Create Social Proof Actions

Social proof is a demonstration that other people have purchased or found value in a product or service. This demonstration helps the product or service stand out from others and makes the customer more likely to purchase it. These demonstrations often reflect existing customer behavior. For example, labelling a product as a best seller is a social proof demonstration.

You can set up a handler for Omnichannel Social Proof 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 decision based on the defined events. `getActionsData` then returns a JSON object containing social proof data that you can then handle in code.

addEvents

This method records a local event.

```
public void addEvent(String context, Object event)
```

Parameters:

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

You can use this method multiple times to add all the necessary events for an experience you want to trigger. The example code uses multiple method calls to fulfill the experience requirements.

```
IpAddress ipAddress = new IpAddress();
ipAddress.setIpAddress("10.0.0.2");

ScreenSize screenSize = new ScreenSize();
screenSize.setHeight(23);
screenSize.setWidth(34);

PageView pageView = new PageView();
pageView.setPageType("Cartpage");
pageView.setUrl("https://www.monetate.com");

// addEvent
personalization.addEvent(EventTypes.ContextScreenSize, screenSize);
personalization.addEvent(EventTypes.ContextIpAddress, ipAddress);
personalization.addEvent(EventTypes.ContextPageView, pageView);
```

getActionsData

This method sends the defined events to Monetate to trigger an experience. If the events fulfill the WHO settings and the other required conditions of an experience, that experience is triggered. A JSON object containing the

experience response is then returned.

```
public String getActionsData(String[] actionTypes)
```

Parameters:

- `actionTypes` is the type of action you want to request. You can specify one action or multiple actions in an array to handle. (Required)

```
responseData = personalization.getActionsData(ActionTypes.OmniChannelJson);
```

Code Samples

When writing your code, you must handle the getActions method in a Thread or AsyncTask. To update the main thread, you can use either Handler or runOnUiThread. The code examples below use Thread and Handler to update the main thread and change the UI.

Viewed Product Code Example

This Social Proof type displays the most viewed products.

```
Personalization personalization = new personalization(user, account);

// Events/context data
Product product = new Product();
product.setProductId("97");
product.setSku("SKU");
ProductDetailView productDetailView = new ProductDetailView();
productDetailView.setProducts(new Product[]
{
  product
});

PageView pageView = new PageView();
pageView.setPageType("Cartpage");
pageView.setUrl("https://www.monetate.com");

// addEvent
personalization.addEvent(EventTypes.ContextProductDetailView, productDetailView);
personalization.addEvent(EventTypes.ContextPageView, pageView);

// getActionsData
String responseData;
new Thread(new Runnable()
{

  @Override
  public void run()
```

```
public void run()
 {

  // Get the responseData using getActionsData
  responseData = personalization.getActionsData(ActionTypes.OmniSocialProofData);

  // Once you receive the responseData from getActionsData, use the Handler and parse the requi
  new Handler(Looper.getMainLooper()).post(new Runnable()
  {
    @Override
    public void run()
    {

      // Parse the received responseData, get the requiredData from it and use it accordingly
      try
      {
        // Example: textView.setText(requiredData);
      }
      catch (Exception ex)
      {
        // Catch any exception that occured while parsing
        ex.printStackTrace();
      }

    }
  });
 }
}).start();
```

Carted Product Code Example

This Social Proof type displays the products most frequently added to a cart. You can use it for a single product or multiple products. Single products requires the `ContextProductDetailView` event to be added. Multiple products requires the `ProductThumbnailView` event to be added.

Single Product

```
Product product = new Product();
product.setProductId("97");
product.setSku("SKU");

ProductDetailView productDetailView = new ProductDetailView();
productDetailView.setProducts(new Product[]
{
 product
});
```

```
PageView pageView = new PageView();
pageView.setPageType("Cartpage");
pageView.setUrl("https://www.monetate.com");

// addEvent
personalization.addEvent(EventTypes.ContextProductDetailView, productDetailView);
personalization.addEvent(EventTypes.ContextPageView, pageView);

// getActionsData
String responseData;
new Thread(new Runnable()
{

  @Override
  public void run()
  {

    // Get the responseData using getActionsData
    responseData = personalization.getActionsData(ActionTypes.OmniSocialProofData);

    // Once you receive the responseData from getActionsData, use the Handler parse the required d
    new Handler(Looper.getMainLooper()).post(new Runnable()
    {
      @Override
      public void run()
      {

        // Parse the received responseData, get the requiredData from it and use it accordingly
        try
        {
          // Example: textView.setText(requiredData);
        }
        catch (Exception ex)
        {
          // Catch any exception that occured while parsing
          ex.printStackTrace();
        }

      }
    });
  }
}).start();
```

