Dynamic Yield Implementation Guide for Mobile iOS Sites

Last updated April 10, 2019

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

1. Installing the SDK Using Cocoapods

1. Configure your Xcode project to work with CocoaPods. Please refer to CocoaPods Getting Started for help.
2. Add the following line to your podfile. For more details, see Using CocoaPods.

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

3. Run pod install or pod update before continuing with the next step.

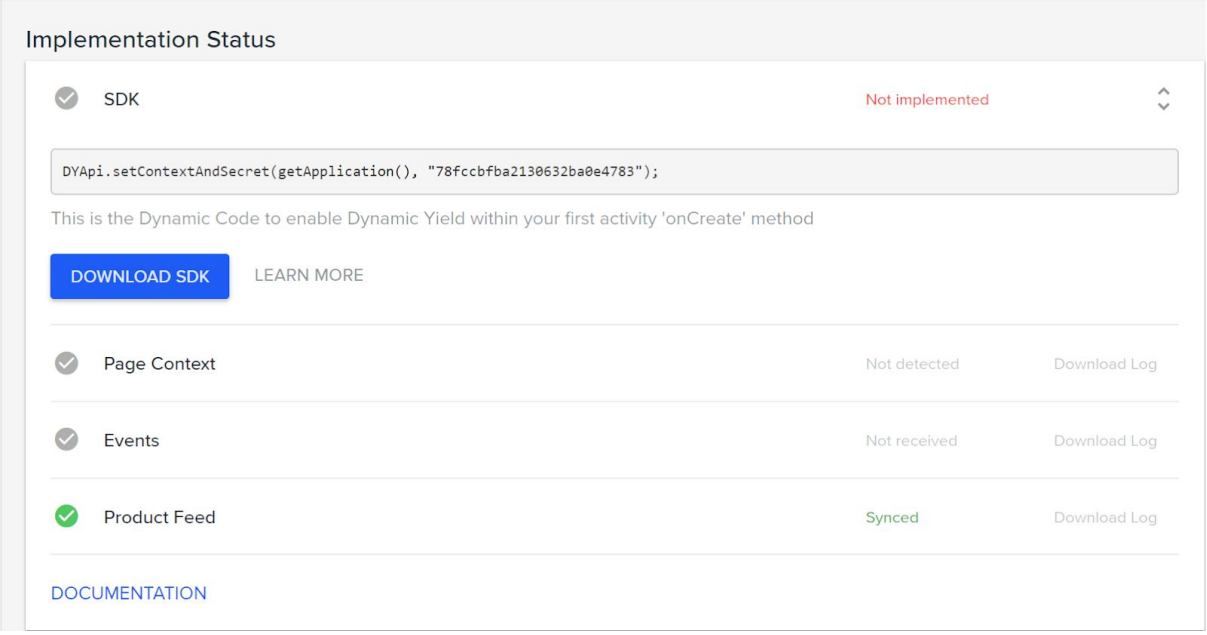
If you want to install the SDK manually instead of using Cocoapods, see SDK Installation for iOS.

