Freestar Ads Mediation Swift iOS

Last Modified on 05/31/2022 11:37 am EDT

Overview

Change Log

Freestar provides an effective ad mediation solution. The Freestar mediation method is universal auction, not the traditional waterfall. Universal auction is more sophisticated than waterfall and provides, by far, the best eCPM. This document describes how to integrate the Freestar SDK into your Swift iOS app quickly and easily. This repo is a fully integrated Swift iOS sample app. Feel free to clone it, install the appropriate Cocoapods, open with Xcode and run it on a device.

> Note: You can remotely toggle on/off any of the following ad providers as you see fit using our web dashboard. *All* applicable providers are enabled by default.

The current version of the Freestar SDK is 5.11.0.

Ad Provider	SDK Version	Ad Unit Types
AdColony	4.7.2	Fullscreen Interstitial & Rewarded
AppLovin	11.0.0	Fullscreen Interstitial & Rewarded, Banner 320x50
AppLovinMax	11.0.0	Fullscreen Interstitial & Rewarded, Banner 300x250, Banner 320x50
Criteo	4.3.1	Fullscreen Interstitial & Rewarded, Banner 300x250, Banner 320x50
Admob	9.2.0	Fullscreen Interstitial & Rewarded, Banner 300x250, Banner 320x50, Native
Google Ads Manager	9.2.0	Fullscreen Interstitial & Rewarded, Banner 300x250, Banner 320x50, Native
MoPub (deprecated)	5.18.2	Fullscreen Interstitial & Rewarded, Banner 300x250, Banner 320x50, Native
Tapjoy	12.8.0	Fullscreen Rewarded
Unity Ads	4.1.0	Fullscreen Interstitial & Rewarded
Vungle	6.10.3	Fullscreen Interstitial & Rewarded
Google IMA	3.13.0	Preroll
Nimbus	1.10.5	Fullscreen Interstitial & Rewarded, Banner 300x250, Banner 320x50

TAM (Amazon Publisher Services)	4.3.0	Fullscreen Interstitial & Rewarded, Banner 300x250, Banner 320x50
Pangle	3.7.0.8	Fullscreen Interstitial & Rewarded, Banner 300x250, Banner 320x50, Native
Hyprmx	6.0.0	Fullscreen Interstitial & Rewarded, Banner 300x250, Banner 320x50
Yahoo	1.14.2	Fullscreen Interstitial, Banner 300x250, Banner 320x50

Project Setup

Your Podfile should have the following:

```
# Uncomment the next line to define a global platform for your project
platform :ios, '10.0' # minimum ios version

target 'FreestarSwiftSample' do # app name
  # Uncomment the next line if you're using Swift or would like to use dynamic frameworks
  # use_frameworks!
  pod 'FreestarAds', '~> 5.0' # required

  # Only specify the partners for which you want to run ads

  # M1 compatible
  pod "FreestarAds-AdColony", "~> 4.7"
  pod "FreestarAds-Tapjoy", "~> 12.8"
  pod "FreestarAds-Criteo", "~> 4.3"
  pod "FreestarAds-AppLovin", "~> 11.0"
  pod "FreestarAds-AppLovinMax", "~> 11.0"
  pod "FreestarAds-TAM", "~> 4.3"
  pod "FreestarAds-Hyprmx", "~> 6.0"
  pod "FreestarAds-Yahoo", "~> 1.14"
  pod 'FreestarAds-Googleadmob', '~> 9.2'
  pod 'FreestarAds-Googleadmob/Facebook', '~> 9.2'
  pod 'FreestarAds-GAM', '~> 9.2'
  pod 'FreestarAds-GAM/Facebook', '~> 9.2'
  pod "FreestarAds-Unity", "~> 4.1"

  # Not M1 compatible pods
  pod "FreestarAds-Vungle", "~> 6.10"
  pod "FreestarAds-Nimbus", "~> 1.10"
  pod "FreestarAds-Google", "~> 3.13"
  pod "FreestarAds-Pangle", "~> 3.7"
end
```

Once the podfile is setup, enter the Xcode project's base directory in the terminal and run `pod install`. Then open the generated `.xcworkspace` project with Xcode.

Info.plist

Some ad networks require adding special parameters in the app's `Info.plist` file. Below are the partner-specific `Info.plist` requirements.

