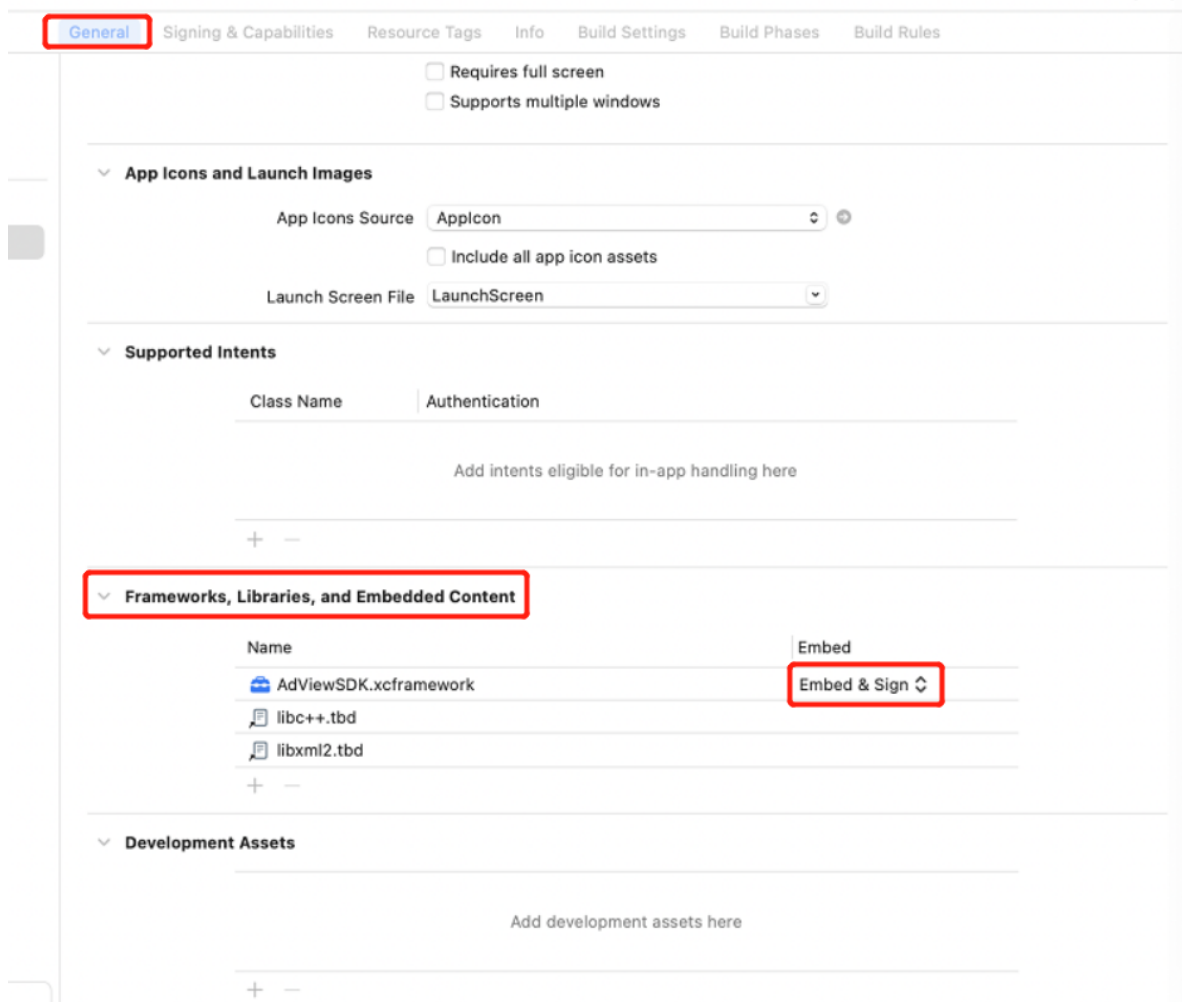
## Step 2: Obtain the Adapter & SDK Files

### Manual SDK Download

- **Option A: Manual Download (Recommended for most users)**

You should receive the following files from your Account Manager or via [partner@adview.com](mailto:partner@adview.com):

- AdviewSDK.xcframework
  - ISAdviewAdapter.xcframework
  - LevelPlay Demo project (optional but highly recommended)
- Important: In Xcode, set AdviewSDK.xcframework → Embed & Sign\*\*.



