Dynamic Yield Implementation Guide for Mobile Android Sites

Last updated April 10, 2019

Implementing Dynamic Yield for a mobile Android site can be broken down the following steps:

1. Installing the SDK Using Maven

In Maven, under your project gradle dependencies, add the following line:

```
compile ('com.dynamicyield:DYAPISDK:+'){ transitive = false;}
```

Use '+' to indicate the latest version, or specify the required SDK version (Example: '2.6.3')

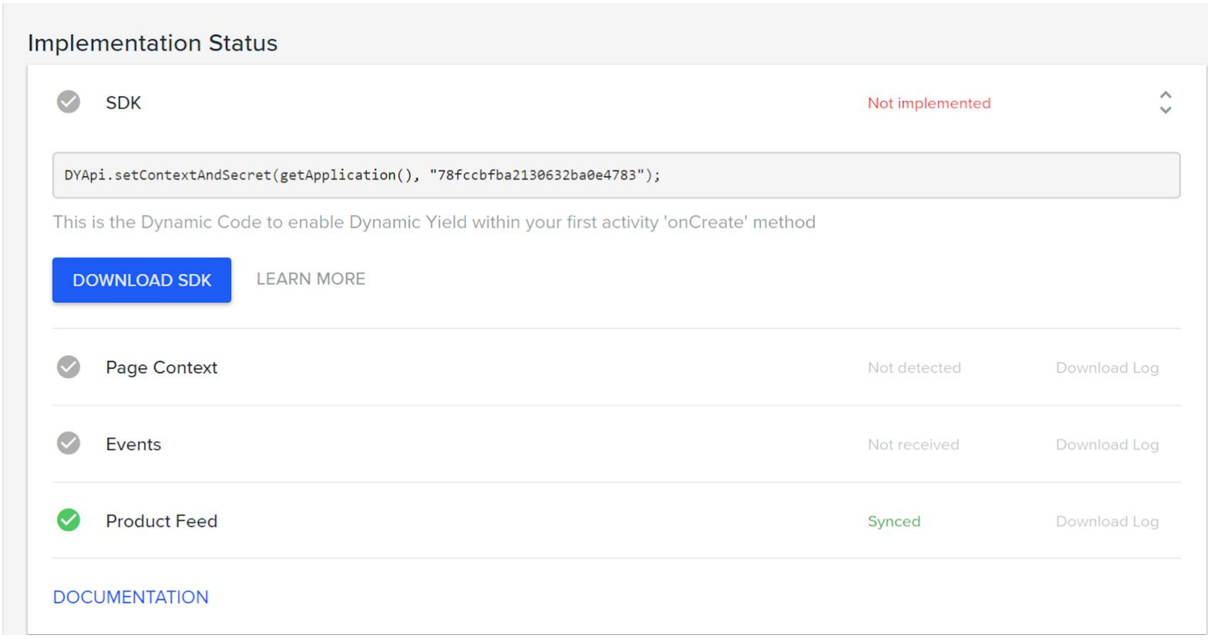
If you want to install the SDK manually instead of using Maven, see SDK Installation for Android.

2. Adding your Secret App Key

Your secret key identifies your application as corresponding to a specific Dynamic Yield site. You need to copy the key from Dynamic Yield and place it in your app.

