Integration Methods/Guides

- iOS SDK
- Android SDK
- Unity SDK
- React Native SDK
- Flutter SDK

iOS SDK

RapidoReach iOS Integration Guide

Get Your API Key

Sign-up for a new developer account and create a new iOS app here and copy your API Key.

Install SDK

Install via Cocoapods (easiest)

```
    pod 'RapidoReachSDK', :git => 'https://github.com/skondgekar/roriossdk.git'
```

To install

```
   pod install
```

Flags

On the Build Settings tab and type in `Other Linker Flags` in the search field. Add the following flags.

- ObjC

Set Required Build Settings

We utilize Apple's Advertising ID (IDFA) to identify users. When uploading your app we recommend that you check all the boxes to note that your app uses IDFA and receives a smooth approval process. On the Info tab add in a Dictionary called `NSAppTransportSecurity`. Make sure you add this dictionary on the `Top Level Key`. Inside this dictionary, add a Boolean called `NSAllowsArbitraryLoads` and set it to YES. An example of your `info.plist` can be found here.

Initialize

After you have finished modifying the project settings, open your `AppDelegate.swift` file and import the RapidoReach SDK. Replace the YOUR_API_TOKEN with the actual api key found on your app. Replace YOUR_USER_ID with your unique ID for your appuser. If you do not have a unique user ID we recommend just using their `Apple Advertising ID (IDFA)`. If you utilize a server-side callback, this is the user ID that will be passed back to you when a user earns a reward.

```
// AppDelegate.swift
import RapidoReachSDK
```

Next initialize the RapidoReach SDK in your applicationDidFinishLaunchingWithOptions method.

```
// AppDelegate.swift

static let RapidoReachAPIKey = "<YOUR_API_TOKEN>"
static let RapidoReachUSER = "<YOUR_USER_ID>"

RapidoReach.shared.configure(apiKey: AppDelegate.RapidoReachAPIKey, user:
AppDelegate.RapidoReachUSER)
```

Reward Center

Open the .swift file of the controller where you want your users to have access to RapidoReach Reward Center. Call the `presentSurvey` method when you are ready to the send the user into the reward center where they can complete surveys in exchange for your virtual currency. We automatically convert the amount of currency a user gets based on the conversion rate specified in your app.

```
// ViewController.swift
// Import rapidoreach SDK
import RapidoReach

// Call for AppUserId
  // Do any additional setup after loading the view, typically from a nib.
  RapidoReach.shared.delegate = self
  // Fetch userId
  RapidoReach.shared.fetchAppUserID()

// Start reward center
```

```
RapidoReach.shared.setNavigationBarText(for: "Rapidoreach")
RapidoReach.shared.presentSurvey()
```

Reward Callback

To ensure safety and privacy, we notify you of all awards via a server side callback. In the developer dashboard for your App add the server callback that we should call to notify you when a user has completed an offer. Note the user ID pass into the initialize call will be returned to you in the server side callback. More information about setting up the callback can be found in the developer dashboard.

The quantity value will automatically be converted to your virtual currency based on the exchange rate you specified in your app. Currency is always rounded in favor of the app user to improve happiness and engagement.

Client Side Award Callback

For security purposes we always recommend that developers utilize a server side callback, however we also provide APIs for implementing a client side award notification if you lack the server structure or a server altogether or want more real-time award notification. It's important to only award the user once if you use both server and client callbacks (though your users may not be opposed!).

```
import RapidoReach

extension ViewController: RapidoReachDelegate {
    func didSurveyAvailable(_ available: Bool) {
        print("ROR: Surveys available "+(available ? "Available" : "Not Available"));
    }



    func didOpenRewardCenter() {
        print("didOpenRewardCenter")
    }

    func didClosedRewardCenter() {
        print("didClosedRewardCenter")
    }

    func didGetRewards(_ reward: RapidoReachReward) {
```

```
        print("RapidoReach Rewards Available: \(reward.total_rewards)")
        self.user?.rewards = reward
        self.bindReward()
    }

    func didGetError(_ error: RapidoReachError) {
        print("didGetError: "+error.localizedDescription)
    }

}
```

Testing SDK

When you initially create your app we automatically set your app to Test mode. While in test mode a survey will always be available. Note - be sure to set your app to Live in your dashboardt before your app goes live or you won't serve any real surveys to your users!

