By using the Google Analytics Enhanced Ecommerce features, you can measure and analyze shopping activity on your website (product impressions, purchases, etc.). Kentico provides functionality that helps get the data of products and purchases in a format suitable for logging via Google Analytics.

Visit the Google Analytics website to set up a Google Analytics account for the website that you wish to monitor.

MVC websites

The information on this page applies to e-commerce sites developed using the portal engine.

If you are developing your on-line store using ASP.NET MVC, see: Google Analytics Enhanced Ecommerce on MVC sites

Adding Google Tag Manager

Before you can start measuring e-commerce activities, you need to add Google Tag Manager to the pages of your website:

1. Open the **Pages** application in Kentico.
2. Select the root page of your site's content tree.
3. Open the **Master page** tab (in **Edit** mode).
4. Copy the snippets from the Google Tag Manager Quick Start Guide into the master page content.
   - Use the appropriate areas to add the snippets into the *<head>* tag and after the opening *<body>* tag.
5. Click **Save**.

Note: You need to repeat the process for any additional master pages that do not inherit content from the site's root page (if you wish to use Google Analytics tracking in the website subsections under the given master pages).

You can now start collecting your site's e-commerce data and tracking events using Google Analytics. See the following sections for more information about setting up individual types of tracking.

Collecting e-commerce data

We recommend using a data layer to collect and send data to Google Analytics. For detailed information, see the Enhanced Ecommerce (UA) Developer Guide.

Kentico provides the following macro methods that allow you to get the appropriate JSON data, which you can push into the data layer:

- **GetGtmProductJson** – returns data for a specified product. Provides the following fields by default: *id* (the product's *SKU* value), *name*, *price* and *brand*

  The *GetGtmProductJson* method is available for both macros (for *Text / XML* transformations and general usage in the Kentico administration interface) and ASCX transformations. The method's ASCX transformation variant always returns data for the currently transformed product.

  Example of returned JSON data

  ```
  {
    "id" : "AER-AP",
    "name" : "AeroPress",
    "price" : 18.12,
    "brand" : "Aerobie"
  }
  ```

- **GetGtmShoppingCartItemsJson** – returns data for all products in the current shopping cart. By default, provides the same fields as the *GetGtmProductJson* method and also adds the unit *quantity* for each product.

Example of returned JSON data

```
[{
  "name": "Anfim Super Caimano",
  "id": "ANF-GR-CAIMANO",
  "price": 1245.92,
  "brand": "Anfim",
  "quantity": 1
},
{
  "name": "Colombia Carlos Imbachi (16 oz)",
  "id": "CO-COL-IMBACHI-16-OZ",
  "price": 14,
  "brand": "SA Coffee",
  "quantity": 4
}]
```

- **GetGtmPurchaseJson** – returns data representing completed orders (purchases). Provides the following JSON data by default:
  - *actionField* object representing the overall purchase, with the following fields:
    - *id* (the integer ID of the Kentico order object)
    - *revenue* (the order's *GrandTotal* value, i.e. the final price including discounts, shipping, tax, and reductions from applied gift cards)
    - *shipping*
    - *tax*
  - *products* object containing all products within the order (the product data format is the same as provided by the *GetGtmShoppingCartItemsJson* method)

Important: The *GetGtmPurchaseJson* method only works on the page that you display to customers after they go through your website's checkout process (the page specified by the **Final step URL** property of your site's **Page wizard manager** web part, typically a payment page or "thank you" page).

Example of returned JSON data

```
{
  "actionField": {
    "id": 123,
    "revenue": 1376.68,
    "shipping": 0,
    "tax": 74.76
  },
  "products": [{
    "name": "Anfim Super Caimano",
    "id": "ANF-GR-CAIMANO",
    "price": 1245.92,
    "brand": "Anfim",
    "quantity": 1
    },
    {
    "name": "Colombia Carlos Imbachi (16 oz)",
    "id": "CO-COL-IMBACHI-16-OZ",
    "price": 14,
    "brand": "SA Coffee",
    "quantity": 4
  }]
}
```

Customizing the returned JSON data

By default, the listed methods do not set all of the optional fields that can be processed by Google Analytics. You can extend the returned data by calling the methods with an additional parameter containing a JSON string. The string is then processed and merged into the resulting JSON data.