## Option B: CocoaPods (Easiest)

Add the adapter to your `Podfile`:

```
pod 'ISAdViewAdapter'
```

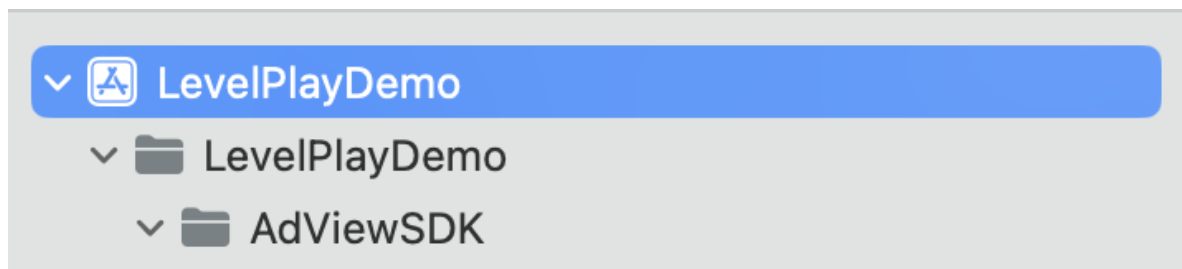
Then run:

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

## Step 3: Integrate into Your Xcode Project


1. Drag the following frameworks into your project (if not using CocoaPods):

- `AdViewSDK.xcframework`
- `ISAdViewAdapter.xcframework`
- `IronSourceSDK` (already added via LevelPlay)




 AdViewSDK


 ISAdViewAdapter


 AppDelegate

 AppDelegate

 DemoViewController


 DemoViewController


 DemoRewardedVideoAdDelegate


 DemoRewardedVideoAdDelegate


 DemoInterstitialAdDelegate


 DemoInterstitialAdDelegate


 DemoBannerAdDelegate

 DemoBannerAdDelegate

 DemoImpressionDataDelegate


 DemoImpressionDataDelegate

 Assets

 LaunchScreen

 Main

 Info

 main

>  Frameworks

✓  Products

 LevelPlayDemo


>  Pods

✓  Pods

 Podfile

>  Frameworks

>  Pods

>  Products

>  Targets Support Files

2. Ensure **Embed & Sign** is applied to AdViewSDK.
3. Build the project to confirm there are no linking errors.

## Demo Project

The provided LevelPlayDemo includes working examples for Banner, Interstitial, and Rewarded Video. Open `DemoViewController.m` to see full implementation with APPKEY and placement usage.



## Step 4: Ad Unit Implementation Examples

### Banner / MREC

Follow the steps described in the IronSource documentation for [Banner/MREC ads](#).

```
eg in DemoViewController:
- (void) createBannerAd {
    // choose banner size
    // 1. recommended - Adaptive ad size that adjusts to the screen width
    self.bannerSize = [LPMAdSize createAdaptiveAdSize];
    // 2. Adaptive ad size using fixed width ad size
    // self.bannerSize = [LPMAdSize createAdaptiveAdSizeWithWidth:400];
    // 3. Specific banner size - BANNER, LARGE, MEDIUM_RECTANGLE
    // self.bannerSize = [LPMAdSize mediumRectangleSize];
    // Create the banner view and set the ad unit id and ad size
    if (self.bannerSize != nil) {
        LPMBannerAdViewConfig *config = [[[LPMBannerAdViewConfigBuilder alloc]
init] initWithAdSize:self.bannerSize] build];
        self.bannerAd = [[LPMBannerAdView alloc] initWithAdUnitId:kBannerAdUnitId
config: config];
        // set the banner listener
        self.bannerAdViewDelegate = [[DemoBannerAdDelegate alloc]
initWithDelegate:self];
        [self.bannerAd setDelegate:self.bannerAdViewDelegate];
        [self addBannerToView];
        [self setEnablementForButton:LoadBannerButtonIdentifier
enable:YES];
    }
    else {
        NSLog(@"Error creating banner size");
    }
}

- (void)addBannerToView {
    dispatch_async(dispatch_get_main_queue(), ^{
        self.bannerAd.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:self.bannerAd];
        self.bannerAd.translatesAutoresizingMaskIntoConstraints = NO;
    });
}
```

```

        [NSLayoutConstraint activateConstraints:@[
            [self.bannerAd.bottomAnchor
            constraintEqualToAnchor:self.view.safeAreaLayoutGuide.bottomAnchor],
            [self.bannerAd.centerXAnchor
            constraintEqualToAnchor:self.view.centerXAnchor],
            [self.bannerAd.widthAnchor
            constraintEqualToConstant:self.bannerSize.width],
            [self.bannerAd.heightAnchor
            constraintEqualToConstant:self.bannerSize.height]
        ]];
    });
}
- (IBAction)loadBannerButtonTapped:(id)sender {
    [self logMethodName:@"loadAdWithViewController for banner"];
    [self.bannerAd loadAdWithViewController:self];
}