Customizing SDK

We provide several methods to customize the navigation bar to feel like your app.

```
RapidoReach.shared.setNavigationBarColor(for: "#00796B")
RapidoReach.shared.setNavigationBarTextColor(for: "#FFFFFF")
RapidoReach.shared.setStatusBarStyle(for: "light") // 'light' or 'dark' depending on what color of Navigation Bar is selected
```

Android SDK

RapidoReach Android Integration Guide

Get Your API Key

Sign-up for a new developer account and create a new Android app here and copy your API Key.

Download the SDK

Download the latest version of the Android SDK here. Add the RapidoReach-1.0.0.aar file to your projects "libs" folder.

Update your module's build.gradle file

Include the rapidoreach.aar, Google Play Services and androidx.appcompat in your build.gradle file. A Google Advertising ID helps us serve offers and surveys so we recommend adding the `Google Play Services SDK` to your project. Be sure your `minSdkVersion` is set to at least 16.

```
apply plugin: 'com.android.application'
...

android {
    ...
    defaultConfig {
        ...
        minSdkVersion 16
        ...
    }
```

```
    ...
}

dependencies {
    ...
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.gms:play-services-ads:19.2.0'
    implementation (name: 'RapidoReach-1.0.0', ext: 'aar')
}
```

Update your projects's build.gradle file

Ensure that your project has access to the 'libs' file by including the following in your project level build.gradle file.

```
buildscript...

allprojects {
    repositories {
        jcenter()
        flatDir {
            dirs 'libs'
        }
    }
}
```

Proguard

If you use proguard in your app. Be sure to add this line to your rules file:

```
-keep class rapidoreach.com.** { *; }
```

Import SDK in your android activity (MainActivity.java)

```
import com.rapidoreach.rapidoreachsdk.RapidoReach;
import com.rapidoreach.rapidoreachsdk.RapidoReachRewardListener;
import com.rapidoreach.rapidoreachsdk.RapidoReachSurveyAvailableListener;
import com.rapidoreach.rapidoreachsdk.RapidoReachSurveyListener;
```

Implement interfaces and methods (MainActivity.java)

```
public class MainActivity extends AppCompatActivity implements RapidoReachRewardListener,
RapidoReachSurveyListener, RapidoReachSurveyAvailableListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public void onReward(int i) {

    }

    @Override
    public void rapidoReachSurveyAvailable(boolean b) {

    }

    @Override
    public void onRewardCenterClosed() {

    }

    @Override
    public void onRewardCenterOpened() {

    }
```

```
}
```

Initialize RapidoReach

In your activity overwrite the onCreate() method and initialize the RapidoReach SDK with the `initWithApiKeyAndUserIdAndActivityContext` call. And implement the RapidoReach onPause() and onResume() calls. Replace the `YOUR_API_TOKEN` with the actual api key found on your app. Replace `YOUR_USER_ID` with your unique ID for your appuser. If you do not have a unique user ID we recommend just using their Google Advertising ID (GPS_ID)

```
//initialize RapidoReach
RapidoReach.initWithApiKeyAndUserIdAndActivityContext(`YOUR_API_TOKEN`, `YOUR_USER_ID`, this);

//customize navigation header
RapidoReach.getInstance().setNavigationBarText("Demo App");
RapidoReach.getInstance().setNavigationBarColor("#211548");
RapidoReach.getInstance().setNavigationBarTextColor("#FFFFFF");

//set reward and survey status listeners
RapidoReach.getInstance().setRapidoReachRewardListener(this);
RapidoReach.getInstance().setRapidoReachSurveyListener(this);
RapidoReach.getInstance().setRapidoReachSurveyAvailableListener(this);
```

Reward Center

Next, in your activity, implement the logic to display the reward center. Call the showRewardCenter method when you are ready to the send the user into the reward center where they can complete surveys in exchange for your virtual currency. We automatically convert the amount of currency a user gets based on the conversion rate specified in your app.

```
Button btn = (Button) findViewById(R.id.button);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Log.d(TAG, "Button is clicked");
        if (RapidoReach.getInstance().isSurveyAvailable()) {
            RapidoReach.getInstance().showRewardCenter();
        }
```

```
    }
});
```

Reward Callback

To ensure safety and privacy, we notify you of all awards via a server side callback. In the developer dashboard for your App add the server callback that we should call to notify you when a user has completed an offer. Note the user ID pass into the initialize call will be returned to you in the server side callback. More information about setting up the callback can be found in the developer dashboard.

