Create a Banner Message

Displaying a banner involves creating an action trigger that activates an Omnichannel experience to return an appropriate JSON object. This object contains data that you can use to populate a banner message. You can use the Get Actions method to set up this trigger and obtain the JSON.

You can use the returned JSON data for multiple purposes. This example uses the JSON to display a banner message.

Creating the Experience

You must first create an Omnichannel experience within Monetate that uses an Omni JSON action type. Refer to Create an Omnichannel Experience for instructions.

You can use the following JSON template for the required input:

```
{
 "meta": {
  "type": "action",
  "tool": "android",
  "version": "1"
 },
 "data": {
  "actionId": "12345",
  "actionType": "monetate:action:OmnichannelJson",
  "component": "home_component_1",
  "impressionId": "5678",
  "json": {
   "text": "Experience Configured for Android Personalization SDK",
   "color": "#FF0000",
   "style": "bold",
   "fontSize":24,
   "buttonText": "Reset All",
   "buttonColor": "#7B68EE",
   "type": "monetate:action:GenericAction"
  }
 }
}
```

This is the JSON object that is returned on the trigger. Modify this template to suit your needs.

Triggering the Experience

The Omnichannel experience is triggered using the `getActions` method. This method is used when you want to satisfy an experience using a single event condition.

```
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)

Full Code Example

Complete code example blocks are listed below. This code triggers the configured Omni JSON experience and handles the experience by displaying a banner. Customize the example code as you see fit.

```
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()
  {

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

    // Once you receive responseData from getActionsData, use the Handler and parse the required
    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();
```

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.