2. Adding your Secret App Key

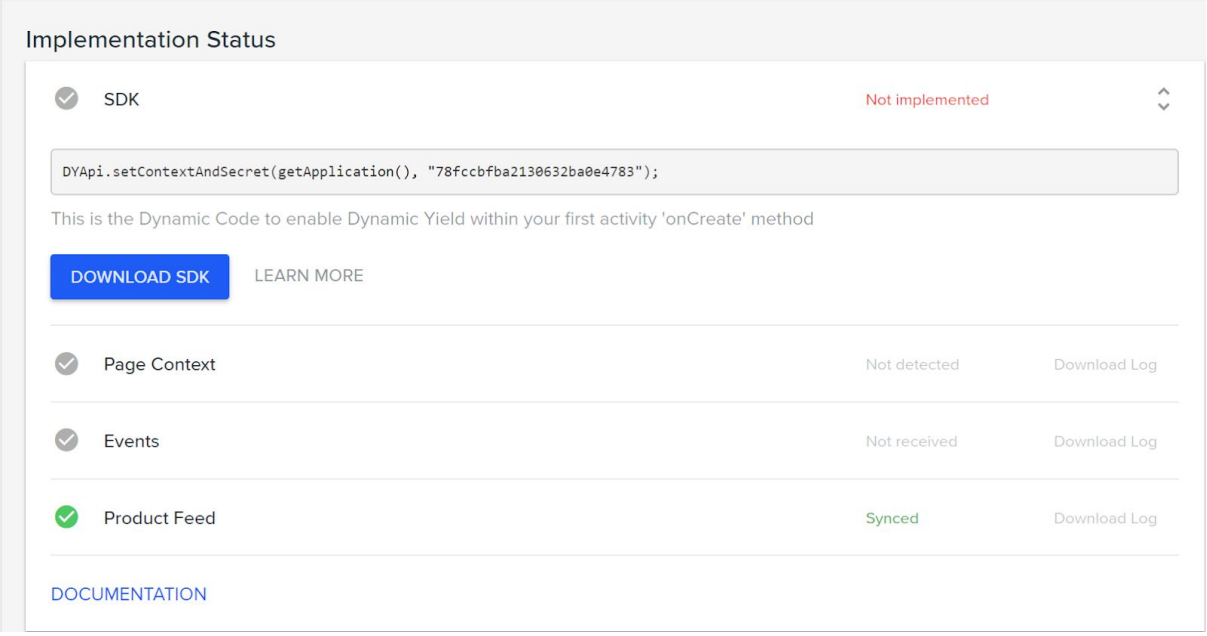
Your secret key identifies your application as corresponding to a specific Dynamic Yield site. You need to copy the key from Dynamic Yield and place it in your app.

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

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



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.



4. In your application AppDelegate's **didFinishLaunchingWithOptions:** method, import the Dynamic Yield header, initialize the object and set your app secret key as in the following examples:
Swift example:

```
import DYSDK
DYApi.getInstance().setSecretKey("YOUR_SECRET_KEY ");
```

Objective C example:

```
#import "DYApi.h"
[[DYApi getInstance] setSecretKey:@"YOUR_SECRET_KEY"];
```

dynamic yield

3. Enabling Communication with Dynamic Yield

Implement the DYDelegateProtocol 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.
  }
}
```

4. 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).

```
-(BOOL)pageView:(NSString*)uniqueId context:(DYPageContext*)DYContext;
```

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

Examples

```
//Objective-C
[[DYApi getInstance] pageView:@"pageUniqueID"  context:context];

//Swift
DYApi.getInstance().pageView("pageUniqueID", context:context);
```

Page Context Examples

Page Type	Required Attributes	Page Context Examples

Homepage	-	`//Objective-C`
		`DYPageContext* context = [[DYPageContext alloc] init];`
		`[context setLanguage:@"en_US"];//Optional`
		`[context setType:DY_TYPE_HOMEPAGE];`
		`[_DY pageView:@"ABCD" name:[self getPageName] context:context];`
		
		`//Swift`
		`let context = DYPageContext.init();`
		`context.language = "en_US";//Optional`
		`context.type = DY_TYPE_HOMEPAGE;`
		`DYApi.getInstance().pageView("pageUnique",name:"pageName", context:context);`

Category	Array of category names in hierarchical order (array of strings)	`//Objective-C`
		`DYPageContext* context = [[DYPageContext alloc] init];`
		`[context setType:DY_TYPE_CATEGORY];`
		`[context setData:@[@"Men's",@"Shoes"]];`
		`[_DY pageView:@"ABCD" name:[self getPageName] context:context];`
		
		`//Swift`
		`let context = DYPageContext.init();`
		`context.data = ["Men's","Shoes"];`

		`context.type = DY_TYPE_CATEGORY;` `DYApi.getInstance().pageView("pageUnique",name:"pageName", context:context);`

Product	SKU (string)	`//Objective-C`
		`DYPageContext* context = [[DYPageContext alloc] init];`
		`[context setType:DY_TYPE_PRODUCT];`
		`[context setData:@[@"101"]];`
		`[_DY pageView:@"ABCD" name:[self getPageName] context:context];`
		
		`//Swift`
		`let context = DYPageContext.init();`
		`context.data = ["101"];`
		`context.type = DY_TYPE_PRODUCT;`
		`DYApi.getInstance().pageView("pageUnique",name:"pageName", context:context);`

Cart	SKUs (array of strings)	`//Objective-C`
		`DYPageContext* context = [[DYPageContext alloc] init];`
		`[context setType:DY_TYPE_CART];`
		`[context setData:@[@"101", @"102"]];`
		`[_DY pageView:@"ABCD" name:[self getPageName] context:context];`
		
		`//Swift`
		`let context = DYPageContext.init();`
		`context.data = ["101", "102"];`
		`context.type = DY_TYPE_CART;`
		`DYApi.getInstance().pageView("pageUnique",name:"pageName", context:context);`

Other (if page matches none of the above page types))	-	`//Objective-C`
		`DYPageContext* context = [[DYPageContext alloc] init];`
		`[context setLanguage:@"en_US"];//Optional`
		`[context setType:DY_TYPE_OTHER];`
		`[_DY pageView:@"ABCD" name:[self getPageName] context:context];`
		
		`//Swift`
		`let context = DYPageContext.init();`
		`context.language = "en_US";//Optional`
		`context.type = DY_TYPE_OTHER;`

		`DYApi.getInstance().pageView("pageUnique",name:"pageName", context:context);`

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

5. 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.

Objective C Example

