# Create an Omnichannel Recommendations Action

Omnichannel recommendations is an integrated recommendations experience that is consistent across all platforms that you connect to Monetate. Whether a customer is viewing your storefront through a mobile app or your site, Omnichannel recommendations ensure a consistent experience.

You can set up a handler for an Omnichannel recommendations action by 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 recommendations data that you can then handle in code.

## Prerequisites

You must first create an Omnichannel experience within Monetate for the methods to reference.

Make note of the WHO settings, as these correspond to events your code listens for. The example experience in this article uses the following WHO settings:

- IP address is 1.0.0.2
- Screen height is at least 500 pixels and screen width is at least 300 pixels

The example code in this article fulfills these conditions and will trigger the Omnichannel experience.

## addEvent

This method reports 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 might 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);

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

## 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.

```
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.OmniChannelRecommendation);
```

## Full Code Example

Complete code example blocks are listed below.

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

// Events/context data
IpAddress ipAddress = new IpAddress();
ipAddress.setIpAddress("10.0.0.2");

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

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

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

```
{

 @Override
 public void run()
 {

  // Gets the responseData using getActionsData
  responseData = personalization.getActionsData(ActionTypes.OmniChannelRecommendation);

  // 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();
```

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 example above uses Thread and Handler to update the main thread and change the UI.