The quantity value will automatically be converted to your virtual currency based on the exchange rate you specified in your app. Currency is always rounded in favor of the app user to improve happiness and engagement.

Client Side Award Callback

For security purposes we always recommend that developers utilize a server side callback, however we also provide APIs for implementing a client side award notification if you lack the server structure or a server altogether or want more real-time award notification. It's important to only award the user once if you use both server and client callbacks (though your users may not be opposed!).

```
public class MyActivity extends Activity implements RapidoReachRewardListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        // initialize RapidoReach
        super.onCreate();
        RapidoReach.initWithApiKeyAndUserIdAndActivityContext("YOUR_API_TOKEN",
"YOUR_USER_ID", "YOUR_ACTIVITY");

        // set RapidoReach client-side reward listener
        RapidoReach.getInstance().setRapidoReachRewardListener(this);

    }

    // implement callback for award notification
    @Override
```

```
    public void onReward(int i) {
        Log.d(TAG, "onReward: " + i);

    }
}
```

Reward Center Events

You can optionally listen for the onRewardCenterOpened and onRewardCenterClosed events by implementing the `RapidoReachSurveyListener` interface.

```
public class MyActivity extends Activity implements RapidoReachSurveyListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        // initialize RapidoReach
        super.onCreate();
        RapidoReach.initWithApiKeyAndUserIdAndActivityContext("YOUR_API_TOKEN",
"YOUR_USER_ID", "YOUR_ACTIVITY");

        // set RapidoReach survey event listener
        RapidoReach.getInstance().RapidoReachSurveyListener(this);

    }

    // reward center opened. time to start earning content!
    @Override
    public void onRewardCenterOpened() {
        Log.d(TAG, "onRewardCenterOpened");
    }

    // reward center closed. restart music/app.
    @Override
    public void onRewardCenterClosed() {
        Log.d(TAG, "onRewardCenterClosed");
    }
}
```

Survey Available Callback

If you'd like to be notified when a survey is available you can add a listener:

```
public class MyActivity extends Activity implements RapidoReachSurveyAvailableListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        // initialize RapidoReach
        super.onCreate();
        RapidoReach.initWithApiKeyAndUserIdAndActivityContext("YOUR_API_TOKEN",
"YOUR_USER_ID", "YOUR_ACTIVITY");

        // set RapidoReach survey available listener
        RapidoReach.getInstance().setRapidoReachSurveyAvailableListener(this);

    }

    // implement callback for survey available
    @Override
    public void rapidoReachSurveyAvailable(int surveyAvailable) {
        Log.d(TAG, "rapidoreachSurveyAvailable: " + surveyAvailable);
    }
}
```

Testing SDK

When you initially create your app we automatically set your app to Test mode. While in test mode a survey will always be available. Note - be sure to set your app to Live in your dashboard before your app goes live or you won't serve any real surveys to your users!

Customizing SDK

We provide several methods to customize the navigation bar to feel like your app.

```
RapidoReach.getInstance().setNavigationBarText("Demo App");
RapidoReach.getInstance().setNavigationBarColor("#17b4b3");
RapidoReach.getInstance().setNavigationBarTextColor("#FFFFFF");
```

Unity SDK

RapidoReach Unity Integration Guide

Get Your API Key

Sign-up for a new developer account and create a new Unity app here and copy your API Key.

Download the Plugin

Download the latest version of the RapidoReach Unity Plugin here.

Import the Unity Package

From Unity go to Assets menu → Import package → Custom package → choose the unzipped unity package.

