Freestar Ads Mediation ReactNative iOS

Last Modified on 05/31/2022 11:25 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 ReactNative iOS app quickly and easily. This repo is a fully integrated ReactNative iOS sample app. Feel free to clone it, install the appropriate Cocoapods, launch ReactNative 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.

See SDK Version and Supported Demand Source Partner Versions

Project Setup

The ReactNative environment generates an Xcode project on creation, including a companion Pods project to install appropriate dependencies via Cocoapods. To integrate the Freestar SDK, install the ReactNative plugin from npm:

```
npm install --save @freestar/freestar-plugin-react-native

NOTE: if you already have the plugin installed, you should update it:

npm update --save @freestar/freestar-plugin-react-native
```

Then edit the Podfile:

1. ReactNative generates a project with the minimum iOS target version of 9.0. FreestarAds mimum requirement is 10.0, and the Podfile needs to reflect that:

```
platform :ios, '10.0' # minimum ios version
```

2. You need to specify the partners for which you want to run ads via Freestar mediation. Append them to the React-generated Podfile:

```
pod 'FreestarAds-AdColony', '~> 5.0'
pod 'FreestarAds-AppLovin', '~> 5.0'
pod 'FreestarAds-Googleadmob', '~> 6.0'
pod 'FreestarAds-Tapjoy', '~> 5.0'
pod 'FreestarAds-Unity', '~> 6.0'
pod 'FreestarAds-Vungle', '~> 5.0'
pod 'FreestarAds-Criteo', '~> 3.0'
pod 'FreestarAds-GAM', '~> 5.0'
pod 'FreestarAds-Mopub', '~> 5.0' # (deprecated)
pod 'FreestarAds-Nimbus', '~> 3.0'
pod 'FreestarAds-TAM', '~> 2.0'
pod 'FreestarAds-Pangle', '~> 1.0'
```

Rerun `pod install` in the `ios` directory of the ReactNative project. 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>CHP_LOGGING_ENABLE</key>
<string>true</string>
```

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 FreestarReactBridge from 'freestar-plugin-react';
```

```
import FreestarReactBridge from "@freestar/freestar-plugin-react-native";
```

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
FreestarReactBridge.subjectToGDPR(
    gdprApplies,      //boolean
    gdprConsentString //string
);
```

Initialize Freestar

Freestar must be initialized 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.

```
export default function App(props) {
    let FREESTAR_API_KEY = "P8RIA3";
    FreestarReactBridge. initWithAdUnitID(FREESTAR_API_KEY);

    //more initialization code

    return (
        //overall react UI
    );
}
```

Interstitial Ad

Provide the SDK with the callback function for events related to interstitial ads:

```
FreestarReactBridge.subscribeToInterstitialCallbacks((eventName) => {
   if(eventName === "onInterstitialLoaded") {
      // ad can now be shown
   } else if (eventName === "onInterstitialClicked") {
      // ad clicked
   } else if (eventName === "onInterstitialShown") {
      // ad shown
   } else if (eventName === "onInterstitialFailed") {
      // no ad available
   } else if (eventName === "onInterstitialDismissed") {
      //ad has closed, resume the app
   } else {
     console.log("unknown event");
   }
 });
```

This allows your app to listen to ad events and act appropriately. In the current sample app, this is implemented in the `FullscreenAds()` function of the App.js file.

To receive any callbacks, you will need to load the ad:

```
//You can load associated to a placement as follows, or pass in
// the empty string for the default placement
FreestarReactBridge.loadInterstitialAd("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 **loadInterstitialAd()** method with `""` 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 **"onInterstitialLoaded** callback event will arrive.

```
FreestarReactBridge.subscribeToInterstitialCallbacks((eventName) => {
   if(eventName === "onInterstitialLoaded") {
      // ad can now be shown
      //You can display the ad now OR show it later; your choice.
      FreestarReactBridge.showInterstitialAd();
   }
   //other events
}
```

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 App.js on this sample for those details.

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

Banner Ad

FreeStar supports **300x250** (`MREC`) and **320x50** (`BANNER`) banner ad formats and allows you to control the refresh intervals remotely. You can use ReactNative to place the ads in appropriate locations in your app.

To do so, first, import the banner ad objects from the plugin:

```
import BannerAd from '@freestar/freestar-plugin-react-native/BannerAd';
import MrecBannerAd from '@freestar/freestar-plugin-react-native/MrecBannerAd';
import MrecBannerAd2 from '@freestar/freestar-plugin-react-native/MrecBannerAd2';
```

MrecBannerAd comes in separate classes to enable displaying the MREC-size ad in different contexts without them overriding each other -- a reload of an ad in one banner does not affect the others. There are 4 built-in options: `MrecBannerAd` , `MrecBannerAd2` , `MrecBannerAd3` and `MrecBannerAd4` , with identical behavior. The same setup applies to the Small and Medium native ad units (see below): 4 distinct options for each.