```
//Populating the Prop parameter
NSMutableArray* cart = [NSMutableArray array];
NSMutableDictionary* itemA = [NSMutableDictionary dictionary];

[itemA setObject:@"item-34454" forKey:@"productId"];
[itemA setObject:@1 forKey:@"quantity"];
[itemA setObject:@65.87 forKey:@"itemPrice"];

[cart addObject:itemA];
 NSMutableDictionary* itemB = [NSMutableDictionary dictionary];

[itemB setObject:@"sku-4324-bg" forKey:@"productId"];
[itemB setObject:@1 forKey:@"quantity"];
[itemB setObject:@12.34 forKey:@"itemPrice"];

[cart addObject:itemB]; //Add more items if needed

NSMutableDictionary* eventDictionary = [NSMutableDictionary dictionary];
[eventDictionary setObject:cart forKey:@"cart"];
[eventDictionary setObject:@90.55 forKey:@"value"];
[eventDictionary setObject:@"any supported currency code" forKey:@"currency"];
//if not using the default currency
[eventDictionary setObject:@"123456" forKey:@"uniqueTransactionId"];
[eventDictionary setObject:@"purchase-v1" forKey:@"dyType"];

//Sending the Event
```

```
[[DYApi getInstance] trackEvent:@"Purchase" prop:eventDictionary];
```

Swift Example

```
//Populating the Prop parameter
var productA = Dictionary<String, String>();
productA["productId"] = "sku-4324-bg"
productA["quantity"] = 1
productA["itemPrice"] = 12.34

var productB = Dictionary<String, String>();
productB["productId"] = ”item-34454”
productB["quantity"] = 1
productB["itemPrice"] = 65.87

var cart = Array<Dictionary<String, String>>();
cart.append(productA);  //Add more items if needed
cart.append(productB);

var propsDic = Dictionary<String,Any>();
propsDic[ "dyType"] = "purchase-v1";
propsDic[ "currency"] = ”any supported currency code”; //If not using the default currency
propsDic[ "value"] = 90.55;
propsDic[ "uniqueTransactionId"] = “12345”;
propsDic[ "cart"] = cart;

//Sending the Event
DYApi.getInstance().trackEvent("purchase", prop: propsDic)
```

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).

Objective C example:

```
//Populating the Prop parameter
NSMutableDictionary* eventDictionary = [NSMutableDictionary dictionary];

NSMutableArray* cart = [NSMutableArray array];

NSMutableDictionary* itemA = [NSMutableDictionary dictionary];

[itemA setObject:@"item-34454" forKey:@"productId"];//item that already exists in the cart
[itemA setObject:@1 forKey:@"quantity"];
[itemA setObject:@65.87 forKey:@"itemPrice"];
```

```
[cart addObject:itemA];


 NSMutableDictionary* itemB = [NSMutableDictionary dictionary]; //The item being added to the cart should be included in the cart and listed last
[itemB setObject:@"sku-4324-bg" forKey:@"productId"];
[itemB setObject:@1 forKey:@"quantity"];
[itemB setObject:@12.34 forKey:@"itemPrice"];