Android

Ensure the RapidoReach-1.0.0.aar and other files were successfully imported with the RapidoReach.cs file in your "Assets/Plugins" folder.

In your player settings ensure the minimum API is set to 15 (Jelly Bean) or higher.

Initialize RapidoReach

We recommend initializing RapidoReach as soon as possible so we can begin preparing surveys for the user. In the Initialize method you'll set your API key and the user's ID that will be passed back into your server side callback when the user has earned currency for completing a survey.

```
#if UNITY_ANDROID || UNITY_IOS
```

```
  // Your GameObject that triggers the Reward Center

  void Start()
  {
    ConfigureRapidoReach();
  }
    public void ConfigureRapidoReach()
  {
#if UNITY_IOS || UNITY_ANDROID
    RapidoReach.Initialize("d5ece53df8ac97409298325fec81f3f7", "ANDROID_TEST_ID");
    RapidoReach.SetListener(gameObject.name);

    // optional
    RapidoReach.SetNavigationBarText("RapidoReach Unity N");
    RapidoReach.SetNavigationBarColor("#211548");
    RapidoReach.SetNavigationBarTextColor("#FFFFFF");
#endif
  }

  // call this function to show reward center on button click etc
  public void showRewardCenter(){
    Debug.Log("UNITY: Calling show reward center");
#if UNITY_IOS || UNITY_ANDROID
    RapidoReach.ShowRewardCenter();
#endif
  }

  void OnReward(string quantity)
  {
    FindObjectOfType<ReceivedRewards>().GetComponent<Text>().text = quantity;
    Debug.Log("RapidoReach OnReward: " + quantity);
  }

  void OnRewardCenterOpened()
  {
    FindObjectOfType<ReceivedRewards>().GetComponent<Text>().text = "Loading ...";
    Debug.Log("RapidoReach OnRewardCenterOpened!");
  }

  void OnRewardCenterClosed()
```

```
  {
    Debug.Log("RapidoReach OnRewardCenterClosed!");
  }

  void RapidoReachSurveyAvailable(string available)
  {
    Debug.Log("RapidoReach RapidoReachSurveyAvailable: " + available);
  }

  #endif
```

Google Play Services

A Google Advertising ID helps us serve offers and surveys so we recommend adding Google Play Services to your project. We provide the files to pull this in for you.

iOS

Ensure the UnityPluginBridge.mm and RapidoReachSDK-Bridging-Header.h files and frameworks folder are in your "Assets/Plugins/iOS" folder and the RapidoReach.cs file is in your "Assets/Plugins" folder.

Initialize RapidoReach

We recommend initializing RapidoReach as soon as possible so we can begin preparing surveys for the user. In the Initialize method you'll set your API key and the user's ID that will be passed back into your server side callback when the user has earned currency for completing a survey.

```
  #if UNITY_ANDROID || UNITY_IOS

  // Your GameObject that triggers the Reward Center

  void Start()
  {
    ConfigureRapidoReach();
  }
    public void ConfigureRapidoReach()
  {
#if UNITY_IOS || UNITY_ANDROID
    RapidoReach.Initialize("d5ece53df8ac97409298325fec81f3f7", "ANDROID_TEST_ID");
```

```
    RapidoReach.SetListener(gameObject.name);

    // optional
    RapidoReach.SetNavigationBarText("RapidoReach Unity N");
    RapidoReach.SetNavigationBarColor("#211548");
    RapidoReach.SetNavigationBarTextColor("#FFFFFF");
#endif
  }

  // call this function to show reward center on button click etc
  public void showRewardCenter(){
    Debug.Log("UNITY: Calling show reward center");
#if UNITY_IOS || UNITY_ANDROID
    RapidoReach.ShowRewardCenter();
#endif
  }

  void OnReward(string quantity)
  {
    FindObjectOfType<ReceivedRewards>().GetComponent<Text>().text = quantity;
    Debug.Log("RapidoReach OnReward: " + quantity);
  }

  void OnRewardCenterOpened()
  {
    FindObjectOfType<ReceivedRewards>().GetComponent<Text>().text = "Loading ...";
    Debug.Log("RapidoReach OnRewardCenterOpened!");
  }

  void OnRewardCenterClosed()
  {
    Debug.Log("RapidoReach OnRewardCenterClosed!");
  }

  void RapidoReachSurveyAvailable(string available)
  {
    Debug.Log("RapidoReach RapidoReachSurveyAvailable: " + available);
  }
```

```
#endif
```

Libraries

We include a post processing script to automatically include all libraries into your project and perform any additional configuration in the build settings. See the iOS guide if you would like to verify your Xcode project setup.