Then setup the ad:

```
  <BannerAd
     style={{width: 320, height: 50}}
     requestOptions={{ size:'BANNER' }}
     onBannerAdLoaded={bannerLoaded}
     onBannerAdFailedToLoad={bannerFailed}
   />
```

And for MREC:

```
<View style={{ flexDirection: 'row', paddingTop: 10}}>
 <BannerAd
    style={{width: 300, height: 250}}
    requestOptions={
     {
        size:'MREC',
        isCoppaEnabled: false,
        targetingParams: {
             'someparam1': 'somevalue1',
             'someparam2': 'somevalue2',
             'someparam3': 'somevalue3',
        },
        testDeviceIds: ['deviceId1','deviceId2', 'deviceId3']
     }
    }
    onBannerAdLoaded={bannerLoaded}
    onBannerAdFailedToLoad={bannerFailed}
 />
</View>

 <View style={{ flexDirection: 'row', paddingTop: 10}}>
 <BannerAd2
  style={{width: 300, height: 250}}
  requestOptions={{}}
  onBannerAdLoaded={bannerLoaded}
  onBannerAdFailedToLoad={bannerFailed}
 />
</View>
```

The `bannerLoaded` and `bannerFailed` should be functions implemented by you to handle the corresponding events. The ad will load automatically after being inserted into the view hierarchy, and display when it is ready.

Note: the `requestOptions` parameter is required. If your app has no additional parameters to pass in, add `requestOptions={{}}` to the banner ad element.

Custom Targeting

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

```
<BannerAd
       style={{width: 300, height: 250}}
       requestOptions={
         {
           size:'MREC',
           targetingParams: {
                'someparam1': 'somevalue1',
                'someparam2': 'somevalue2',
                'someparam3': 'somevalue3'
           }
         }
       }
       onBannerAdLoaded={bannerLoaded}
      onBannerAdFailedToLoad={bannerFailed}
      />
```

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.

Provide the SDK with the callback function for events related to rewarded ads:

```
FreestarReactBridge.subscribeToRewardCallbacks((eventName, rewardName = '', rewardAmount = 0) => {
   if (eventName === "onRewardedFailed") {
     // no ad available
   } else if (eventName === "onRewardedDismissed") {
     // ad closed
   } else if(eventName === "onRewardedLoaded") {
     // ad ready
   } else if (eventName === "onRewardedCompleted") {
     // received reward
     console.log("reward ad completed: awarded " + rewardAmount + ' ' + rewardName);
   } else if (eventName === "onRewardedShown") {

   } else if (eventName === "onRewardedShowFailed") {
     // couldn't display ad
   } else {
     console.log("unknown event");
   }
 });
```

This allows your app to listen to ad events and act appropriately. In the current sample app, this is implemented in the `FullscreenAds()` function of the App.js file.

To receive any callbacks, you will need to load the ad:

```
//You can load associated to a placement as follows, or pass in
// the empty string for the default placement
FreestarReactBridge.loadRewardAd("rewarded_p1");
```

When the rewarded ad is ready, the"**onRewardedLoaded** callback event will arrive.

```
FreestarReactBridge.subscribeToRewardCallbacks((eventName, rewardName = '', rewardAmount = 0) => {
   if(eventName === "onRewardedLoaded") {
     // ad can now be shown
     //You can display the ad now OR show it later; your choice.
     FreestarReactBridge.showRewardAd(
        "Coins",   //reward name
        50,        //reward amount
        "myuserId",//user id
        "12345678" //secret key
     );
   }
   //other callback events
});
```

When the user has fully watched the rewarded ad (or when the given ad partner determines sufficient watch time for the reward), the `onRewardedCompleted` callback event will arrive.

When the user has closed the rewarded ad, the `onRewardedDismissed` callback event will arrive.

> If the user does not watch the rewarded ad thru to completion, **onRewardedCompleted** callback event will not arrive, but the **onRewardedDismissed** event always arrive 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 Ads

Organizationally, native ads work similar to banner ads, but permit customization of how ad data is displayed. They are only supported by the following partners:

- Mopub (deprecated)
- Googleadmob
- GAM

To display native ads in a ReactNative app, first import the native ad objects from the plugin:

```
import SmallNativeAd from '@freestar/freestar-plugin-react-native/SmallNativeAd';
import MediumNativeAd2 from '@freestar/freestar-plugin-react-native/MediumNativeAd';
```

And then add them to the app at the appropriate location:

```
<View style={{ flexDirection: 'row', paddingTop: 30}}>
 <MediumNativeAd
    style={{width: 300, height: 250}}
    requestOptions={
     {
        targetingParams: {
            'someparam1': 'somevalue1',
            'someparam2': 'somevalue2',
            'someparam3': 'somevalue3',
        },
        testDeviceIds: ['deviceId1','deviceId2', 'deviceId3']
     }
    }
    onNativeAdLoaded={nativeLoaded}
    onNativeAdFailedToLoad={nativeFailed}
 />
 </View>

 <View style={{ flexDirection: 'row', paddingTop: 10}}>
 <SmallNativeAd
  style={{width: 320, height: 50}}
  requestOptions={{}}
  onNativeAdLoaded={nativeLoaded}
  onNativeAdFailedToLoad={nativeFailed}
 />
</View>
```

Just like with banners, the `nativeLoaded` and `nativeFailed` should be functions implemented by you to handle the corresponding events, and the `requestOptions` parameter is required. If your app has no additional parameters to pass in, add `requestOptions={{}}` to the native ad element.

Sample Project

All of this and more, can be seen in the sample FreestarReactNativeSample:

https://github.com/freestarcapital/SDK_documentation_iOS/blob/master/FreestarReactNativeSample

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.