Dynamic Yield Implementation Guide for Mobile Apps Using React Native

Last updated June 15, 2020

When working with React Native, you can create a Dynamic Yield site of either iOS or Android. You can manage both your android and iOS output in one Dynamic Yield site of either iOS or Android. However, If you are running different experiments in on iOS and Android, you can also create two Dynamic Yield sites.

Implementing Dynamic Yield for React Native can be broken down the following steps:

1. Installing the SDK for iOS

1. Create a ReactNative project or use your existing project
2. Navigate into folder **ios** inside your project's path
3. To install using Cocoapods, add the following text to the end of the pod list inside **Podfile**:

```
pod 'Dynamic-Yield-iOS-SDK'
```

   Run **pod install**

4. Clone the following github repository: https://github.com/DynamicYield/mobile-sdk-react-native-intg. For access to this repository, speak to your Customer Success Manager.
5. In your ReactNative project's ios folder, open the **.xcworkspace** file to open the xcode project.
6. From the repository you just cloned, copy **DYReactImplTest/ios/DYReact.m** and **DYReactImplTest/ios/DYReact.h** into the xcode project

2. Installing the SDK for Android

1. Create a ReactNative project or use your existing project
2. Start Android studio and open the project inside the android folder
3. Add **implementation ('com.dynamicyield:DYAPISDK:+'){ transitive = false;}** to the end of the dependencies list inside the **build.gradle** file of your app module
4. Sync the gradle file
5. Clone this repo
6. Copy **DYReactImplTest/android/app/src/main/java/com/dyreactimpltest/bridge/DYReact.java** and **DYReactImplTest/android/app/src/main/java/com/dyreactimpltest/bridge/DYReactPackage.java** into your project
7. Add **packages.add(new DYReactPackage());** in the **getPackages** method of the **MainApplication.java** file

3. React Native Usage

1. In your ReactNative project, in every file that will reference the Dynamic Yield SDK, add the following code:
   ```
   import {NativeModules} from 'react-native';
   const DYReact = NativeModules.DYReact;
   ```
2. Inside your ReactNative project, in the root file (main.js by default) add the following code to the default class App method:
3. Listen to **this.state.DYReturned**, or start using DY inside the **setSecret** callback

4. Initiating Dynamic Yield

Implement the following API that will let you to know if DY is ready to be activated.

```
DYReact.setSecret(secretKey,callback)
```

Parameters

- *secretKey: String key obtained from the DY Admin Console*
- *callback: A function of type (state: ExperimentsState) that is called with the results of the Dynamic Yield SDK initialization results.*

Example

```
DYReact.setSecret('569bb4b1234564315e122c', state => {
```

```
      console.log(`DY ready with state: ${state} `);
    });
```

5. Tracking Pageviews

To send information to Dynamic Yield about user activity on your app, you need to trigger an explicit API call each time a user views a new page (or screen). This allows you to monitor activity in your app and to target experiences based on specific page views. In addition, many campaigns are initiated, triggered and/or targeted by the page the user is viewing. This is dependent on the correct implementation of page view tracking.

We recommend sending PageViews for every "logical" page change done by the user (i.e a new view has been opened).

```
DYReact.pageView(pageUniqueID, context, callback)
```

Parameters

- *pageUniqueID*: Unique identifier of the page as maintained in the application
- *context*: The current screen's context as a JSON object.
- *callback*: A function of type (name: String) that is called when a pageview is tracked. This is used to wait for the data to be available before running something that is dependent on that data.

Examples

```
DYReact.pageView('homepage', {lng: 'en_US', type: 'HOMEPAGE'}, pageName
=> {
      if (pageName) {
        console.log(`page ${pageName} processed`);
      }
    });
```

Page Context Examples

Page Type	Required Attributes	Page Context Examples
Homepage	-	`{type: 'HOMEPAGE'}`
Category	Array of category names in hierarchical order (array of strings)	`{type:'CATEGORY', data: ['TOP_LEVEL_CAT', 'CHILD_CAT','GRANDCHILD_CAT']}`
Product	SKU (string)	`{type:'PRODUCT', data: ['SKU123']}`

Cart	SKUs (array of strings)	`If there are items in the cart:`
		`{type:'CART', data: ['SKU123','SKU234']}`
		`If cart is empty:`
		`{type:'CART', data: ['']}`

Other	-	`{type: 'OTHER'}`

For more information on page context, see Page Context for Mobile.

6. Reporting Events

Events send data to Dynamic Yield representing user actions such as purchases or logins. They are used for behavioral targeting, reporting and recommendations. Implement the following required events on your app as specified. For more information about these events, or to learn about additional events you can implement, see Events for Mobile. **The following events should be implemented for eCommerce applications.**

1.Purchase

Trigger this event on every successful user purchase.

