Freestar Ads Mediation React Native Android

Last Modified on 05/31/2022 12:10 pm EDT

Freestar Ads Mediation provides support for React Native

Getting Started

Start displaying Freestar Ads in your existing React Native app running on Android today by following the simple steps below.

Requirements

• Before we begin, you must have a working React Native app running on an Android device. This document will not show how to create a React Native app running under Android as that would be beyond the scope of Freestar Ads Mediation.

• Make sure you have the latest version of Android Studio.

Modify Android Project

Open the Android native build project using Android Studio. It is located in **[your react project]/android** folder.

Follow the Native Android Project Setup Guide here, which is modification of the **build.gradle** and **AndroidManifest.xml**

After making the changes to **build.gradle and AndroidManifest.xml**, return here. It should not take very long. Make sure you uncommented the following line in your app/build.gradle:

```
implementation 'com.freestar.android.ads:react-native-android:1.1.1'
```

Remove Old Java Files

In our older integration documentation, there were 3 java source files:

```
FreestarReactPackage.java
FreestarReactBridge.java
FreestarBannerAdViewManager.java
```

Please remove these as they are now located within the dependency:

```
implementation 'com.freestar.android.ads:react-native-android:1.1.1'
```

Install our npm package

In your React Native project root folder, install our Freestar npm package:

```
npm install --save @freestar/freestar-plugin-react-native
(the latest npm package version is 1.2.3)

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

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

Modify MainApplication.java

In Android Studio, open **MainApplication.java**

In the **getPackages()** method, add our **FreestarReactPackage**:

It should look *something* similar to this:

```
    @Override
    protected List getPackages() {
      List packages = new PackageList(this).getPackages();
      packages.add(new ModuleRegistryAdapter(mModuleRegistryProvider));
      packages.add(new com.freestar.android.ads.react.FreestarReactPackage());  //This is the only Freestar line in this
file
      return packages;
    }
```

Modify MainActivity.java

In Android Studio, open **MainActivity.java**

In the **onCreate** method of **MainActivity.java** initialize Freestar. It should look *something* like this:

```
//Put these import statements at the top
import com.freestar.android.ads.AdRequest;
import com.freestar.android.ads.FreeStarAds;

 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  //Freestar begin

  /**
   * Note: When the enable-test-ads flag is set to [true], there will be a dialog prompt to choose
   * mediation partners.  Set to [false] to remove that dialog and when getting ready to release
   * to production.
   *
  */
  FreeStarAds.enableTestAds(true); //set false for production
  FreeStarAds.enableLogging(true); //set false for production
  AdRequest adRequest = new AdRequest(this);
  adRequest.addCustomTargeting("someParam1", "someValue1"); //optional targeting param for pre-fetched ads
  adRequest.addCustomTargeting("someParam2", "someValue2"); //optional targeting param for pre-fetched ads

  //Uncomment the following line if you wish to utilize App Open Ads in your Android app
  //FreeStarAds.requestAppOpenAds("app-open-ad-placement", true, null);

  FreeStarAds.init(this, "XqjhRR", adRequest);  //Use our test key "XqjhRR" for all testing runs

  //The following init also works if you don't use custom targeting
  //FreeStarAds.init(this, "XqjhRR");

  //Freestar end
 }
```

Displaying Ads

The final piece to the puzzle is displaying ads in React Native js code.

In your application js, make sure to import our Freestar React plugin:

```
import FreestarReactBridge from '@freestar/freestar-plugin-react-native';
import BannerAd from '@freestar/freestar-plugin-react-native/BannerAd';
import MrecBannerAd from '@freestar/freestar-plugin-react-native/MrecBannerAd';
import SmallNativeAd from '@freestar/freestar-plugin-react-native/SmallNativeAd';
import MediumNativeAd from '@freestar/freestar-plugin-react-native/MediumNativeAd';
```

App Open Ads

App Open Ad is a fullscreen ad that shows when the app starts (including 'cold starts') and when the app is resumed from the background.

To utilize App Open Ads in your Android app, call the following method before FreeStarAds.init in MainActivity.java:

```
FreeStarAds.requestAppOpenAds("app-open-ad-placement", true, null);   //Obtain a proper placement from our Solutio
ns Team
FreeStarAds.init(...);
```

Fullscreen Interstitial Ads

We start by subscribing to the Interstitial callbacks:

```
FreestarReactBridge.subscribeToInterstitialCallbacks((eventName, placement) => {
      if(eventName === "onInterstitialLoaded") {

        if (placement == 'not defined') {  //Important: Check if the placement comes in as the literal 'not defined'
           placement = null;
        }
        FreestarReactBridge.showInterstitialAd(placement);  //placement cannot be empty string '', but can be null or an
y string.