Google Ad Manager

```
<key>GADIsAdManagerApp</key>
<true/>
```

Google Admob

```
<key>GADApplicationIdentifier</key>
<string>{YOUR_ADMOB_KEY}</string>
```

AppLovin

```
<key>AppLovinSdkKey</key>
<string>{YOUR_APPLOVIN_KEY}</string>
```

AdColony

```
<key>NSCalendarsUsageDescription</key>
<string>Adding events</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>Taking selfies</string>
<key>NSCameraUsageDescription</key>
<string>Taking selfies</string>
<key>NSMotionUsageDescription </key>
<string>Interactive ad controls</string>
```

In the above entry, you should change the reasons to be appropriate for your app.

Logging

You can enable detailed logging from the SDK to inspect the ad mediation process in detail via an app's console. This is done by setting a flag in the `Info.plist` file:

```
<key>FSTR_LOGGING_ENABLE</key>
<true />
```

Once this change is made, rebuild the app to see the logs.

> ⚠Warning: For both performance and security reasons, it is not advisable to have detailed logging in production apps. Remove the flag in the Info.plist before submitting to the App Store.

Using the Freestar SDK

To interface with the Freestar ad mediation, have the following at the top of the code file where you want to make use of the SDK:

`import FreestarAds`

GDPR Support

Freestar is GDPR-ready and supports the IAB Standards for GDPR compliance.

Use the following simple api in conjunction with your existing Consent Management Provider. If you do not have a CMP solution, that's ok, too! Our mediation sdk will detect if the user is in the EU and automatically apply GDPR actions to the ad request. So, by default, you do not have to do any extra work to use our sdk in a GDPR-compliant fashion.

```
// Save GDPR consent string
Freestar.privacySettings().subject(
  toGDPR: gdprApplies,
  withConsent: gdprConsentString)
```

Initialize Freestar

Freestar must be initialized in the `application(_:didFinishLaunchingWithOptions:)` of your `AppDelegate` (or at another time as close as possible to the app launch). This gives the prefetch mechanism time work and thus, makes ad fill more likely when a request is made.

```
class AppDelegate: UIResponder, UIApplicationDelegate {
  static let FREESTAR_APP_KEY = "P8RIA3"

  func application(_ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
      // Override point for customization after application launch.

      Freestar.initWithAppKey(AppDelegate.FREESTAR_APP_KEY)
      //more initialization code

      return true
    }
}
```

> Note: if you would like to see ads from TAM (Amazon Publisher Services), make sure your `AppDelegate` implements the `window` property from `UIApplicationDelegate` protocol:
>
> `var window : UIWindow? = UIWindow(frame: UIScreen.main.bounds)`
>
> The `window` parameter **must** be a `var` declaration with type `UIWindow?`

Interstitial Ad

Implement the `FreestarInterstitialDelegate` protocol in one of your classes:

```
func freestarInterstitialLoaded:(_ ad: FreestarInterstitialAd) {}
func freestarInterstitialFailed:(_ ad: FreestarInterstitialAd because reason: FreestarNoAdReason) {}
func freestarInterstitialShown:(_ ad: FreestarInterstitialAd) {}
func freestarInterstitialClicked:(_ ad: FreestarInterstitialAd) {}
func freestarInterstitialClosed:(_ ad: FreestarInterstitialAd) {}
```

This allows your app to listen to ad events and act appropriately. You will pass an instance of this object to the Freestar SDK when loading interstitial ad. In the current sample app, the class implementing this protocol is the `FullscreenAdViewController`, and the implementation is located in FullscreenAdViewController.swift

The actual implementation resides in an extension to the `FullscreenAdViewController` class

```
//self is the object that implements FreestarInterstitialDelegate
self.interstitialAd = FreestarInterstitialAd(delegate: self)

//You can load associated to a placement as follows, or pass in
//nil for the default placement
self.interstitialAd?.loadPlacement("interstitial_p1")
```

> If you plan to use more than one placement in your app, please adhere to the placement naming convention as follows:
>
> "my_placement_name_pN", where N is the number of your placement.
>
> For example, let us assume you are using 2 interstitial ad placements in your game or app. The first placement would be the default placement; simply do not specify a placement name by calling the **loadPlacement()** method with nil as the argument. The second placement would be, for example, "my_search_screen_p1". The ending "p1" tells the SDK to use the second placement you created in our web dashboard for the interstitial ad unit.
>
> This placement format is the same for all the other ad units, such as rewarded ads and banner ads.

