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.

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

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.

```
personalizationInstance.addEvent(EventTypes.ContextScreenSize, { width: 23, height: 34 });

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

Depending on the Social Proof type you want to use, you must use add certain events in addition to these. Refer to the code examples below for these events.

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.

```
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)

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

Viewed Product Code Example

This Social Proof type displays the most viewed products.

```
// Add Context / Events
//--------------------------------------------------------------------------
personalizationInstance.addEvent(EventTypes.ContextScreenSize, { width: 23, height: 34 });

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

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

// Get Social Proof Actions
// ------------------------------------------------
let socialProofData;
personalizationInstance
  .getActionsData(ActionTypes.OmniSocialProofData)
  .then(res => {
    socialProofData = res[0].actions;
  })
  .catch(error => {
    console.warn('Error!', error);
  });
```

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

```
// Add Context / Events
//--------------------------------------------------------------------------
personalizationInstance.addEvent(EventTypes.ContextScreenSize, { width: 23, height: 34 });

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

personalizationInstance.addEvent(EventTypes.ContextProductDetailView, {
  products: [
    {
      productId: "94",
      sku: "sku",
    },
  ],
});

// Get Social Proof Actions
// ------------------------------------------------
let socialProofData;
personalizationInstance
  .getActionsData(ActionTypes.OmniSocialProofData)
  .then(res => {
    socialProofData = res[0].actions;
  })
  .catch(error => {
    console.warn('Error!', error);
  });
```

Multiple Products

```
// Add Context / Events
//------------------------------------------------------------------------
personalizationInstance.addEvent(EventTypes.ContextScreenSize, { width: 23, height: 34 });

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

personalizationInstance.addEvent(EventTypes.ContextProductThumbnailView, {
  "products": ["104","94","97" ]
});

// Get Social Proof Actions
// -----------------------------------------------------
let socialProofData;
personalizationInstance
  .getActionsData(ActionTypes.OmniSocialProofData)
  .then(res => {
    socialProofData = res[0].actions;
  })
  .catch(error => {
    console.warn('Error!', error);
  });
```

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

```
// Add Context / Events
//-------------------------------------------------------------------------
personalizationInstance.addEvent(EventTypes.ContextScreenSize, { width: 23, height: 34 });

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

personalizationInstance.addEvent(EventTypes.ContextProductDetailView, {
  products: [{productId: "94",sku: "sku"}],
});

// Get Social Proof Actions
// ------------------------------------------------------
let socialProofData;
personalizationInstance
  .getActionsData(ActionTypes.OmniSocialProofData)
  .then(res => {
    socialProofData = res[0].actions;
  })
  .catch(error => {
    console.warn('Error!', error);
  });
```

Multiple Products

```
// Add Context / Events
//--------------------------------------------------------------------------
personalizationInstance.addEvent(EventTypes.ContextScreenSize, { width: 23, height: 34 });

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

personalizationInstance.addEvent(EventTypes.ContextProductThumbnailView, {
  "products": ["88","86","97" ]
});

// Get Social Proof Actions
// ------------------------------------------------------
let socialProofData;
personalizationInstance
  .getActionsData(ActionTypes.OmniSocialProofData)
  .then(res => {
    socialProofData = res[0].actions;
  })
  .catch(error => {
    console.warn('Error!', error);
  });
```

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.

```
// Add Context / Events
//------------------------------------------------------------------------
personalizationInstance.addEvent(EventTypes.ContextScreenSize, { width: 23, height: 34 });

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

personalizationInstance.addEvent(EventTypes.ContextProductDetailView, {
  products: [{productId: "94",sku: "sku"}],
});

// Get Social Proof Actions
// -------------------------------------------------------
let socialProofData;
personalizationInstance
  .getActionsData(ActionTypes.OmniSocialProofData)
  .then(res => {
    socialProofData = res[0].actions;
  })
  .catch(error => {
    console.warn('Error!', error);
  });
```

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

```
// Add Context / Events
//-------------------------------------------------------------------------
personalizationInstance.addEvent(EventTypes.ContextScreenSize, { width: 23, height: 34 });

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

personalizationInstance.addEvent(EventTypes.ContextProductDetailView, {
  products: [{productId: "94",sku: "sku"}],
});

// Get Social Proof Actions
// ------------------------------------------------------
let socialProofData;
personalizationInstance
  .getActionsData(ActionTypes.OmniSocialProofData)
  .then(res => {
    socialProofData = res[0].actions;
  })
  .catch(error => {
    console.warn('Error!', error);
  });
```

Multiple Products

```
// Add Context / Events
//------------------------------------------------------------------------
personalizationInstance.addEvent(EventTypes.ContextScreenSize, { width: 23, height: 34 });

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

personalizationInstance.addEvent(EventTypes.ContextProductThumbnailView, {
  "products": ["90","91","92" ]
});

// Get Social Proof Actions
// ------------------------------------------------------
let socialProofData;
personalizationInstance
  .getActionsData(ActionTypes.OmniSocialProofData)
  .then(res => {
    socialProofData = res[0].actions;
  })
  .catch(error => {
    console.warn('Error!', error);
  });
```