      } else if (eventName === "onInterstitialClicked") {

      } else if (eventName === "onInterstitialShown") {

      } else if (eventName === "onInterstitialFailed") {

         Alert.alert('Interstitial Ad not available');

      } else if (eventName === "onInterstitialDismissed") {

      } else {
        console.log("unknown event");
      }
    });
```

To load an interstitial ad:

Note: Please do not "prefetch" the next interstial ad on app startup or after dismissals or no-fills; we do this automatically and internally for you.

```
//In this case, we use the default placement, so we pass null for the placement name.
FreestarReactBridge.loadInterstitialAd(null);
```

Fullscreen Rewarded Ads

We start by subscribing to the Rewarded callbacks:

```
FreestarReactBridge.subscribeToRewardCallbacks((eventName, placement, rewardName = '', rewardAmount = 0) =>
 {
    if (eventName === "onRewardedFailed") {

        Alert.alert('Reward Ad not available');

    } else if (eventName === "onRewardedDismissed") {

    } else if(eventName === "onRewardedLoaded") {

       if (placement == 'not defined') {  //Important: Check if the placement comes in as the literal 'not defined'
          placement = null;
       }
       //placement cannot be empty string '', but can be null or any string.
       FreestarReactBridge.showRewardAd(placement, "Coins", 50, "myuserId", "12345678");

    } else if (eventName === "onRewardedCompleted") {

       console.log("reward ad completed: awarded " + rewardAmount + ' ' + rewardName);

    } else if (eventName === "onRewardedShown") {

    } else if (eventName === "onRewardedShowFailed") {

        Alert.alert('Reward Ad was available but failed to show');

    } else {
      console.log("unknown event");
    }
  });
```

To load a Rewarded ad:

Note: Please do not "prefetch" the next rewarded ad on app startup or after dismissals or no-fills; we do this automatically and internally for you.

```
//In this case, we use the default placement, so we pass null for the placement name.
FreestarReactBridge.loadRewardAd(null);
```

Small Banner Ad 320x50

```
    <BannerAd
      style={{width: 320, height: 50}}    //Do not use other values besides 320 and 50 for small banner
      requestOptions={
        {
          size: 'BANNER',
          //placement: 'home_page_p1' //NOTE: if this placement has not been setup in the back-end, then do NOT spe
cify placement
          targetingParams: {
              'someparam1': 'somevalue1',
              'someparam2': 'somevalue2',
              'someparam3': 'somevalue3',
          },
          testDeviceIds: ['deviceId1','deviceId2', 'deviceId3']
        }
      }
      onBannerAdLoaded={someBannerLoadedHandler}
      onBannerAdFailedToLoad={someBannerFailedHandler}
    />
```

Medium Rectangle Banner Ad 300x250 - MREC

```
    <BannerAd
      style={{width: 300, height: 250}}    //Do not use other values besides 300 and 250 for mrec banner
      requestOptions={
        {
          size: 'MREC',
          //placement: 'home_page_p1' //NOTE: if this placement has not been setup in the back-end, then do NOT spe
cify placement
          targetingParams: {
              'someparam1': 'somevalue1',
              'someparam2': 'somevalue2',
              'someparam3': 'somevalue3',
          },
          testDeviceIds: ['deviceId1','deviceId2', 'deviceId3']
        }
      }
      onBannerAdLoaded={someBannerLoadedHandler}
      onBannerAdFailedToLoad={someBannerFailedHandler}
    />
```

Another simple way to display MREC is as follows. Specify 'MrecBannerAd' and do not specify the 'size' in requestOptions.