When the interstitial ad is ready, the **freestarInterstitialLoaded()** callback will occur.

```
func freestarInterstitialLoaded(_ ad: FreestarInterstitialAd){
  //self in this case should be an instance of UIViewController
  ad.showFrom(self)  //You can display the ad now OR show it later; your choice.
}
```

There are other callbacks that will occur in other events, such as in the rare event where a load ad request does not result in a fill. Please see the FullscreenAdViewController.swift on this sample for those details.

> ⚠Warning: Attempting to load a new ad from the `freestarInterstitialFailed(because:)` method is **strongly** discouraged. If you must load an ad from `freestarInterstitialFailed:because:`, limit ad load retries to avoid continuous failed ad requests in situations such as limited network connectivity.

> The clicked callback is no longer reported for Googleadmob or GAM ads. This also applies to banner ads as well as interstitial.

Custom Targeting

Freestar Ads allows for custom targeting parameters that will be sent on to our Google Ads Manager adapter.

```
interstitial.addCustomTargeting("key1" as:"value1") //optional
interstitial.addCustomTargeting("key2" as:"value2") //optional
```

> Note: the `addCustomTargeting(:as)` method works in the same manner on rewarded, banner and preroll ads.

Banner Ad

FreeStar supports **300x250** and **320x50** banner ad formats and allows you to control the refresh intervals remotely.

```
smallBanner = FreestarBannerAd(delegate: self, andSize: .banner320x50)
smallBanner.loadPlacement(nil) //nil is the 'default' placement or pass in specific placement e.g. "inview_p1"
```

When the banner ad is ready, the **freestarBannerLoaded()** callback will occur.

```
func freestarBannerLoaded:(_ ad: FreestarBannerAd) {
  NSLog(@"banner ad as been loaded: %@", ad);
}
```

If you insert the banner ad into the view hierarchy at this point, the ad will display automatically.

Banner ads can also be specified in Interface Builder layout and will be automatically loaded. If you do this, however, you will need to set the `size` and `delegate` properties on the ad object before calling `loadPlacement()`

The `320x50` ad size a standard for iPhones, on iPads, the corresponding banner ad size is `728x90`. By default, a banner ad created with `.banner320x50` as the size parameter will be rendered as a `320x50` ad on iPhones, and as a `728x90` ad on iPads. If you want to *always* render the ad as `320x50`, regardless of device, you should set the `fixedSize` flag on the ad object:

```
smallBanner = FreestarBannerAd(delegate: self, andSize: .banner320x50)
smallBanner.fixedSize = true;
smallBanner.loadPlacement(nil) //or pass in specific placement
```

This will always show the in `320x50` size, no matter what device your app is running on.

Adaptive Banner (320x50 only)

Adaptive banner is a feature that allows for the option to serve dynamic banner sizes, i.e. a fullscreen width and a variable height. This presentation can give users an improved UX because of the fact that it is a fullscreen width format. Normal small banners are fixed at 320px in width, while adaptive banner can often times be 390px or more.
Note: To properly support adaptive banner, it is likely that the current project autolayout will need to be modified. Please see below examples.

Enable Adaptive Banner in AppDelegate

```
Freestar.setAdaptiveBannerEnabledIfAvailable(true)
Freestar.initWithAppKey('FREESTAR_API_KEY')
```

Calculate and Set the *adaptiveBannerWidth* Property (Optional)

Note: This property defaults to fullscreen width based on orientation

```
banner.adaptiveBannerWidth = calculateAdaptiveViewWidth()
banner.translatesAutoresizingMaskIntoConstraints = false  // autolayout is the preferred approach
func calculateAdaptiveViewWidth() -> CGFloat {
   let frame = { () -> CGRect in
     // Here safe area is taken into account, hence the view frame is used
     // after the view has been laid out.
     if #available(iOS 11.0, *) {
       return view.frame.inset(by: view.safeAreaInsets)
     } else {
       return view.frame
     }
   }()
   return frame.size.width
}
```