Reward Callback

To ensure safety and privacy, we notify you of all awards via a server side callback. This callback will be triggered for both Android and iOS apps.

In the developer dashboard for your App add the server callback that we should call to notify you when a user has completed an offer. Note the user ID pass into the initialize call will be returned to you in the server side callback. More information about setting up the callback can be found in the developer dashboard.

The quantity value will automatically be converted to your virtual currency based on the exchange rate you specified in your app. Currency is always rounded in favor of the app user to improve happiness and engagement.

Client Side Award Callback

For security purposes we always recommend that developers utilize a server side callback, however we also provide APIs for implementing a client side award notification if you lack the server structure or a server altogether or want more real-time award notification. It's important to only award the user once if you use both server and client callbacks (though your users may not be opposed!).

In order to receive notifications you must implement the OnReward method on the GameObject you'd like to receive notifications that a user earned content. Then you must register that gameObject with the SetListener method.

Additionally you can also implement OnRewardCenterOpened and OnRewardCenterClosed to listen to events and RapidoReachSurveyAvailable to listen to when a survey is available.

```
#if UNITY_ANDROID || UNITY_IOS

// Your GameObject that triggers the Reward Center
```

```
  void Start()
  {
    ConfigureRapidoReach();
  }
    public void ConfigureRapidoReach()
  {
#if UNITY_IOS || UNITY_ANDROID
    RapidoReach.Initialize("d5ece53df8ac97409298325fec81f3f7", "ANDROID_TEST_ID");
    RapidoReach.SetListener(gameObject.name);

    // optional
    RapidoReach.SetNavigationBarText("RapidoReach Unity N");
    RapidoReach.SetNavigationBarColor("#211548");
    RapidoReach.SetNavigationBarTextColor("#FFFFFF");
#endif
  }

  // call this function to show reward center on button click etc
  public void showRewardCenter(){
    Debug.Log("UNITY: Calling show reward center");
#if UNITY_IOS || UNITY_ANDROID
    RapidoReach.ShowRewardCenter();
#endif
  }

  void OnReward(string quantity)
  {
    FindObjectOfType<ReceivedRewards>().GetComponent<Text>().text = quantity;
    Debug.Log("RapidoReach OnReward: " + quantity);
  }

  #endif
```

Testing SDK

When you initially create your app we automatically set your app to Test mode. While in test mode a survey will always be available. Note - be sure to set your app to Live in your dashboardt before your app goes live or you won't serve any real surveys to your users!

Customizing SDK

We provide several methods to customize the navigation bar to feel like your app.

```
RapidoReach.SetNavigationBarText("RapidoReach Unity N");
RapidoReach.SetNavigationBarColor("#211548");
RapidoReach.SetNavigationBarTextColor("#FFFFFF");
```

React Native SDK

@rapidoreachsdk/react-native-rapidoreach

Before you start

Get your API key

Sign-up for a new developer account and create a new app here and copy your API Key.

Getting started

`$ npm install @rapidoreachsdk/react-native-rapidoreach`

`$ yarn add react-native-webview`

`$ cd ios && pod install && cd ..` # CocoaPods on iOS needs this extra step

We are all set up! Now let's use the module.

