Configure an Omnichannel Recommendations Action

Follow these steps to build an Omnichannel experience with a Product Recommendations action. The Engine API responds with an array of product details.

> Product IDs must match across the Monetate JavaScript API, the product catalog, and the request.

The requesting application is responsible for creating the container or carousel and all desired styling for the product recommendations.

1. Create an Omnichannel experience, and then configure the WHY and WHO settings.
2. Click **WHAT** and then click **ADD ACTION**.
3. Click **Product Recommendations**.
4. Select the recommendations template that you want to use.
5. Select from **Product Recommendation** the recommendation strategy you want the action to use.
6. Select from **Primary Fallback** the recommendation strategy that you want used first to supplement the strategy you selected in the previous step if it can't display enough recommended products to meet the minimum.
7. Select from **Secondary Fallback** the recommendation strategy that you want used if the strategies that you selected in steps 5 and 6 can't display enough recommended products to meet the minimum.
8. Enter the minimum number of products the primary recommendation strategy must display before the action triggers the primary fallback recommendation strategy.

   > Setting the minimum to zero prevents the fallback recommendation strategy from rendering and thus causes the action to fire in all scenarios, even if no products are recommended. This situation can also result in customers being counted in the experience without seeing recommendations.

   > If in step 5 you selected a recommendation strategy with **Prepend context item in recommendation** enabled, then the context item counts toward meeting the threshold you set in **Minimum products returned**.

9. Enter the maximum number of products to include in the recommendations results.

    > If in step 5 you selected a recommendation strategy with **Prepend context item in recommendation** enabled, then the context item counts toward meeting the threshold you set in **Maximum products returned**.

10. Enter into **Pinned Products** the `id` attribute value for each product that you want to always appear at the beginning of the recommendations.

    > If you enter product IDs into **Pinned Products** and if in step 5 you selected a recommendation strategy with **Prepend context item in recommendation** enabled, then the products identified at the action level appear *before* the context product in the recommendations results.

11. Select a product catalog attribute on which duplicate recommended products are removed from the results.

    > Ensure that the attribute that you select appears in the product catalogs used in the recommendation strategies that you selected in steps 5, 6, and 7.

12. Enter the identifier for the recommendations results.

13. Click **ADD CONDITION**, select an attribute category, and then select and configure an option to set any conditions that must be met for the action to fire. See Action Conditions for more information.

14. Click **CREATE**.

Using Product Catalog Custom Fields in the Action

You can use custom fields (for example, star ratings) from a product catalog in an Omni Intelligent Recommendations action.

> Contact your dedicated Services team members to request the custom fields be added to the Omni Intelligent Recommendations action template.

Any custom field that you use in the action must also appear in the product catalog configured as part of the recommendation strategy that you selected for the action.

Configuring the Engine API Request

Ensure that you include any relevant context in the request.

This example is for recommendation strategies that use the Most Viewed (Product Detail Page), Top Selling by Purchase Count, or Top Selling by Gross Revenue recommendation algorithms.

```
{
 "channel": "a-927c8120/p/example.com",
 "events":
 [
   {
     "eventType": "monetate:decision:DecisionRequest",
     "requestId": "11111",
     "includeReporting": true
   },
   {
     "eventType": "monetate:context:IpAddress",
     "ipAddress": "188.221.179.186"
   },
   {
     "eventType": "monetate:context:PageView",
     "url": "https://www.example.com",
     "pageType": "Index"
   }
 ]
}
```

This example is for recommendation strategies that use the Viewed and Also Viewed recommendation algorithm with recommendations based on the first item viewed on the current page or the last item viewed in any session.

```
{
 "channel": "a-927c8120/p/example.com",
 "events":
 [
   {
     "eventType": "monetate:decision:DecisionRequest",
     "requestId": "11111",
     "includeReporting": true
   },
   {
     "eventType": "monetate:context:IpAddress",
     "ipAddress": "188.221.179.186"
   },
   {
     "eventType": "monetate:context:ProductDetailView",
     "products":
     [
       {
         "productId": "product72",
         "sku": "product72-large-green"
       }
     ]
   },
   {
     "eventType": "monetate:context:PageView",
     "url": "https://www.example.com",
     "pageType": "ProductPage"
   }
 ]
}
```

This example is for recommendation strategies that use the Viewed and Also Viewed recommendation algorithm based on item(s) viewed on the current page.

```
{
 "channel": "a-927c8120/p/example.com",
 "events":
 [
   {
     "eventType": "monetate:decision:DecisionRequest",
     "requestId": "11111",
     "includeReporting": true
   },
   {
     "eventType": "monetate:context:IpAddress",
     "ipAddress": "188.221.179.186"
   },
   {
     "eventType": "monetate:context:ProductThumbnailView",
     "products":
     [
       "123",
       "456",
       "789"
     ]
   },
   {
     "eventType": "monetate:context:PageView",
     "url": "https://www.example.com",
     "pageType": "Index"
   }
 ]
}
```

This example is for recommendation strategies that use the Viewed and Also Viewed and the Viewed and Later Purchased recommendation algorithms based on the last item purchased in any session.