Autolayout Using Constraints

Since adaptive banner is a flexible width and/or height format, it may be required to change your current layout to support dynamic size. This is just one example of how this can be achieved, but is by no means, the only approach.

```
func freestarBannerLoaded(_ ad: FreestarBannerAd) {
   container?.addSubview(ad) // add banner to container

   ad.centerXAnchor.constraint(equalTo: container!.centerXAnchor).isActive = true  // horizontal center
   ad.centerYAnchor.constraint(equalTo: container!.centerYAnchor).isActive = true  // vertical center
   if (ad.isAdaptive) {  // check if current banner is adaptive
      // set adaptive width (but also ensure, using autolayout, there is sufficient height available for up to 90x)
      ad.widthAnchor.constraint(equalToConstant: calculateAdaptiveViewWidth()).isActive = true
   } else {
      // non-adaptive so set fixed width
      ad.widthAnchor.constraint(equalToConstant: ad.frame.width).isActive = true
      ad.heightAnchor.constraint(equalToConstant: ad.frame.height).isActive = true
   }
}
```

Rewarded Ad

> A common myth regarding Rewarded Ads is publishers are required to *give something* to the user. But, that's not true. You can simply tell the user they must watch the ad in order to be able to proceed to the next level or proceed to content.

Implement the `FreestarRewardedDelegate` protocol in one of your classes:

```
func freestarRewardedLoaded(_ ad: FreestarRewardedAd) {}
func freestarRewardedFailed(_ ad: FreestarRewardedAd, because reason: FreestarNoAdReason) {}
func freestarRewardedShown(_ ad: FreestarRewardedAd) {}
func freestarRewardedClosed(_ ad: FreestarRewardedAd) {}
func freestarRewardedFailedToStart(_ ad: FreestarRewardedAd, because reason: FreestarNoAdReason) {}
func freestarRewardedAd(_ ad: FreestarRewardedAd, received rewardName: String, amount rewardAmount: Int) {}
```

This allows your app to listen to ad events and act appropriately. You will pass an instance of this object to the Freestar SDK when loading rewarded ad. In the current sample app, the class implementing this protocol is the `FullscreenAdViewController`, and the implementation is located in FullscreenAdViewController.swift

```
//this may be overriden in the ad portal
let rew = FreestarReward.blank()
rew.rewardName = "Coins"
rew.rewardAmount = 1000
rew.userID = "CoolGuy"
rew.secretKey = "Bx34da3yn"

      //Note: If you want to use server-to-server rewarded callbacks, the Freestar team will provide
      //      you with a secret key and only that will work.
      //      However, if you want to use client-side only rewarded callbacks, then you can supply
      //      your own "secretKey" or null.

self.rewardedAd = FreestarRewardedAd(delegate: self, andReward: rew)
self.rewardedAd?.loadPlacement("rewarded_p1") //or pass in nil for default
```

When the rewarded ad is ready, the **freestarRewardedLoaded()** callback will occur.

```
func freestarRewardedLoaded:(_ ad: FreestarRewardedAd) {
  //self in this case should be an instance of UIViewController
  ad.showFrom(self) //You can display the ad now OR show it later; your choice.
}
```

When the user has fully watched the rewarded ad (or when the given ad partner determines sufficient watch time for the reward), the following callback will occur:

```
func freestarRewardedAd(_ ad: FreestarRewardedAd, received rewardName: String, amount rewardAmount: Int) {
  //allow user to proceed to app content or next level in app/game
  //can use the name/amount to show change in UI
}
```

When the user has closed the rewarded ad, the following callback will occur:

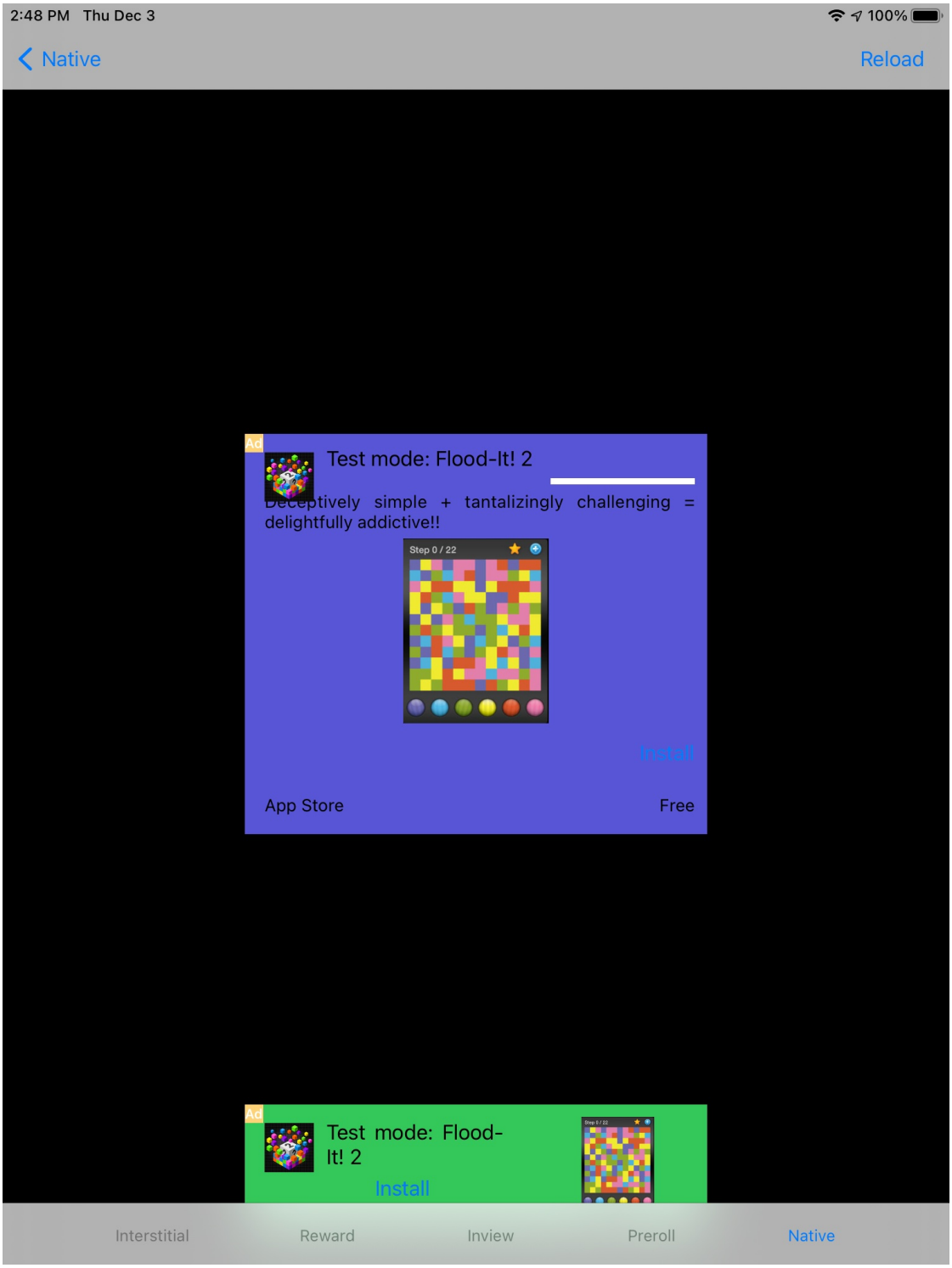
```
func freestarRewardedClosed(_ ad: FreestarRewardedAd) {}
```

> If the user does not watch the rewarded ad thru to completion, **freestarRewardedAd(received:amount:)** will not occur. However, the **freestarRewardedClosed()** will always occur when the rewarded ad is dismissed regardless if the user watched the entire rewarded ad or not.

> ⚠ Please assume that ads will expire in about 1 hour after the loaded callback. Meaning, you may *cache* an ad in your app or game, but must be displayed with the allotted hour.

Native Ad

Medium and Small Native ADs



Freestar supports the Native Ad unit. Native ads are well suited for content-based or feed-based apps because developers can actually customize the ad layout to better match their user experience.

Get Started

```
import FreestarAds
...

self.nativeAd = FreestarNativeAd(delegate: self, andSize: .medium) //or .small
self.nativeAd.loadPlacement(nil);
//Note: you may pass in a "placement" parameter in loadPlacement but it requires prior remote staff setup
```

Assuming you implemented the `FreestarNativeAdDelegate` protocol:

```
func freestarNativeClosed(_ ad: FreestarNativeAd)
func freestarNativeFailed(_ ad: FreestarNativeAd, because reason: FreestarNoAdReason)
func freestarNativeLoaded(_ ad: FreestarNativeAd)
func freestarNativeShown(_ ad: FreestarNativeAd)
func freestarNativeClicked(_ ad: FreestarNativeAd)
```

and set the delegate object on `nativeAd`, when the native ad is ready, the `freestarNativeLoaded` callback will occur.

```
func freestarNativeLoaded(_ ad: FreestarNativeAd) {
   self.nativeAd.center = self.view.center //choose the position of your ad within your app view
   self.view.addSubview(self.nativeAd)
}
```

Another Way

Another way to display native ads is to put `FreestarNativeAd` directly into your Interface Builder layout by placing a view where you want and assigning its class to be `FreestarNativeAd`.