```
    <MrecBannerAd
      style={{width: 300, height: 250}}    //Do not use other values besides 300 and 250 for mrec banner
      requestOptions={
        {
          //placement: 'home_page_p1' //NOTE: if this placement has not been setup in the back-end, then do NOT spe
cify placement
          targetingParams: {
              'someparam1': 'somevalue1',
              'someparam2': 'somevalue2',
              'someparam3': 'somevalue3',
          },
          testDeviceIds: ['deviceId1','deviceId2', 'deviceId3']
        }
      }
      onBannerAdLoaded={someBannerLoadedHandler}
      onBannerAdFailedToLoad={someBannerFailedHandler}
    />
```

Small Native Ad 360x100

```
    //Note: The ad width should actually be the full screen width.  360 (below) is for demonstration purposes.
    <SmallNativeAd
      style={{width: 360, height: 100}}
      requestOptions={
        {
          //placement: 'home_page_p1' //NOTE: if this placement has not been setup in the back-end, then do NOT spe
cify placement
          targetingParams: {
              'someparam1': 'somevalue1',
              'someparam2': 'somevalue2',
              'someparam3': 'somevalue3',
          },
          testDeviceIds: ['deviceId1','deviceId2', 'deviceId3']
        }
      }
      onNativeAdLoaded={someNativeLoadedHandler}
      onNativeAdFailedToLoad={someNativeFailedHandler}
    />
```

Medium Native Ad 360x350

```
    //Note: The ad width should actually be the full screen width.  360 (below) is for demonstration purposes.
    <MediumNativeAd
      style={{width: 360, height: 350}}
      requestOptions={
        {
          //placement: 'home_page_p1' //NOTE: if this placement has not been setup in the back-end, then do NOT spe
cify placement
          targetingParams: {
              'someparam1': 'somevalue1',
              'someparam2': 'somevalue2',
              'someparam3': 'somevalue3',
          },
          testDeviceIds: ['deviceId1','deviceId2', 'deviceId3']
        }
      }
      onNativeAdLoaded={someNativeLoadedHandler}
      onNativeAdFailedToLoad={someNativeFailedHandler}
    />
```

For more information on how to optionally customize the look and feel of the **SmallNativeAd** or **MediumNativeAd** please see our Android Native Ad Unit document.

Please have a look at our sample code to see how it all works: App.js

Privacy - Google Play Families Policy Compliance

If your game or app is officially under the Google Play Families program, Freestar provides such support:

```
FreeStarAds.setGoogleFamilyPolicyMode( GoogleFamilyPolicyMode.app, true); //If your app is designed only for childre
n
//FreeStarAds.setGoogleFamilyPolicyMode( GoogleFamilyPolicyMode.mixed, false);  //If your app is designed for famili
es with children
FreeStarAds.init(...);
```

If your app is not officially under the Google Play Families program, then you do not need to set the Google Family Policy mode.

```
/**
 * Only set Google Families Policy Mode if your app is required to comply with Google Play’s
 * Families Policy program.
 *
 * @param googleFamilyPolicyMode app: the app is child-directed and will not receive
 *                               personalize or contextual ads.
 *                               mixed: the app is directed at mixed audiences.
 *                               none: (default) the app is not required to comply with Google
 *                                     Play's Family Policy
 *
 * @param onlyNonPersonalizedAds true: if 'mixed' mode, then only personalized or contextual ads
 *                                    may be served.
 *                               false: if 'mixed' mode, then personalized or contextual ads
 *                                     may be served.
 *                               note:  if 'app' mode, then personalized or contextual ads
 *                                     may not be served regardless of this parameter.
 */
public static void setGoogleFamilyPolicyMode(GoogleFamilyPolicyMode googleFamilyPolicyMode,
                                     boolean onlyNonPersonalizedAds)
```

GDPR

Freestar SDK is GDPR compliant. In order for your users to be able to receive any ad fills in GDPR-affected countries, you, as a publisher, will need to implement a 3rd party Consent Management Platform (CMP). Freestar SDK will automatically detect user interactions with the CMP and act accordingly. Please see our Freestar GDPR Frequently Asked Questions for complete details and our recommended list of CMP service providers.

Testing

For Android, please use our test key **XqjhRR** for all your testing runs and enable test mode true. 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.

Automated Testing - Bypassing Ads

Are your automated tests failing after integrating Freestar Mediation Ads into your mobile application or game? Are you not sure it could be due to Freestar or something else? We have a feature called **Automated Test Mode** where you can run your automated tests to bypass Freestar or run Freestar in 'Limited Mediation' mode without making drastic changes to your code:

In your automated test suite code, **before** FreeStarAds.init is called:

```
FreeStarAds.setAutomatedTestMode( FreeStarAds.AutomatedTestMode.BYPASS_ALL_ADS )
  //OR
FreeStarAds.setAutomatedTestMode( FreeStarAds.AutomatedTestMode.LIMITED_MEDIATION )    //only runs AdMob & GAM
```

Again, this is only for your automated tests and not for production use.

Release Build

Final note when creating your release build: if you use **proguard or minify**, please add our proguard rules to your app proguard file.