Multiple Products

```
// Events/context data
ProductThumbnailView productThumbnailView = new ProductThumbnailView();
productThumbnailView.setProducts(new String[] {"104","94","97"});

PageView pageView = new PageView();
pageView.setPageType("Cartpage");
pageView.setUrl("https://www.monetate.com");

// addEvent
personalization.addEvent(EventTypes.ContextProductThumbnailView, productThumbnailView);
personalization.addEvent(EventTypes.ContextPageView, pageView);

// getActionsData
String responseData;
new Thread(new Runnable()
{

  @Override
  public void run()
  {

    // Get the responseData using getActionsData
    responseData = personalization.getActionsData(ActionTypes.OmniSocialProofData);

    // Once you receive the responseData from getActionsData, use the Handler parse the required d
    new Handler(Looper.getMainLooper()).post(new Runnable()
    {
      @Override
      public void run()
      {

        // Parse the received responseData, get the requiredData from it and use it accordingly
        try
        {
          // Example: textView.setText(requiredData);
        }
        catch (Exception ex)
        {
          // Catch any exception that occured while parsing
          ex.printStackTrace();
        }

      }
    });
  }
}).start();
```

Purchased Product Code Example

This Social Proof type displays the products most frequently purchased. You can use it for a single product or multiple products. Single products requires the `ContextProductDetailView` event to be added. Multiple products requires the `ProductThumbnailView` event to be added.

Single Product

```
Product product1 = new Product();
product1.setProductId("97");
product1.setSku("SKU");

ProductDetailView productDetailView = new ProductDetailView();
productDetailView.setProducts(new Product[]
{
  product1
});

PageView pageView = new PageView();
pageView.setPageType("Cartpage");
pageView.setUrl("https://www.monetate.com");

// addEvent
personalization.addEvent(EventTypes.ContextProductDetailView, productDetailView);
personalization.addEvent(EventTypes.ContextPageView, pageView);

// getActionsData
String responseData;
new Thread(new Runnable()
{

  @Override
  public void run()
  {

    // Get the responseData using getActionsData
    responseData = personalization.getActionsData(ActionTypes.OmniSocialProofData);

    // Once you receive the responseData from getActionsData, use the Handler parse the required d
    new Handler(Looper.getMainLooper()).post(new Runnable()
    {
      @Override
      public void run()
      {

        // Parse the received responseData, get the requiredData from it and use it accordingly
        try
```

```
      {
        // Example: textView.setText(requiredData);
      }
      catch (Exception ex)
      {
        // Catch any exception that occured while parsing
        ex.printStackTrace();
      }

    }
  });
 }
}).start();
```

Multiple Products

```
// Events/context data
ProductThumbnailView productThumbnailView = new ProductThumbnailView();
productThumbnailView.setProducts(new String[] {"104","94","97"});

PageView pageView = new PageView();
pageView.setPageType("Cartpage");
pageView.setUrl("https://www.monetate.com");

// addEvent
personalization.addEvent(EventTypes.ContextProductThumbnailView, productThumbnailView);
personalization.addEvent(EventTypes.ContextPageView, pageView);

// getActionsData
String responseData;
new Thread(new Runnable()
{

  @Override
  public void run()
  {

    // Get the responseData using getActionsData
    responseData = personalization.getActionsData(ActionTypes.OmniSocialProofData);

    // Once you receive the responseData from getActionsData, use the Handler parse the required d
    new Handler(Looper.getMainLooper()).post(new Runnable()
    {
      @Override
      public void run()
      {

        // Parse the received responseData, get the requiredData from it and use it accordingly
        try
        {
          // Example: textView.setText(requiredData);
        }
        catch (Exception ex)
        {
          // Catch any exception that occured while parsing
          ex.printStackTrace();
        }

      }
    });
  }
}).start();
```