1. Go to **Settings** › **General Settings**.

2. In the **Implementation Status** area, expand the SDK section by clicking the arrows.


Implementation Status

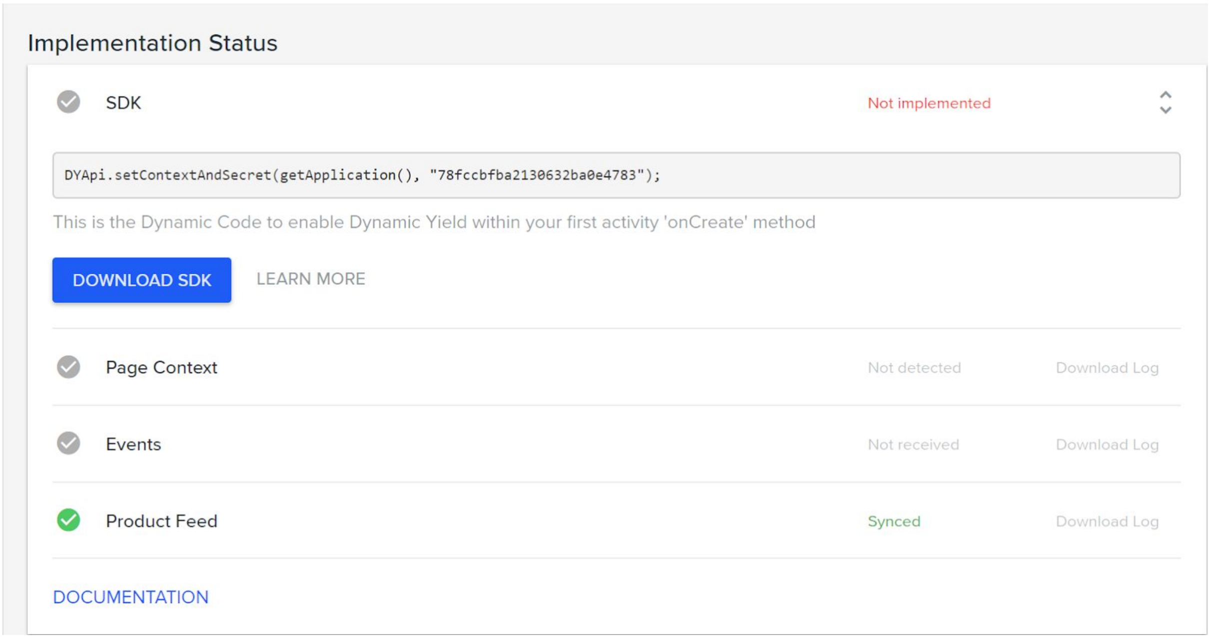
SDK — Not implemented

```
DYApi.setContextAndSecret(getApplication(), "78fccbfba2130632ba0e4783");
```

This is the Dynamic Code to enable Dynamic Yield within your first activity 'onCreate' method

Page Context	Not detected	Download Log
Events	Not received	Download Log
Product Feed	Synced	Download Log

3. The long letter/number combination in quotes is your secret key. Copy it and save it, you will need it later in this procedure.


Implementation Status

SDK — Not implemented

```
DYApi.setContextAndSecret(getApplication(), "78fccbfba2130632ba0e4783");
```

This is the Dynamic Code to enable Dynamic Yield within your first activity 'onCreate' method

Page Context	Not detected	Download Log
Events	Not received	Download Log
Product Feed	Synced	Download Log

4. In your Application class, initialize the object as seen in the following example code:

```
@Override
public void onCreate() {
    super.onCreate();

DYApi.setContextAndSecret(getApplicationContext(),YOUR_SECRET_KEY);
//replace with your secret key
    // Other init code
  }
```

3. Working with Proguard

DynamicYield SDK uses open source software which does not support Proguard, if your app is using Proguard, please add this to your build.gradle file:

```
proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.txt'
```

The content of proguard-rules.txt should be:

```
-keep class com.dynamicyield.** { *; }
-keeppackagenames com.dynamicyield.**
```

4. Enabling Communication with Dynamic Yield

Implement the DYListenerItf listener class to send data events back to your app by implementing the following API. This enables you to know if DY is ready to be activated.

```
public void experimentsReadyWithState(DYExperimentsState
dyExperimentsState)
```

Parameters

- *dyExperimentsState*: An enum holding the state with the following possible values:
    - DY_NOT_READY experiments are not ready to use
    - DY_READY_LOCAL_CACHE experiments loaded from local cache
    - DY_READY_AND_UPDATED experiments were updated from the server and are ready to use
    - DY_READY_NO_UPDATE_NEEDED experiments loaded from local cache and no other updates are needed

Example

```
@Override
public void experimentsReadyWithState(DYExperimentsState
dyExperimentsState) {
  if (dyExperimentsState == DYExperimentsState.DY_READY_NO_UPDATE_NEEDED
|| dyExperimentsState == DYExperimentsState.DY_READY_AND_UPDATED) {
  //start interacting with DY, e.g. ask for recommendations, track
pageviews, // events, etc.
  }
}
```

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 “ViewWillAppear” / "OnResume" with page context, or every “logical” page change done by the user (i.e a new view has been opened on current ViewController).

```
public boolean trackPageView(java.lang.String uniqueID, DYPageContext
context)
```

Parameters

- *uniqueId*: Unique identifier of the page as maintained in the application
- *context*: (Optional) The current screen’s context, for details see the table below

Example

```
DYApi.getInstance().trackPageView("main_shopping_cart", context);
```

Page Context Examples (Java)

Page Type	Required Attributes	Page Context Examples
Homepage	-	`DYApi.getInstance().trackPageView("Homepage",new DYPageContext("en_US",DYPageContext.HOMEPAGE,null));`