[cart addObject:itemB];

[eventDictionary setObject:@"any supported currency code" forKey:@"currency"];
//If you are not using the default currency
[eventDictionary setObject:@12.34 forKey:@"value"];//The item being added to the cart
[eventDictionary setObject:@"sku-4324-bg" forKey:@"productId"];//The item being added to the cart
[eventDictionary setObject:@1 forKey:@"quantity"];//The item being added to the cart
[eventDictionary setObject:cart forKey:@"cart"];
[eventDictionary setObject:@"add-to-cart-v1" forKey:@"dyType"]; // Identifies this event as Add to Cart. Do Not Change

//Sending the Event
[[DYApi getInstance] trackEvent:@"Add to Cart" prop:eventDictionary];
```

Swift Example:

```
//Populating the Prop parameter
var productA = Dictionary<String, String>();
productA["productId"] = "sku-4324-bg"
productA["quantity"] = 1
productA["itemPrice"] = 12.34

var productB = Dictionary<String, String>();
productB["productId"] = ”item-34454”
productB["quantity"] = 1
productB["itemPrice"] = 65.87

var cart = Array<Dictionary<String, String>>(); //The item being added to the cart should be included in the cart and listed last. In this case, productB is being added
cart.append(productA);
cart.append(productB);

var product = Dictionary<String, String>();

product["cart"] = cart;
product["productId"] = "item-34454" //The item being added to the cart
product["quantity"] = 1 //The item being added to the cart
product["value"] = 65.87 //The item being added to the cart
product["currency"] = ”any supported currency code”; //If not using default currency
```

```
product["dyType"] = "add-to-cart-v1" // Identifies this event as Add to Cart. Do Not Change

//Sending the Event
DYApi.getInstance().trackEvent("add to cart", prop: product)
```

3. Login

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

Objective C example:

```
//Populating the Prop parameter
NSMutableDictionary* eventDictionary = [NSMutableDictionary dictionary];

[eventDictionary setObject:”..” forKey:@"hashedEmail"]; // SHA256 encoding of the lowercase e-mail. Must be a string.
[eventDictionary setObject:@"login-v1" forKey:@"dyType"]; // Must be "login-v1” .

//Sending the Event
[[DYApi getInstance] trackEvent:@"login" prop:eventDictionary];
```

Swift example:

```
//Populating the Prop parameter
var propsDic = Dictionary<String,Any>();

propsDic[ "hashedEmail"] = “..”; // SHA256 encoding of the lowercase e-mail. Must be a string.
propsDic[ "dyType"] = "login-v1"; // Must be "login-v1” .

//Sending the Event
DYApi.getInstance().trackEvent("login", prop: propsDic)
```

4. Signup

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

Objective C Example

```
//Populating the Prop parameter
NSMutableDictionary* eventDictionary = [NSMutableDictionary dictionary];
[eventDictionary setObject:”..” forKey:@"hashedEmail"]; // SHA256 encoding of the lowercase e-mail. Must be a string.
```

```
[eventDictionary setObject:@"signup-v1" forKey:@"dyType"]; // Must be
"signup-v1"
[eventDictionary setObject:@{@"gender":@"M"} forKey:@"metadata"]; // Can be any
other metadata

//Sending the Event
[[DYApi getInstance] trackEvent:@"Signup" prop:eventDictionary];
```

Swift Example

```
//Populating the Prop parameter
var propsDic = Dictionary<String,Any>();
propsDic[ "hashedEmail"] = "..."; // SHA256 encoding of the lowercase e-mail. Must
be a string.
propsDic[ "dyType"] = "signup-v1"; // Must be "signup-v1"
propsDic["metadata"] = ["gender":"M"]  // Can be any other metadata

//Sending the Event
DYApi.getInstance().trackEvent("Signup", prop: propsDic)
```

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).

Objective C Example:

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

//SHA 256 hashed email of the plain text email in lower case
[user
setEmail:@"18B258E9ADE091623DD829132E1E9D8B4A7F7D90ABEA194001C54341
C3F27B79"];
[_DY identifyUser:user];
```

Swift Example:

```
let user = DYUserData.create();
user.email =
"18B258E9ADE091623DD829132E1E9D8B4A7F7D90ABEA194001C54341C3F27B79";
DYApi.getInstance().identifyUser(user);
```

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

dynamic yield

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.