JavaScript Example

```
DYReact.trackEvent(
      'Purchase',{
     uniqueTransactionId: "123456", // Will remove redundant events. Must be a
string. Max 64 characters.
    dyType: "purchase-v1", // Identifies this event as a purchase
    value: 90.55, // Total cart value in actual payment currency, as float
(dollars dot cents)
    currency: "any supported currency code", // Optional non-default currency
used
    cart: [
      // itemPrice is per quantity of one, and in the same format as the total
"value":
      // float of dollars.cents in the payment currency (which is only defined
once above)
      {
        productId: "item-34454", // SKU exactly as in the product feed!
        quantity: 1,
        itemPrice: 65.87,
        size: "XL" // Size is *optional and a customer-specific string*
      }, {
        productId: "sku-4324-bg",
        quantity: 2,
        itemPrice: 12.34,
        size: "M"
      }
    ]
  }
});
```

2. Add to Cart

Trigger one add to cart event every time a user adds a product to the cart anywhere on your app (product page, category page, etc).

JavaScript example:

```
DYReact.trackEvent(
      'Add to Cart',
      {
        dyType: 'add-to-cart-v1',
        value: 59.13,
        currency: 'any supported currency code',
        productId: 'item-34454',
        quantity: 1,
        size: 'XL',
        cart: [
          {
            productId: 'sku-4324-bg',
            quantity: 2,
            itemPrice: 12.34,
          },
          {
            productId: 'item-34454',
            quantity: 1,
            itemPrice: 34.45,
          },
        ],
      },
      name => {
        console.log(`event: ${name}`);
      },
    );
```

3. Login

Trigger this event every time the user logs in, or on the first pageview.

JavaScript example:

```
DYReact.trackEvent(
      'Login',
      {
        dyType: 'login-v1',
        hashedEmail: '...', // SHA256 encoding of the lowercase e-mail, if known
(optional)
        cuid: '...', // If not identifying by e-mail, pass the customer ID
(optional)
        cuidType: '...', // If not identifying by e-mail, pass the customer ID
type (optional)
      },
      name => {
        console.log(`event: ${name}`);
      },
    );
```

4. Signup

Trigger this event when the user signs up, or on the first pageview. You can add metadata to the event as desired.

JavaScript Example

```
DYReact.trackEvent(
      'Signup',
      {
        dyType: 'signup-v1',
        hashedEmail: '...', // SHA256 encoding of the lowercase e-mail, if the
e-mail is known
      },
      name => {
        console.log(`event: ${name}`);
      },
    );
```

5. Identify API

Used to identify the user across platforms (e.g. web, iOS, CRM, email); returns Yes if identifiers is not null or empty. We recommend calling this method wherever the user enters their customer ID (e.g. checkout).

JavaScript Example:

```
DYReact.identifyUser(
      {
        uid: 'c0e93cee791b35af528a825f6476e8108e5f03e481ee39800a31a75559cdba2e',
//SHA 256 hashed email of the plain text email in lower case
```

```
        type: 'he',
      },
      err => {
        if (err) {
          console.log(`got error: ${err}`);
        }
      },
    );
```

6. Remove from Cart (required for Triggered Emails)

Trigger this event when the user removes an item from the cart. This ensures that any emails triggered based on the purchase have up to date information about the user’s cart.

JavaScript Example

```
DYReact.trackEvent(
      'Remove from Cart',
      {
        dyType: 'remove-from-cart-v1',
        value: 34.45,
        currency: 'any supported currency code',
        productId: 'gswefd-34-454',
        quantity: 1,
        size: 'XL',
        cart: [
          {
            productId: 'sku-4324-bg',
            quantity: 2,
            itemPrice: 12.34,
          },
          {
            productId: 'item-34454',
            quantity: 1,
            itemPrice: 34.45,
          },
        ],
      },
      name => {
        console.log(`event: ${name}`);
      },
    );
```

7. Sync Cart (required for Triggered Emails & Push Notifications)

Trigger this event when the user makes any changes to their cart such as cart loaded from the server when a user logs-in. This ensures that any relevant emails or push notifications presenting the cart content have up to date information.

JavaScript Example:

```
DYReact.trackEvent(
      'Sync cart',
      {
        dyType: 'sync-cart-v1',
        currency: 'any supported currency code', // Optional non-default currency
used
        cart: [
          //Mandatory, the order of products should be from oldest to newest
          {
            productId: 'sku-4324-bg',
            quantity: 2,
            itemPrice: 12.34,
          },
          {
            productId: 'item-34454',
            quantity: 1,
            itemPrice: 34.45,
          },
        ],
      },
      name => {
        console.log(`event: ${name}`);
      },
    );
```

7. Uploading and Synchronizing a Data Feed

Create and synchronize your product catalog to Dynamic Yield. For details, see Data Feeds.

8. Getting Started with Your First Campaign

Now that Dynamic Yield is fully implemented, you are all ready to get started by creating your first mobile campaign such as:

- Recommendations
- Variable Sets
- Dynamic Content
- Push Notifications

- Custom Actions

and much more! Be creative, experiment, and have fun!