NOTE: If you choose this route, then you do not need any code in Objective-C or Swift since the ad will automatically be fetched and displayed. Also, if you need to set targeting parameters in the AdRequest, then please use the programmatic approach (above) and do not put the native ad in the layout. However, to receive the delegate callbacks listed above, you will need to assign the delegate object to the ad from the XIB once it is loaded.

Native Templates

Freestar supports two native ad templates: **small and medium**

By default, you are not required to modify the native ad templates. However, if you want to modify, keep reading...

Modify Small and/or Medium Templates

If you wish to modify the templates, get a copy here.

Get the `Freestar-Native-Medium-Custom.xib` and `Freestar-Native-Small-Custom.xib` files and add them to your App bundle with Xcode. You may edit them with Interface Builder to better meet your needs. However, when you modify either of the `.xib` files, *please do not remove any elements, rename any `IBOutlets`, or change the tag property on any element in the hierarchy.* To drop back to using the default Freestar templates, simply remove the custom files from your app bundle.

App Open Ad

To enable App Open Ad(s), please add this code after SDK initialization and UI window setup.

```
Freestar.requestAppOpenAds(withPlacement: "interstitial_p1", waitScreen: true) { placement, event, error in }
```

Sample Project

All of this and more, such as *Preroll Ads* can be seen in the sample FreestarSwiftSample:

https://github.com/freestarcapital/SDK_documentation_iOS/blob/master/FreestarSwiftSample/FreestarSwift

Important iOS 14 Changes

One thing you must do as a publisher is set the Facebook `setAdvertiserTrackingEnabled` flag appropriately, since Facebook is one of our partners. Here is the official document from Facebook. Generally speaking, one typical you could approach this is if the user has granted the app permission to track them at the OS level, then you would set this flag to true. By default, the flag will be false, meaning that Facebook will not serve any ads. We leave this implementation detail to you in order to give you, the app publisher, more control.

GDPR Update

We are releasing a major update to the Freestar SDK later this week that will require publishers to utilize a GDPR TCF 2.0 compliant CMP in order to render ads to users in a GDPR affected country. If the publisher is not utilizing a GDPR TCF 2.0 compliant CMP once upgraded to this SDK version, then ads will not serve to users in a GDPR affected country.

We put together a comprehensive list of FAQs below to help break down how the changes may impact your business. If you have additional questions, please reach out to your dedicated Account Manager and we will be happy to help on a case by case basis.

What is the new SDK version?

4.0.0 for iOS and Android

How do I become GDPR compliant?

In order to be able to serve ads to your users who reside in any EU country, you will need to implement an IAB TCF 2.0 CMP service.

The CMP service is essentially a personal data and privacy form that must be presented to users to collect their consent or dissent. This form can be presented as often as you like throughout the session of your game or app, as you may require your users to see ads. The most typical setup would be for the first visit to an App with an option to edit your preferences through some other call to action.

What if I already have a CMP implemented?

If you already have a CMP implemented, you will need to add Publisher First, Inc. (Freestar's official legal entity) as a vendor, as well as each of our partners listed here if not already included.

You will need to retrigger the consent form for GDPR affected users once Freestar and additional vendors are added to the vendor list.

If the list of vendors has already been included, no additional effort is required.

How does the Freestar SDK work with a CMP?

Freestar SDK will automatically detect the user's CMP response and act accordingly. More specifically, if the user consents, then Freestar SDK will be allowed to serve ads. If the user dissents, then Freestar SDK will not show ads.

What if I choose not to implement a CMP?

If a CMP is not implemented at all, then Freestar SDK will not show ads to users who reside in a GDPR affected country.

What do I need to do once I choose a CMP?