Objective C Example

```
//Populating the Prop parameter
NSMutableDictionary* eventDictionary = [NSMutableDictionary dictionary];
NSMutableArray* cart = [NSMutableArray array];
NSMutableDictionary* itemA = [NSMutableDictionary dictionary];

[itemA setObject:@"item-34454" forKey:@"productId"];
[itemA setObject:@1 forKey:@"quantity"];
[itemA setObject:@65.87 forKey:@"itemPrice"];

[cart addObject:itemA];
 NSMutableDictionary* itemB = [NSMutableDictionary dictionary];

[itemB setObject:@"sku-4324-bg" forKey:@"productId"];
[itemB setObject:@1 forKey:@"quantity"];
[itemB setObject:@12.34 forKey:@"itemPrice"];

[cart addObject:itemB]; //Cart should not include the removed item

[eventDictionary setObject:@12.55 forKey:@"value"];//The item to remove
[eventDictionary setObject:"it-23-d" forKey:@"productId"];//The item to remove
[eventDictionary setObject:@1 forKey:@"quantity"]; //The item to remove

[eventDictionary setObject:cart forKey:@"cart"];
[eventDictionary setObject:@"remove-from-cart-v1" forKey:@"dyType"]; // Identifies
this event as Remove from Cart. Do Not Change

//Sending the Event
[[DYApi getInstance] trackEvent:@"Remove from Cart" prop:eventDictionary];
```

Swift Example

```
//Populating the Prop parameter
var product = Dictionary<String, String>();

var productA = Dictionary<String, String>();
productA["productId"] = "sku-4324-bg"
productA["quantity"] = "1"
productA["itemPrice"] = "12.34"

var productB = Dictionary<String, String>();
productB["productId"] = "item-34454"
productB["quantity"] = "1"
productB["itemPrice"] = "65.87"

var cart = Array<Dictionary<String, String>>(); //The cart state after the item
has been removed
```

```
cart.append(productA);
cart.append(productB);

product["cart"] = cart;
product["productId"] = “id-0d8” //The item to remove
product["quantity"] = “1” //The item to remove
product["value"] = “12.55” //The item to remove
product["dyType"] = "remove-from-cart-v1" // Identifies this event as Remove from Cart. Do Not Change

//Sending the Event
DYApi.getInstance().trackEvent("remove from cart", prop: product)
```

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.

Objective C Example (1 item in cart):

```
//Populating the Prop parameter
NSMutableArray* cart = [NSMutableArray array];
NSMutableDictionary* item = [NSMutableDictionary dictionary];

[itemA setObject:”id-a8r-a” forKey:@"productId"];
[itemA setObject:@1 forKey:@"quantity"];
[itemA setObject:@12.55 forKey:@"itemPrice"];

[cart addObject:itemA]; //Add more items if needed

NSMutableDictionary* eventDictionary = [NSMutableDictionary dictionary];
[eventDictionary setObject:cart forKey:@"cart"];
[eventDictionary setObject:@”any supported currency code” forKey:@"currency"];
[eventDictionary setObject:@"sync-cart-v1" forKey:@"dyType"]; //Identifies this event as Sync Cart. Do Not Change


//Sending the Event
[[DYApi getInstance] trackEvent:@"Sync Cart" prop:eventDictionary];
```

Swift Example (1 item in cart):

```
//Populating the Prop parameter
var product = Dictionary<String, String>();
productA["productId"] = “id-a8r-a”
productA["quantity"] = “1”
productA["itemPrice"] = “12.55”
```

```
var cart = Array<Dictionary<String, String>>();
cart.append(productA);  //Add more items if needed

var propsDic = Dictionary<String,Any>();
propsDic[ "dyType"] = "sync-cart-v1"; //Identifies this event as Sync Cart. Do Not
Change
propsDic[ "uniqueTransactionId"] = “12345”;
propsDic[ "cart"] = cart;


//Sending the Event
DYApi.getInstance().trackEvent("Sync Cart", prop: propsDic)
```

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

6. Uploading and Synchronizing a Data Feed

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

7. 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.

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!