Recommended Product Code Example

This Social Proof type uses a recommendation strategy to determine and display a product the customer is most likely to purchase. It requires the `ContextProductDetailView` event to be added.

```
Product product1 = new Product();
product1.setProductId("97");
product1.setSku("SKU");

ProductDetailView productDetailView = new ProductDetailView();
productDetailView.setProducts(new Product[]
{
  product1
});

PageView pageView = new PageView();
pageView.setPageType("Cartpage");
pageView.setUrl("https://www.monetate.com");

// addEvent
personalization.addEvent(EventTypes.ContextProductDetailView, productDetailView);
personalization.addEvent(EventTypes.ContextPageView, pageView);

// getActionsData
String responseData;
new Thread(new Runnable()
{

  @Override
  public void run()
  {

    // Get the responseData using getActionsData
    responseData = personalization.getActionsData(ActionTypes.OmniSocialProofData);

    // Once you receive the responseData from getActionsData, use the Handler parse the required d
    new Handler(Looper.getMainLooper()).post(new Runnable()
    {
      @Override
      public void run()
      {

        // Parse the received responseData, get the requiredData from it and use it accordingly
        try
        {
          // Example: textView.setText(requiredData);
        }
```

```
      catch (Exception ex)
      {
        // Catch any exception that occured while parsing
        ex.printStackTrace();
      }

    }
  });
 }
}).start();
```

Inventory Code Example

This Social Proof type displays products based on your current inventory, which may indicate a popular product due to limited inventory count. You can use it for a single product or multiple products. Single products requires the `ContextProductDetailView` event to be added. Multiple products requires the `ProductThumbnailView` event to be added.

Single Product

```
Product product1 = new Product();
product1.setProductId("97");
product1.setSku("SKU");

ProductDetailView productDetailView = new ProductDetailView();
productDetailView.setProducts(new Product[]
{
 product1
});

PageView pageView = new PageView();
pageView.setPageType("Cartpage");
pageView.setUrl("https://www.monetate.com");

// addEvent
personalization.addEvent(EventTypes.ContextProductDetailView, productDetailView);
personalization.addEvent(EventTypes.ContextPageView, pageView);

// getActionsData
String responseData;
new Thread(new Runnable()
{

 @Override
 public void run()
 {
```

```
 {

  // Get the responseData using getActionsData
  responseData = personalization.getActionsData(ActionTypes.OmniSocialProofData);

  // Once you receive the responseData from getActionsData, use the Handler parse the required d
  new Handler(Looper.getMainLooper()).post(new Runnable()
  {
    @Override
    public void run()
    {

      // Parse the received responseData, get the requiredData from it and use it accordingly
      try
      {
        // Example: textView.setText(requiredData);
      }
      catch (Exception ex)
      {
        // Catch any exception that occured while parsing
        ex.printStackTrace();
      }

    }
  });
 }
}).start();
```

Multiple Products

```
ProductThumbnailView productThumbnailView = new ProductThumbnailView();
productThumbnailView.setProducts(new String[] {"104","94","97"});

PageView pageView = new PageView();
pageView.setPageType("Cartpage");
pageView.setUrl("https://www.monetate.com");

// addEvent
personalization.addEvent(EventTypes.ContextProductThumbnailView, productThumbnailView);
personalization.addEvent(EventTypes.ContextPageView, pageView);

// getActionsData
String responseData;
new Thread(new Runnable()
{

  @Override
  public void run()
  {

    // Get the responseData using getActionsData
    responseData = personalization.getActionsData(ActionTypes.OmniSocialProofData);

    // Once you receive the responseData from getActionsData, use the Handler parse the required d
    new Handler(Looper.getMainLooper()).post(new Runnable()
    {
      @Override
      public void run()
      {

        // Parse the received responseData, get the requiredData from it and use it accordingly
        try
        {
          // Example: textView.setText(requiredData);
        }
        catch (Exception ex)
        {
          // Catch any exception that occured while parsing
          ex.printStackTrace();
        }

      }
    });
  }
}).start();
```