Category	Array of category names in hierarchical order (array of strings)	`JSONArray ctxData = new JSONArray();`
		`ctxData.put("clothes");`
		
		`DYApi.getInstance().trackPageView("category page",new`

		`DYPageContext("en_US",DYPageContext.CATEGORY,ctxData));`

Product	SKU (string)	`JSONArray ctxData = new JSONArray();`
		`ctxData.put("WT03");`
		
		`DYApi.getInstance().trackPageView("product page",new DYPageContext("en_US",DYPageContext.PRODUCT,ctxData));`

Cart	SKUs (array of strings)	`JSONArray ctxData = new JSONArray();`
		`ctxData.put("WT03");`
		
		`DYApi.getInstance().trackPageView("cart page",new DYPageContext("en_US",DYPageContext.CART,ctxData));`

Other (if page matches none of the above page types))	-	`DYApi.getInstance().trackPageView("page name",new DYPageContext("en_US",DYPageContext.OTHER,ctxData));`

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 site 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. The event must be triggered on a page that has the Dynamic Yield script in the header, so make sure this is true especially if the event is triggered on a “Thank You” page.

Java Example:

```java
//Populating the Prop parameter
JSONArray cart = new JSONArray();
JSONObject product = new JSONObject();

product.put("productId", PRODUCT_ID);
product.put("quantity", QUANTITY);
product.put("itemPrice", ITEM_PRICE);

cart.put(product);

JSONObject purchase = new JSONObject();

purchase.put("dyType", "purchase-v1");
purchase.put("value", CART_VALUE);
purchase.put("cart", cart);

//Sending the Event
DYApi.getInstance().trackEvent("purchase", purchase);
```

2. Add to Cart

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

Java example:

```java
//Populating the Prop parameter
JSONObject props = new JSONObject();

props.put( "value", ITEM_VALUE  );
props.put( "productId", PRODUCT_ID );
props.put( "quantity", QUANTITY );
props.put( "dyType","add-to-cart-v1" );


//Sending the Event
DYApi.getInstance().trackEvent("Add to Cart", props );
```

3. Login

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

Java example:

```
//Populating the Prop parameter
JSONObject login = new JSONObject();

login.put("hashedEmail", HASHED_EMAIL);

login.put("dyType", "login-v1");


//Sending the Event
DYApi.getInstance().trackEvent("login", login);
```

4. Signup

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

Java Example:

```
//Populating the Prop parameter
JSONObject Signup = new JSONObject();
Signup.put("dyType", "signup-v1");
Signup.put("hashedEmail", HASHED_EMAIL);


//Sending the Event
DYApi.getInstance().trackEvent("Signup", Signup);
```

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).

Example:

```
DYUserData* user = [DYUserData create];

//SHA256 hashed email of the plain text email in lower case
[user
setEmail:@"180123456789ABCDEF0123456789ABCDEF0123456789ABCDEFC54341
C3F27B79"];
[_DY identifyUser:user];
```

6. Remove from Cart (required for Triggered Emails & Push Notifications)

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.

Java Example:

```
//Populating the Prop parameter
JSONObject props = new JSONObject();

props.put( "value", ITEM_VALUE  );
props.put( "productId", PRODUCT_ID );
props.put( "quantity", QUANTITY );
props.put( "dyType","remove-from-cart-v1" );


//Sending the Event
DYApi.getInstance().trackEvent("Remove from Cart", props );
```

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.

Java Example (1 item in cart):

```
//Populating the Prop parameter
JSONArray cart = new JSONArray();
JSONObject product = new JSONObject();

product.put("productId", PRODUCT_ID);
product.put("quantity", QUANTITY);
product.put("itemPrice", ITEM_PRICE);

cart.put(product);

JSONObject purchase = new JSONObject();
```

```
purchase.put("dyType", "sync-cart-v1");
purchase.put("cart", cart);


//Sending the Event
DYApi.getInstance().trackEvent("Sync Cart", purchase);
```

Validate your events using the Implementation Helper (see below) and the Implementation Status dashboard.

7. Uploading and Synchronizing a Data Feed

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

8. Validating your Implementation

Use the Mobile Implementation Helper to verify your implementation of page context, events, and API calls.

1. Go to **Settings** › **General Settings**.
2. In the Implementation Helper section, click Launch Helper.
3. Connect your device to the admin console and browse in your app to ensure your implementation is correct.

9. 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!