Dynamic Yield Implementation Guide for eCommerce Sites

Last updated April 04, 2020

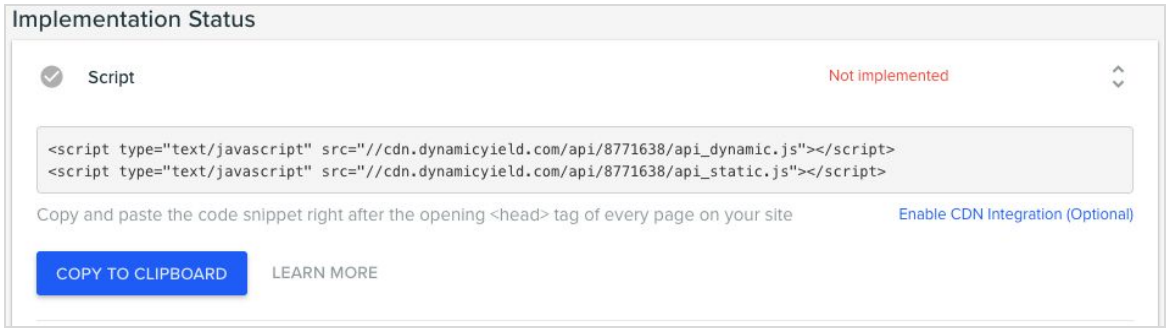
This guide provides the basic workflow for implementing Dynamic Yield on your eCommerce website, broken down into a required 4-step process:

1. Placing a Script on your site to enable communication with Dynamic Yield.
2. Adding page type descriptions called Page Context to each page on your site.
3. Configuring required Events to report user behaviors such as purchases and logins.
4. Uploading and synchronizing your Product Feed.
5. Configuring your application server to serve DYID cookies to support Safari users.

Dynamic Yield Script

To enable communication with Dynamic Yield, allow us to gather your users' online activity and render experiences, begin by placing the Dynamic Yield script on each page of your site.

1. In the Dynamic Yield admin, go to **Settings** › **General Settings** and copy the code.



2. Paste the code snippet synchronously, immediately after the opening <head> tag of every page on your site. Common implementation mistakes to avoid:

    - Implementing the script within the <BODY>
    - Appending to the <HEAD> from the <BODY>
    - Inserting Dynamic Yield Scripts using an external script
    - Implementing the script using a Tag Manager

3. The implementation status dashboard will indicate when the script was properly implemented. Learn more about Validating your Dynamic Yield Script.

Learn more about the Dynamic Yield Script.

Page Context

In order for Dynamic Yield to deliver the right experiences and gather accurate behavioral data, place the a Page Context variable for every page on your site before the Dynamic Yield script in the <head> tag of your site:

> **Note:** In areas that your site has single page application functionality, you need to report pageviews and their page context using a dedicated API. For details, see Single Page Application Support.

Page Type	Description	Example
Homepage		`DY.recommendationContext = {type: "HOMEPAGE"};`
Category Page	Full hierarchy of category names. Keep the same format and hierarchy as reported in the data feed.	`DY.recommendationContext = {type:"CATEGORY", data:["TOP_LEVEL_CAT", "CHILD_CAT","GRANDCHILD_CAT" ]};`
Product Page	SKU (formatted as a string). Use the exact	`DYD .recommendationContext =`

	same SKU as reported in events and data feed.	`{type:"PRODUCT", data: ["SKU123"]};`
Cart	SKUs (formatted as a list of strings). Use the exact same SKU as reported in events and data feed.	`DY.recommendationContext = {type:"CART", data: ["SKU123","SKU234"]};`
Other	Use 'other' when your page type does not match any pre-defined page types.	`DY.recommendationContext = {type: "OTHER"};`

Multi local support example:

For multi-locale support, add the field "lng" with the relevant locale, for instance "en_GB"; the locale syntax can be any string, but it must match the column translations used in the data feed.

```
DY.recommendationContext = {type: "HOMEPAGE", lng:"en_GB"};
```

Page Context implementation example:

The header of each page should now include both the script and the page context as seen in this example:

```
<script type = "text/javascript" >
  window.DY = window.DY || {};
DY.recommendationContext = {
  type: "PRODUCT",
  data: ["SKU123"],
  lng: "en_GB"
}; // this is an example of a product page
</script>
<link rel="dns-prefetch" href="//cdn.dynamicyield.com" />
<link rel="dns-prefetch" href="//st.dynamicyield.com" />
<link rel="dns-prefetch" href="//rcom.dynamicyield.com" />

<script src = "//cdn.dynamicyield.com/api/[SECTION_ID]/api_dynamic.js"
type = "text/javascript" > < /script> <
  script src = "//cdn.dynamicyield.com/api/[SECTION_ID]/api_static.js"
type = "text/javascript" > < /script>
```

To validate your implementation of page context, see Validating Page Context.

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.

> **Note:** Make sure each event is only fired once, and only when the event properties are available (consider use cases of page refresh, back/forward on browser, etc.)

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.

Purchase event with two items example:

```
DY.API("event", {
  name: "Purchase", // Human-readable name, not used to identify an event type
  properties: {
    uniqueTransactionId: "123456", // Optional. Will ensure only one purchase is
reported per transaction. Must be a string. Max 64 characters.
    dyType: "purchase-v1", // Identifies this event as a purchase. Do Not Change.
    value: 90.55, // Total paid value (after discounts/promotions). Must be positive.
    currency: "any supported currency code", // If not using default currency,
specify the currency here.
    cart: [
      // itemPrice is per quantity of one in float format (dollars.cents)
      {
        productId: "item-34454", //SKU exactly as in the product feed and context.
```

```
        quantity: 1,
        itemPrice: 65.87
      }, {
        productId: "sku-4324-bg",
        quantity: 2,
        itemPrice: 12.34
      }
    ]
  }
});
```

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).

Add to cart event (when there is already one item in a cart) example:

```
DY.API("event", {
  name: "Add to Cart", // Human-readable name, not used to identify an event type
  properties: {
    dyType: "add-to-cart-v1", // Identifies this event as Add to Cart. Do Not Change
    value: 34.45, // The actual payment value. Must be positive
    currency: "any supported currency code", // If not using default currency,
specify the currency here.
    productId: "item-34454", // SKU exactly as in the product feed and context.
    quantity: 1,
    cart: [ // The absolute current cart state including the item currently being
added. Products should be in order from oldest to newest.
      {
        productId: "sku-4324-bg",
        quantity: 2,
        itemPrice: 12.34,
      },
      {
        productId: "item-34454",
        quantity: 1,
        itemPrice: 34.45
        size: "XL" // optional string, can be any value
      }
    ]
  }
```

```
});
```

3. Login

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

Login event example:

```
DY.API("event", {
  name: "Login", // Human-readable name, not used to identify an event type
  properties: {
    dyType: "login-v1", // Must be "login-v1″ .
    hashedEmail: "..." // SHA256 encoding of the lowercase e-mail. Must be a string.
  }
});
```

4. Signup

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

Signup event (with gender metadata) example:

```
DY.API("event", {
  name: "Signup", // Human-readable name, not used to identify an event type
  properties: {
    dyType: "signup-v1", // Must be "signup-v1″
    hashedEmail: "...", // SHA256 encoding of the lowercase e-mail. Must be a string.
    metadata: {
      gender: "M" // Can be any other metadata
    }
  }
});
```

5. 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.

Remove from cart event example:

```
DY.API("event", {
  name: "Remove from Cart",// Human-readable name, not used to identify an event type
  properties: {
    dyType: "remove-from-cart-v1",// Identifies this event as Remove from Cart. Do
Not Change
    value: 34.45, // The actual payment value. Must be positive
    currency: "any supported currency code", // If not using default currency,
specify the currency here.
    productId: "gswefd-34-454",
    quantity: 1,
    size: "XL" // optional
    cart: [{ // The absolute current cart state (including the last added item),
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
      }
    ]
  }
});
```

6. Sync Cart (required for Triggered Emails)

Trigger this event when the user makes any changes to their cart such as increasing the quantity of an item or changing the size. This ensures that any emails triggered based on the purchase have up to date information about the user's cart.