After you have chosen a CMP Service provider, during configuration and setup of your CMP, you will need to include a list of supported vendors. We recommend that you select 'Include All Vendors' rather than choosing specific vendors, as your list of vendors may change over time.

If you choose to add vendors to your list manually, you will need to include Publisher First, Inc. (Freestar's official legal entity) as a vendor as well as Freestar's list of supported vendors here.

Does Great Britain still observe GDPR requirements despite leaving the E.U.?

Yes, it does still apply. In anticipation of Brexit, a new domestic data privacy law called the UK-GDPR took effect on 1/31/20. The UK-GDPR is almost word for word completely identical to the EU's GDPR.

CMP Recommendations:

Consent Manager - https://www.consentmanager.net/ App Consent - https://sfbx.io/en/produits/

Here is a list of IAB approved CMP Service providers you can implement in your game or app: https://iabeurope.eu/cmp-list/

Testing

For iOS, please use our iOS test key **91784edd-3492-4111-8742-f71bd3803dd3** (we have a different key for Android, so don't use for both iOS and Android) for all your iOS testing runs and enable test mode.

Turn on test mode:

```
FSTR_TEST_ADS
```

You will usually get 100% fill on all ad units.

It is not recommended to use your production key for testing runs as that is strictly prohibited by our partners and bad things may happen to us on the business side of things.
Do not forget to uninstall and re-install your app when changing keys on your device.

When you are satisfied with your testing, please make a release build with your production key, and turn test mode off. Publish to store.

SKAdNetwork IDs

iOS 14 changed the way advertising works on iOS devices. Ad effectiveness tracking requires the usage of `SKAdNetwork` APIs. To enable this, `SKAdNetwork` keys should be included in your app's `Info.plist` file.

Please see our list: https://github.com/freestarcapital/SDK_documentation_iOS/wiki/iOS-14-SKAdNetwork-IDs

M1 Macs

Currently, we do not support the building of our SDK on M1 macs. This will be addressed in an upcoming release.

Xcode 12 related cocoapods build errors

Recently, in Xcode 12.2, Apple made some breaking changes in Xcode related to build settings. By default, arm64 architecture is now added to ARCHS_STANDARD. In addition, they have removed support for VALID_ARCHS setting from Xcode. Apple also added, in Xcode 12, a new build setting called **Excluded Architectures**. The reason Apple added arm64 is to support simulator on M1 Mac hardware. However this causes build issues running simulator on Intel Macs. As a result, publishers running Xcode on Intel Macs will need to make usage of the EXCLUDED_ARCHS build setting, for simulator testing, to exclude the arm64 simulator. So, if your Xcode build is failing with arm64 simulator errors in the log, such as:

`building for iOS Simulator, but linking in object file built for iOS, file for architecture arm64`

then, it is suggested to use this cocoapods post install hook in your Podfile:

```
post_install do |installer|
    installer.pods_project.build_configurations.each do |config|
        config.build_settings["EXCLUDED_ARCHS[sdk=iphonesimulator*]"] = "arm64"
    end
end
```

This is a temporary fix and should address the issue of build failures resulting from Xcode attempting to build (arm64) simulator on Intel Macs. As of our latest SDK release, we do this automatically in the podspec via xcconfigs, however not every SDK vendor has adopted this workaround. This post install hook is only needed if you are running Xcode 12 on Intel hardware and wanting to run simulator, and you have a cocoapods dependency that is not Xcode 12 / arm64 compliant.

Firebase Dependency

If your app integrated the Firebase SDK, reference the Firebase version compatible with current release 7.0

pod 'Firebase/Core', '~> 7.0'

For further reference:
https://github.com/CocoaPods/CocoaPods/issues/10104

https://stackoverflow.com/questions/63607158/xcode-12-building-for-ios-simulator-but-linking-in-object-file-built-for-ios/63955114#63955114

Yahoo Demand Partner Configuration

If Yahoo is being added as a demand partner in your Podfile, then it is required to add an entry into your Info.plist. See example below:

```
<key>VerizonAdsSourceAppId</key>
<string>Replace_this_with_your_App's_App_Store_ID</string>
```

GAM and Googleadmob OB adapters

If you would like to make usage of Google Open Bidding (OB) adapters for FreestarAds, please contact your account manager (AM) for further instructions. We do not advise including OB adapters into your project without first consulting with your AM.