> You must send the product ID of the last purchase.

```
{
 "channel": "a-927c8120/p/example.com",
 "events":
 [
   {
     "eventType": "monetate:decision:DecisionRequest",
     "requestId": "11111",
     "includeReporting": true
   },
   {
     "eventType": "monetate:context:IpAddress",
     "ipAddress": "188.221.179.186"
   },
   {
     "eventType": "monetate:context:Purchase",
     "productId": "product43",
     "purchaseId": "123456789",
     "purchaseLines":
     [
       {
         "sku": "product72colour2",
         "pid": "product72",
         "quantity": 2,
         "currency": "GBP",
         "value": "24.00"
       }
     ]
   },
   {
     "eventType": "monetate:context:PageView",
     "url": "https://www.example.com",
     "pageType": "Index"
   }
 ]
}
```

This example is for recommendation strategies that use the Viewed and Also Viewed and the Later Purchased recommendation algorithms based on the last item carted in any session.

> You must send the product ID of the last item added to the cart.

```
{
 "channel": "a-927c8120/p/example.com",
 "events":
 [
   {
    "eventType": "monetate:decision:DecisionRequest",
    "requestId": "11111",
    "includeReporting": true
   },
   {
    "eventType": "monetate:context:IpAddress",
    "ipAddress": "188.221.179.186"
   },
   {
    "eventType": "monetate:context:Cart",
    "cartLines":
    [
      {
       "sku": "product72colour2",
       "pid": "product72",
       "quantity": 2,
       "currency": "GBP",
       "value": "24.00"
      }
    ]
   },
   {
    "eventType": "monetate:context:PageView",
    "url": "https://www.example.com",
    "pageType": "Index"
   }
 ]
}
```

Request Best Practices by Integration Type

If your site uses a hybrid integration, keep in mind these best practices when configuring an Omni Intelligent Recommendations action:

- You only have to send product IDs if you base recommendations on a specific product (for example, a recommendation strategy configured with the Viewed and Also Viewed recommendation algorithm based on the last item viewed in any session or the first item viewed on the current page).
- `ProductDetailView`, `CartLines`, and `PurchaseLines` events are obtained from the Monetate tag. You're not required to send these.

If your site uses a pure Engine API implementation, keep in mind that you must send all `ProductDetailView`, `CartLines`, and `PurchaseLines` events.

When selecting recommendation strategies for this type of action, be aware of the following conditions related to the recommendation algorithm configured as part of the recommendation strategy:

- **Most Viewed (Product Detail Page), Top Selling by Purchase Count, and Top Selling by Gross Revenue**: You don't need to send any product-specific information in the requests.
- **Viewed and Also Viewed based on the first item viewed on the current page or on the last item viewed in any session**: Send the `ProductDetailView` event in the request.
- **Viewed and Also Viewed based on item(s) viewed on the current page**: Send all product IDs in the `ProductThumbnailView` event.

Product Recommendations Reporting

When using the Engine API for recommendations, you must make subsequent requests when products are visually presented to the customer and when a customer clicks or taps any of the recommended products. This request is necessary to record the applicable item impressions and item clicks.

- Use `monetate:record:RecImpressions` whenever products are presented to the customer or whenever a customer swipes along a carousel to see additional recommendations.
- Use `monetate:record:RecClicks` whenever a customer clicks or taps a product recommendation so that Monetate can record the click.

> See the Engine API documentation in the Monetate platform for code examples for using `monetate:record:RecImpressions` and `monetate:record:RecClicks`.

> The `monetate:record:EndcapImpressions` and `monetate:record:EndcapClicks` requests are deprecated. Use `monetate:record:RecImpressions` and `monetate:record:RecClicks` instead.

Complete the following steps to report product recommendations.

1. In the initial request, send the relevant product IDs depending on the recommendation strategy that you're using.
2. Find the unique `recToken` in the initial response. The response contains a unique `recToken` for each product within the recommendation strategy for that experience, such as the one in this example:

   ```
   "recToken": "rt.1.WzL4LjI3NTQsICIwNDkwNTMiLCAiLDQ5LDUzIiwgMTQ4ODYsIDAsICIyMDIxLTE
   ```

3. Send the `recToken` in a second request when the impression occurs, followed by a third request when the click occurs.
   - **Impressions**

```
{
 "eventType": "monetate:record:RecImpressions",
 "recImpressions":
 [
  "rt.1.WzL4LjI3NTQsICIwNDkwNTMiLCAiLDQ5LDUzIiwgMTQ4ODYsIDAsICIyMDIxLTExLTA
  "rt.1.yyy"
 ]
}
```

- **Clicks**

```
{
 "eventType": "monetate:record:RecClicks",
 "recClicks":
 [
  "rt.1.WzL4LjI3NTQsICIwNDkwNTMiLCAiLDQ5LDUzIiwgMTQ4ODYsIDAsICIyMDIxLTExLTA
  "rt.1.yyy"
 ]
}
```