Sync cart event example:

```
DY.API("event", {
  name: "Sync cart",// Human-readable name, not used to identify an event type
  properties: {
    dyType: "sync-cart-v1",// Identifies this event as Sync Cart. Do Not Change
    currency: "any supported currency code", // If not using default currency,
specify the currency here.
```

```
    cart: [{ // The absolute current cart state (including the last added item),
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
      }
    ]
  }
});
```

Validate your events using the Implementation Helper and the Implementation Status dashboard. For details, see Validating Events.

Product Feed

Product feeds enable powerful features such as recommendations and social proof messaging. Make sure the following columns are included with the appropriate values. Once you have a feed that meets these requirements, upload and synchronize your feed to Dynamic Yield. For details, see Data Feeds.

> **Note:** When your feed does not match these requirements, you may be able to use a parsing function to modify it. For more details, see Advanced Feed Configuration.

Columns	Description	Format	Example
sku	Product Unique Identifier. Note: SKUs cannot contain spaces	String (no spaces)	“111”

group_id	Identifies a group of products that differ in some product attributes. Note: This value cannot contain spaces.	String (no spaces)	“22”
name	Product name	String	“t_shirt”
url	URL to the product details page (must be a valid URL, starting with HTTP/HTTPS)	URL, string representation	"https://www.dy.com/shop/111.html?ch=22"
price	Price of the product (must be a number)	Float	9.50
in_stock	Indication whether the product is in stock (noted by ‘true/false’)	Boolean	true
image_url	The URL to the product image (must be a valid URL, starting with HTTP/HTTPS)	URL, string representation	"http://dy.com/shop/111/ex.jpg"
categories	The categories associated with the product from general to specific	Strings separated by pipes with no spaces	“Women’s|Trousers & Jeans|Chinos”
Keywords	Optional,but strongly recommended. Any additional information describing the product, separated by pipes. Used for our machine	String	“sale_item”

	learning and affinity algorithms.		
Custom columns	Optional. Column name can be any string. You can add up to 30 additional columns to the feed.	String	“custom_column_value”
Column Translations	To support multiple languages, you can specify different values for different languages for any field. Add an additional row and use the format “lng:<language name>:<column name>”: “<value>” The language name should match the name you use in the page context. For more details, see Multi-Language Support.	String	“lng:en_EN:name":" White Pants” “lng:de_DE:name":"weiße Hosen”

Product Feed Example (XML):

```
<items>
 <item>
   <sku>111</sku>
   <group_id>22</group_id>
   <price>9.5</price>
   <categories>mens|sport</categories>
   <in_stock>true</in_stock>
   <name>t_shirt</name>
```

```
   <image_url>http://dy.com/shop/111/ex.jpg</image_url>
   <url>https://www.dy.com/shop/111.html?ch=22</url>
 </item>
</items>
```

Safari User Support

Apple's Safari includes an enhancement to Webkit's Internet Tracking Prevention (ITP) policy, published on March 24th, 2020. The new ITP policy forces a 7-day expiration for all script-writable storage. To allow the continuous valid functionality of Dynamic Yield for users visiting your site via Safari,the DYID cookie **should be set by the backend application serving your website on your domain**. Cookies set this way are not affected by ITP and can have a longer expiration.

1. When a user visits your site, a request is sent from the browser to the backend application serving your site. For returning users, the request includes the **_dyid** cookie (together with all other first-party cookies).
2. When receiving this request, within your backend application, duplicate the **_dyid** cookie value into a new cookie. Use the key **_dyid_server** and return the new cookie as a response header for this request, setting a 1-year expiration date. This will set the **_dyid_server** cookie as a server-side first-party cookie in the user's browser which, as recommended by the ITP policy, is not affected by the cookie expiration enforcement.
3. Our script will use the **_dyid_server** cookie as it runs on your page and as a result, no returning-user data will be lost, even if the user has not visited your site for over 7 days.

Additional Resources and Validation

For more details about any of the components of implementing Dynamic Yield, see the following articles:

- Dynamic Yield Script
- Page Context for Web
- Events
- Data Feeds

It is important to validate your implementation using the Implementation Status Dashboard and the Implementation Helper. For details, see Validating Web Implementations.