Usage

Initialize RapidoReach

First, you need to initialize the RapidoReach instance with `initWithApiKeyAndUserId` call.

```
// Import RapidoReach native module
import RapidoReach from '@rapidoreachsdk/react-native-rapidoreach';

componentDidMount() {
  // In your app initialization, initialize RapidoReach
  RapidoReach.initWithApiKeyAndUserId('YOUR_API_TOKEN', 'YOUR_USER_ID');
}
```

Reward Center

Next, implement the logic to display the reward center. Call the `showRewardCenter` method when you are ready to send the user into the reward center where they can complete surveys in exchange for your virtual currency. We automatically convert the amount of currency a user gets based on the conversion rate specified in your app.

```
onPressShowRewardCenter = () => {
  RapidoReach.isSurveyAvailable((isAvailable) => {
    // if a survey is available, show the reward center
    if (isAvailable) {
      RapidoReach.showRewardCenter();
    }
  })
}
```

Reward Callback

To ensure safety and privacy, we recommend using a server side callback to notify you of all awards. In the developer dashboard for your App add the server callback that we should call to notify you when a user has completed an offer. Note the user ID pass into the initialize call will be returned to you in the server side callback. More information about setting up the callback can be found in the developer dashboard.

The quantity value will automatically be converted to your virtual currency based on the exchange rate you specified in your app. Currency is always rounded in favor of the app user to improve happiness and engagement.

Client Side Award Callback

If you do not have a server to handle server side callbacks we additionally provide you with the ability to listen to client side reward notification.

First, import Native Module Event Emitter:

```
import { RapidoReachEventEmitter } from '@rapidoreachsdk/react-native-rapidoreach';
```

Then, add event listener for award notification (in `componentWillMount`, for example):

```
this.onRewardListener = RapidoReachEventEmitter.addListener(
  'onReward',
  this.onReward,
);
```

Implement the callback:

```
onReward = (quantity) => {
  console.log('reward quantity: ', quantity);
}
```

Reward Center Events

You can optionally listen for the `onRewardCenterOpened` and `onRewardCenterClosed` events that are fired when your Reward Center modal is opened and closed.

Add event listeners for `onRewardCenterOpened` and `onRewardCenterClosed`:

```
this.onRewardCenterOpenedListener = RapidoReachEventEmitter.addListener(
  'onRewardCenterOpened',
  this.onRewardCenterOpened,
);
this.onRewardCenterClosedListener = RapidoReachEventEmitter.addListener(
  'onRewardCenterClosed',
  this.onRewardCenterClosed,
);
```

Implement event callbacks:

```
onRewardCenterOpened = () => {
  console.log('onRewardCenterOpened called!');
}

onRewardCenterClosed = () => {
  console.log('onRewardCenterClosed called!');
```

```
}
```

Survey Available Callback

If you'd like to be proactively alerted to when a survey is available for a user you can add this event listener.

First, import Native Module Event Emitter:

```
import { RapidoReachEventEmitter } from '@rapidoreachsdk/react-native-rapidoreach';
```

Then, add event listener for award notification (in `componentWillMount`, for example):

```
this.rapidoreachSurveyAvailableListener = RapidoReachEventEmitter.addListener(
  'rapidoreachSurveyAvailable',
  this.rapidoreachSurveyAvailable,
);
```

Implement the callback:

```
rapidoreachSurveyAvailable = (surveyAvailable) => {
  if (surveyAvailable == "true") {
    console.log('rapidoreach survey is available');
  } else {
    console.log('rapidoreach survey is NOT available');
  }
}
```

Finally, don't forget to remove your event listeners in the `componentWillUnmount` lifecycle method:

```
componentWillUnmount() {
  this.onRewardListener.remove();
  this.onRewardCenterOpenedListener.remove();
  this.onRewardCenterClosedListener.remove();
  this.rapidoreachSurveyAvailableListener.remove();
}
```

Contact

Please send all questions, concerns, or bug reports to admin@rapidoreach.com.

FAQ

What do you do to protect privacy?

We take privacy very seriously. All data is encrypted before being sent over the network. We also use HTTPS to ensure the integrity and privacy of the exchanged data.

What kind of analytics do you provide?

Our dashboard will show metrics for sessions, impressions, revenue, and much more. We are constantly enhancing our analytics so we can better serve your needs.

What is your fill rate?

We have thousands of surveys and add hundreds more every day. Most users will have the opportunity to complete at least one survey on a daily basis.

I'm ready to go live! What are the next steps?

