

Create an Image-Based Badging Action

The product badging feature allows clients to apply some treatment to specific items on product listing pages and product detail pages to draw site visitors' attention to those items to foster engagement. Product badging highlights key attributes of items that might not otherwise jump out to visitors navigating products on the site.

A product badge frequently appears as a digital promotional sticker displayed atop the product image to indicate that the item is a bestseller or a top-rated product. These labels serve as customer endorsements, allowing shoppers to more quickly locate popular products on a page.

You can set up a handler for Omnichannel badging actions using two methods. The `addEvents` method defines the events that can trigger the action. The `getActionsData` method is then used as the trigger and requests the experience based on the defined events. `getActionsData` then returns a JSON object containing badging image data that you can then handle in code.

Prerequisites

You must first create an Omnichannel badging action within Monetate for the methods to reference. Refer to [Configure an Omnichannel Product Badging Action](#) for instructions.

This action uses Product Thumbnail View as an event listener. To accomplish this, you must set **Thumbnail's product ID** = as one of the conditions. Make note of this product ID so that you can pass the appropriate ID in code.

Optionally, you can add other conditions to this action. If you do, make note of those so that you can pass those as relevant events.

addEvents

This method records a local event in the defined context.

```
addEvent(context: <ContextEnum>, event: <MEvent?>)
```

Parameters:

- `context` is name of the event. (Required)
- `event` is the event data. (Required)

For an Omnichannel Badging action, use the `ContextProductThumbnailView` event as your context. Pass a list of product IDs that you want to set up a badge for as your events data.

Code Example

```
Personalization.shared.addEvent(context: .ProductThumbnailView, event: ProductThumbnailView(p
```

If you added other conditions to your Omnichannel Badging action, use additional calls of this method to handle

them with the appropriate events and data.

getActionsData

This method sends the defined events to Monetate to trigger an experience. If the events fulfill the WHO settings of an experience, then that experience is triggered. A JSON object containing the experience response is then returned.

```
getActionsData(requestId: <String>, arrActionTypes: <[ActionTypeEnum]>)
```

Parameters:

- `requestID` is the request ID for the API.
- `actionType` is the type of action you want to request. You can specify multiple actions in an array to handle. (Required)

Use `OmniChannellImageBadging` as the action type for this method.

Code Example

```
Personalization.shared.getActionsData(  
    requestId: "test_request_id", arrActionTypes: [.OmniImageBadging]  
)  
.on { (res) in  
    if res.status == 200 {  
        let data = JSON(res.data)  
        self.handleBadging(res: res)  
        print(data)  
    } else {  
    }  
}
```

Full Code Example

Complete code example blocks are listed below.

```
// Add the required events used in experience.
```

```
Personalization.shared.addEvent(  
    context: .ProductThumbnailView,  
    event: ProductThumbnailView(  
        products: Set(["BackP_001", "BackP_002", "BackP_003", "BackP_004", "BackP_005", "BackP_006"
```

```
// -----  
// Fetch badging experience data
```

```
Personalization.shared.getActionsData(
    requestId: "test_request_id", arrActionTypes: [.OmniImageBadging]
).on { (res) in
    if res.status == 200 {
        let data = JSON(res.data)
        self.handleBadging(res: res)
        print(data)
    } else {
    }
}
```

```
// -----
// Handle badging experience response
```

```
private func handleBadging(res: APIResponse) {
    let data = JSON(res.data!)
    for item in data["data"]["responses"].arrayValue {
        if item["requestId"].string == res.requestId {
            for oneAction in item["actions"].arrayValue {
                if oneAction["actionType"].string == "monetate:action:OmniChannellImageBadging" {
                    if let plds = oneAction["pids"].array {
                        arrBadgedProductIds = oneAction["pids"].rawValue as! [String]
                    }
                    if !arrBadgedProductIds.isEmpty {
                        let imageContent = oneAction["image_content"]
                        let imageURL = URL(string: imageContent["ref"].stringValue)
                        if imageURL != nil {
                            downloadImage(from: imageURL!)
                        }
                    }
                    return
                }
            }
        }
    }
}
```

Response Data

An example of the JSON data returned from `getActionsData` is below. Handle this response in your code accordingly to render the data in your UI as you see fit.

```
{
  "meta":
  {
    "code": 200
  },
  "data":
  {
    "responses":
    [
      {
        "requestId": "sample-unique-request-id-ABC-123",
        "actions":
        [
          {
            "image_content":
            {
              "version": 2,
              "clickzones": [],
              "contentId": 785039,
              "top": 0,
              "title": "",
              "href": "",
              "iwidth": 75,
              "alt": "",
              "iheight": 100,
              "ref": "https://sb.monetate.net/img/1/1190/4919639.png",
              "left": 180
            },
            "actionType": "monetate:action:OmniChannelImageBadging",
            "actionId": 4959735,
            "application_data":
            {
              "color": "#FFFFFF",
              "fontSize": 10,
              "style": "bold",
              "badge": "Save 15% with 'SAVE15'",
              "background-color": "#2A519E"
            },
            "positioning_hint": "top-left",
            "pids":
            [
              "BackP_009",
```

"BackP_010"

}

}

}

}

}

}

}