```

## Interstitial

Follow the steps described in the IronSource documentation for [interstitial ads](#).

```

eg in DemoViewController:
- (void) createInterstitialAd {
    self.interstitialAd = [[LPMInterstitialAd alloc]
    initWithAdUnitId:kInterstitialAdUnitId];
    self.interstitialAdDelegate = [[DemoInterstitialAdDelegate alloc]
    initWithDelegate:self];
    self.interstitialAd.delegate = self.interstitialAdDelegate;
    [self setEnablementForButton:LoadInterstitialButtonIdentifier
    enable:YES];
}
- (IBAction)loadInterstitialButtonTapped:(id)sender {
    // This will load an Interstitial ad
    [self logMethodName:@"loadAd for interstitial"];
    self.interstitialAd = nil;
    [self createInterstitialAd];
    [self.interstitialAd loadAd];
}
- (IBAction)showInterstitialButtonTapped:(id)sender {
    // It is advised to make sure there is available ad that isn't capped before
    attempting to show it
    if ([self.interstitialAd isAdReady]) {
        // This will present the Interstitial.
        [self logMethodName:@"showAdWithViewController for interstitial"];
        [self.interstitialAd showAdWithViewController:self placementName:NULL];
    } else {
        // load a new ad before calling show
    }
}
}

```

## Rewarded ads

Follow the steps described in the IronSource documentation for [rewarded ads](#).

```

eg in DemoViewController:

```

```

- (void) createRewardedAd {
    self.rewardedAd = [[LPMRewardedAd alloc] initWithAdUnitId:kRewardedAdUnit];
    self.rewardedVideoDelegate = [[DemoRewardedVideoAdDelegate alloc]
initWithDelegate:self];
    self.rewardedAd.delegate = self.rewardedVideoDelegate;
    [self setEnablementForButton:LoadRewardedVideoButtonIdentifier
        enable:YES];
}
- (IBAction)loadRewardedButtonTapped:(id)sender {
    // This will load rewarded ad
    [self logMethodName:@"loadAd for rewarded"];
    [self.rewardedAd loadAd];
}
- (IBAction)showRewardedVideoButtonTapped:(id)sender {
    // It is advised to make sure there is available ad before attempting to
show an ad
    if (self.rewardedAd != nil && self.rewardedAd.isAdReady) {
        // This will present the Rewarded Video.
        [self logMethodName:@"showAdWithViewController for rewarded"];
        [self.rewardedAd showAdWithViewController:self placementName:nil];
    } else {
        // load a new ad before calling show
    }
}
}

```

NOTE :Native Ad are not yet supported for custom adapters via ironSource

## Step 5: Privacy & Compliance

### GDPR / CCPA / GPP

AdViewSDK automatically reads consent information via delegate methods or `NSUserDefaults`.

