SDK Methods

The following methods are available to use in the SDK. All methods are part of the personalization class.

For parameters that require event names, refer to SDK Events and Action Types for syntax.

report

Reports an event to Monetate. This allows data to later be used for decisions within Monetate. Use this method to report events.

```
public void report(String context, Object events){}
```

Parameters:

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

Example code:

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

IpAddress address = new IpAddress();
address.setIpAddress("127.0.0.0.");

new Thread(new Runnable() {
  @Override
  public void run() {
    personalization.report(EventTypes.ContextIpaddress, address);
  }
}).start();
```

addEvent

Reports an event to Monetate. Use this method to report events that you intend to use to trigger experiences. You can use multiple calls of this method to report multiple events for experiences that require multiple conditions.

This method is used to report events that are used by the `getActionsData` method to determine an experience. If you use this method, you must use `getActionsData` to trigger the experience that satisfies the reported events.

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

Parameters:

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

- `events` is the event data. (Required)

Example code:

```
IpAddress ipAddress = new IpAddress();
ipAddress.setIpAddress("1.0.0.0");

ScreenSize screenSize = new ScreenSize();
screenSize.setHeight(1011);
screenSize.setWidth(2011);

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

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

getActionsData

Request an experience decision from Monetate based off the action type. You can specify one action or multiple action types in an array to get multiple responses.

The experience decision depends on event data reported using `addEvent` calls. Use `addEvent` to add all of the relevant events before you use this method to request a decision.

```
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)

Example code:

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

// Events/context
IpAddress ipAddress = new IpAddress();
ipAddress.setIpAddress("1.0.0.0");

ScreenSize screenSize = new ScreenSize();
screenSize.setHeight(1011);
screenSize.setWidth(2011);

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
String responseData;
new Thread(new Runnable() {
  @Override  public void run() {
    // Gets the responseData using getActionsData functionality
    responseData = personalization.getActionsData(ActionTypes.OmniChannelJson);

    // Once you receive the responseData from getActionsData, use the Handler to parse the require
     new Handler(Looper.getMainLooper()).post(new Runnable() {
      @Override
      public void run()
              // Parse the received responseData, get the requiredData and use it accordingly
        try{
          // Example: textView.setText(requiredData);
        }
        catch(Exception ex){
          // Catch any exception that occurs while parsing
           ex.printStackTrace();
             }
           }
      });
  }
}).start();
```

getActions

Reports an event and immediately requests a decision from Monetate. Use this method if an experience you want to trigger requires a single event. This method returns a JSON object that includes the response data.

The response data can then be used in your application. For example, you can use this method to obtain data to display as a banner on a page.

```
public String getActions(String context, Object event, String[] actionTypes) {returns response;}
```

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

Example code:

```
Account account = new Account("localhost.org", "a-701b337c", "p","localhost");
User user = new User();
user.setCustomerId(null);
user.setDeviceId("DeviceID");

Personalization personalization = new Personalization(user,account);
IpAddress address = new IpAddress();
address.setIpAddress("127.0.0.0.");

String responseData;
new Thread(new Runnable() {
  @Override
  public void run() {

    // Gets the responseData using getActions
    responseData = personalization.getActions(EventTypes.ContextIpAddress, address,ActionTypes.

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

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

  }
}).start();
```

flush

Immediately sends all event data that are currently queued. Use this method if you want to report an event immediately. This method returns a response of success or failure.

This method might throw the following exceptions that you must handle:

- `InterruptedException`
- `ExecutionException`
- `TimeoutException`

```
public String flush() throws InterruptedException, ExecutionException, TimeoutException {}
```

Example code:

```
Personalization personalization = new Personalization(user,account);
IpAddress address = new IpAddress();
address.setIpAddress("127.0.0.0.");
new Thread(new Runnable() {
  @Override
  public void run() {
    Personalization.report(EventTypes.ContextIpaddress, address);
    personalization.flush();
  }
}).start();
```

setCustomerId

Updates the `customerId` within the User object.

```
public void setCustomerId(String customerId);
```

Parameters:

- `customerId` is a string containing the customer ID. (Required)

Example code:

```
personalization.setCustomerId("test_customer_id");
```