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.

```
report(context: EventTypes, events: ContextData): void
```

Parameters:

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

Example code:

```
const personalizationInstance = new Personalization(user, account);

const cart = {
  "cartLines": [{
    "sku": "SKU1",
    "pid": "PID1",
    "quantity": 2,
    "currency": "USD",
    "value": "24.00"
  }]
};
personalizationInstance.report(
  EventTypes.ContextCart,
  { cartLines: cart }
);
```

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.

```
addEvent(context: EventTypes, events: ContextData): void
```

Parameters:

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

Example code:

```
personalizationInstance.addEvent(EventTypes.ContextIpAddress, {
  ipAddress: "10.0.0.2",
});

personalizationInstance.addEvent(EventTypes.ContextPageView, {
  url: "http://www.monetate.com/index.html",
  pageType: "Home"
});
```

getActionsData

Request an experience decision from Monetate based off the action type. You can specify 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 report all of the relevant events before you use this method to request a decision.

```
getActionsData(actionType: ActionType) : Promise<any>
```

Parameters:

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

Example code specifying a single action:

```
personalizationInstance
  .getActionsData(ActionTypes.OmniChannelRecommendation)
  .then(res => {
    recsData = res[0].actions;
  })
  .catch(error => {
    console.warn('Error!', error);
  });
```

Example code specifying multiple actions:

```
// For multiple actions, send an array
// When an empty array in passed as an argument, it will fetch all the configured Omnichannel acti
const requiredActions : [ ActionTypes.OmniChannelRecommendation
 , ActionTypes.OmniSocialProofData ]
personalizationInstance
 .getActionsData(requiredActions)
 .then(res => {
   recsData = res[0].actions;
 })
 .catch(error => {
   console.warn('Error!', error);
 });
```

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.

```
getActions(actionType, context?, eventData?) : Promise<any>
```

Parameters:

- actionType is the type of action you want to request. You can specify multiple actions in an array to handle. (Required)
- context is name of the event. (Optional)
- eventData is the data associated with the event. (Optional)

Example code:

```jsx
import React, { useEffect, useState} from "react" ;
import Personalization from '@personalization-js-sdk/web';
import { ActionTypes, EventTypes } from "@personalization-js-sdk/common";

const test = () => {
  const personalizationInstance = new Personalization(account, user)
  const [actionData, setActionData] = useState({});

  const handleGetActions = () => {
    const eventType = EventTypes.ContextCart;
    personalizationInstance.getActions(ActionTypes.OmniChannelJson, eventType)
      .then((res) => {
        if (res.length > 0) {
          res[0].actions.map((action) => {
            if (action.json.meta && action.json.meta.tool === 'web') {
              const data = {
                color: action.json.data.json.color,
                text: action.json.data.json.text,
                fontSize: action.json.data.json.fontSize,
                style: action.json.data.json.style,
                buttonText: action.json.data.json.buttonText,
                buttonColor: action.json.data.json.buttonColor,
              };
              setActionData(prevState => {
                return {
                  ...prevState,
                  decisionEventData: data,
                }
              });
            } else {
              console.warn('Meta Object not present in Omni JSON.');
            }
          });
        }
      })
      .catch((error) => console.warn('Please try after some time!', error));
  };

  useEffect(() => {
    handleGetActions();
  }, []);

  return <div>
      <p> {actionData.buttonText} //Render Data On UI </p>
    </div>;
};
```

flush

Immediately sends all event reports 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

```
flush()
```

Example code:

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

PersonalizationInstance.report (
  'monetate:context:IpAddress',
  { ipAddress: 10.10.10.10 }
),
Personalization.setCustomerId("test_customer_id");
Personalization.flush();
```

You can also capture the method response in a variable:

```
const response = Personalization.flush()
```

setCustomerId

Updates the customerId within the User object.

```
setCustomerId(customerId)
```

Parameters:

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

Example code:

```
setCustomerId("test_customer_id");
```