#### Option 1 – Delegate (Recommended)

Implement the following protocols on your ad delegates:

```

@protocol AdViewViewDelegate <AdViewGDPRProtocolV1, AdViewGDPRProtocolV2>
@protocol AdViewVideoDelegate <AdViewGDPRProtocolV1, AdViewGDPRProtocolV2>
@protocol AdViewNativeAdDelegate <AdViewGDPRProtocolV1, AdViewGDPRProtocolV2>

@protocol AdViewGDPRProtocolV1 <NSObject>
@optional
- (BOOL)CMPPresent;
- (BOOL)subjectToGDPR;
- (NSString *)userConsentString;
- (NSString *)parsedPurposeConsents;
- (NSString *)parsedVendorConsents;

@protocol AdViewGDPRProtocolV2 <NSObject>
@optional
- (BOOL)GDPRApplies;
- (NSString *)TCFString;

```

#### Option 2 – NSUserDefaults (Quick setup)

```
// GDPR and CCPA save example in NSUserDefaults
/*AdView_IABConsent_CMPPresent = "IABConsent_CMPPresent";
AdView_IABConsent_SubjectToGDPR = "IABConsent_SubjectToGDPR";
AdView_IABConsent_ConsentString = "IABConsent_ConsentString";
AdView_IABConsent_ParsedPurposeConsents =
"IABConsent_ParsedPurposeConsents";
AdView_IABConsent_ParsedVendorConsents = "IABConsent_ParsedVendorConsents";
AdView_IABConsent_CCPA = "IABUSPrivacy_String";*/
/*[[NSUserDefaults standardUserDefaults] setObject:@"testCMPPresent"
forKey:AdView_IABConsent_CMPPresent];
[[NSUserDefaults standardUserDefaults] setObject:@"1"
forKey:AdView_IABConsent_SubjectToGDPR];
[[NSUserDefaults standardUserDefaults] setObject:@"testConsentString"
forKey:AdView_IABConsent_ConsentString];
[[NSUserDefaults standardUserDefaults] setObject:@"testPurposeConsents"
forKey:AdView_IABConsent_ParsedPurposeConsents];
[[NSUserDefaults standardUserDefaults] setObject:@"testVendorConsents"
forKey:AdView_IABConsent_ParsedVendorConsents];
[[NSUserDefaults standardUserDefaults] setObject:@"testCCPA"
forKey:AdView_IABConsent_CCPA];*/

/*GPP save example in NSUserDefaults
AdView_IABGPP_GppString = "IABGPP_HDR_GppString";
AdView_IABGPP_GppSID = "IABGPP_GppSID" */
/*[[NSUserDefaults standardUserDefaults] setObject:@"testGpp"
forKey:AdView_IABGPP_GppString];
[[NSUserDefaults standardUserDefaults] setObject:@"1,2"
forKey:AdView_IABGPP_GppSID];
```

## COPPA

For purposes of the [Children's Online Privacy Protection Act \(COPPA\)](#), there is a setting called isSubjectCOPPA.

As an app developer, you can indicate whether you want AdViewSDK to treat your content as child-directed when you make an ad request. When you indicate that you want AdViewSDK to treat your content as child-directed, AdViewSDK takes steps to disable IBA and remarketing ads on that ad request. The setting options are as follows:

Set isSubjectCOPPA to @YES to indicate that you want your content treated as child-directed for purposes of COPPA. This prevents the transmission of the Advertising Identifier, IDFA.

Set isSubjectCOPPA to @NO to indicate that you don't want your content treated as child-directed for purposes of COPPA.

Don't set isSubjectCOPPA if you don't want to indicate how you would like your content treated with respect to COPPA.

Call this `+(void)isSubjectCOPPA:(NSNumber *)isUserCoppa` before AdViewSDK requesting and loading ads.

```
eg:
[AdViewView isSubjectCOPPA:@YES];
self.banner = [AdViewView requestBannerSize:AdViewBannerSize_320x50
                                bannerType:AdViewBannerType_normal
                                positionId:@"POSID6uooxjk893j0"
                                delegate:self];
```

## **You're Done!**

Your app should now serve AdView ads through LevelPlay mediation.

Otherwise please contact your AM or [partner@adview.com](mailto:partner@adview.com).