Let us know! We'd love to help ensure everything flows smoothly and help you achieve your monetisation goals!

Following the rewarded and/or theOfferwall approach

An example is provided on Github that demonstrates how a publisher can implement the rewarded and/or the Offerwall approach. Upon survey completion, the publisher can reward the user.

Limitations / Minimum Requirements

This is just an initial version of the plugin. There are still some limitations:

- You cannot pass custom attributes during initialization

- No tests implemented yet
- Minimum iOS is 9.0 and minimum Android version is 16

For other RapidoReach products, see RapidoReach docs.

ReactNativeSDK

Flutter SDK

flutter_rapidoreach

A plugin for Flutter that supports rendering surveys using RapidoReach SDKs. You can install rapidoreach flutter plugin from here

Note: RapidoReach iOS SDK utilizes Apple's Advertising ID (IDFA) to identify and retarget users with RapidoReach surveys. As of iOS 14 you should initialize RapidoReach Flutter plugin in iOS only if the relevant IDFA permission was granted by the user

Initializing the plugin

The RapidoReach plugin must be initialized with a RapidoReach API Key. You can retrieve an API key from RapidoReach Dashboard when you sign up and create a new app.

Usage

Initialize RapidoReach

First, you need to initialize the RapidoReach instance with `init` call.

```
// Import RapidoReach package
import 'package:rapidoreach/RapidoReach.dart';


RapidoReach.instance.init(apiKey: 'YOUR_API_TOKEN', userId: 'YOUR_USER_ID')
```

Reward Center

Next, implement the logic to display the reward center. Call the `show` method when you are ready to send the user into the reward center where they can complete surveys in exchange for your virtual currency. We automatically convert the amount of currency a user gets based on the

conversion rate specified in your app.

```
RapidoReach.instance.show(),
```

Reward Callback

To ensure safety and privacy, we recommend using a server side callback to notify you of all awards. In the developer dashboard for your App add the server callback that we should call to notify you when a user has completed an offer. Note the user ID pass into the initialize call will be returned to you in the server side callback. More information about setting up the callback can be found in the developer dashboard.

The quantity value will automatically be converted to your virtual currency based on the exchange rate you specified in your app. Currency is always rounded in favor of the app user to improve happiness and engagement.

Client Side Award Callback

If you do not have a server to handle server side callbacks we additionally provide you with the ability to listen to client side reward notification.

```
RapidoReach.instance.setOnRewardListener(onRapidoReachReward);
```

Implement the callback:

```
void onRapidoReachReward(int quantity) {
    print('TR: $quantity');
}
```

Reward Center Events

You can optionally listen for the `setRewardCenterOpened` and `setRewardCenterClosed` events that are fired when your Reward Center modal is opened and closed.

Add event listeners for `onRewardCenterOpened` and `onRewardCenterClosed`:

```
RapidoReach.instance
        .setRewardCenterClosed(onRewardCenterClosed);
RapidoReach.instance
        .setRewardCenterOpened(onRewardCenterOpened);
```

Implement event callbacks:

```
void onRewardCenterOpened() {
  print('onRewardCenterOpened called!');
}

void onRewardCenterClosed() {
  print('onRewardCenterClosed called!');
}
```

Survey Available Callback

If you'd like to be proactively alerted to when a survey is available for a user you can add this event listener.

First, import Native Module Event Emitter:

```
RapidoReach.instance
        .setSurveyAvaiableListener(onRapidoReachSurveyAvailable);
```

Implement the callback:

```
void onRapidoReachSurveyAvailable(int survey) {
    print('TR: $survey');
}
```

Following the rewarded and/or theOfferwall approach

An example is provided on Github that demonstrates how a publisher can implement the rewarded and/or the Offerwall approach. Upon survey completion, the publisher can reward the user.

Limitations / Minimum Requirements

This is just an initial version of the plugin. There are still some limitations:

- You cannot pass custom attributes during initialization
- No tests implemented yet
- Minimum iOS is 9.0 and minimum Android version is 16

For other RapidoReach products, see RapidoReach docs.

Getting Started

If you would like to review an example in code please review the Github project.