For example, the following macro returns JSON data for a specified product, including the default fields (*id*, *name*, *price*, *brand*) and an additional *list* field:

```
{% GetGtmProductJson(SKUID, "{'list' : 'Search Results'}" %}
```

If you require more advanced modifications, developers can fully customize how the system creates the JSON data for products and orders.

See: Customizing e-commerce data for Google Analytics

Use the methods to create a Google Analytics implementation that suits the purposes of your website. The following sections contain best practices for various types of e-commerce tracking on Kentico portal engine sites:

- Product impressions
- Product clicks
- Views of product details
- Checkout steps
- Purchases

Product impressions

To log impressions of products displayed to visitors:

1. Edit the transformations that you use to display products on your website.
2. Identify a suitable HTML element wrapping the content of individual products within the transformation code and use the **GetGtmProductJson** method to add the product JSON data into a data attribute.

   Text / XML transformation example

   ```
   <article class="product-tile" data-gtm-product='{% GetGtmProductJson(SKUID) %}'>
   ```

   ASCX transformation example

   ```
   <article class="product-tile" data-gtm-product='<%# GetGtmProductJson() %>'>
   ```

   Note: Use single quotes to enclose the values of the data attribute to ensure that the returned JSON value containing double quotes is processed correctly.

3. Add the **Javascript** web part onto the templates of pages that display products.
4. Run a script that collects the values of the product data attributes and pushes them into the data layer. For example, configure the web part in the following way:

   - **In-line script**:

   ```
   document.addEventListener("DOMContentLoaded", function(event) {
     let productElements = document.getElementsByClassName("product-tile");
     let gtmJsonProductImpressions = [];
     for (let i=0;i<productElements.length;i++) {
       gtmJsonProductImpressions.push(productElements[i].getAttribute("data-
   gtm-product"));
     }
     if (gtmJsonProductImpressions.length > 0) {
       dataLayer.push({
             "ecommerce": {
               "currencyCode": "{% ECommerceContext.CurrentCurrency.
   CurrencyCode %}",
           "impressions": gtmJsonProductImpressions
         }
       });
     }
   });
   ```

   - **In-line script page location**: Startup script *(renders the script tag near the end of the page's <body> content)*
   - **Generate script tags**: yes (checked)

Product clicks

To log clicks of product links:

1. Define a JavaScript function for the pages containing product links (for example using the **Javascript** web part or by linking an external .js file).

   - Within the function, push the product data into the data layer. The function must have parameters containing the product JSON data and the URL of the clicked link.
   - Use the *eventCallback* datalayer variable to perform redirection after the product data is sent to Google Analytics.

Function example

```
function productClick(productJson, link) {
  dataLayer.push({
    "event": "productClick",
    "ecommerce": {
      "click": {
        "actionField": {"list" : "Search Results"},
        "products": [productJson]
      }
    },
    "eventCallback": function () {
      document.location.href = link;
    }
  });
}
```

2. Add an **onclick** script to your product links (for example within product list transformations) and call the defined product click function.
   - Use the **GetGtmProductJson** method to get the JSON data of the clicked product, either directly in the *onclick* attribute or get the value from a data attribute of a related product element.

Text / XML transformation example

```
<article class="product-tile" data-gtm-product='{% GetGtmProductJson(SKUID)
%}'>
  <a href="{% GetDocumentUrl() %}" onclick='productClick(this.parentNode.
getAttribute("data-gtm-product"), "{% GetDocumentUrl() %}" );return false;'>
    ...
  </a>
</article>
```

ASCX transformation example

```
<article class="product-tile" data-gtm-product='<%# GetGtmProductJson() %>'>
  <a href="<%# GetDocumentUrl() %>" onclick='productClick(this.parentNode.
getAttribute("data-gtm-product"), "<%# GetDocumentUrl() %>" );return
false;'>
    ...
  </a>
</article>
```

Views of product details

To log views of product detail pages by visitors:

1. Edit the transformations that you use to display product details on your website.
2. Add a script tag to the end of the transformation, and write a script that pushes the product data into the data layer. Use the **GetGtmProductJson** method to get the JSON data of the transformed product.

ASCX transformation example

```
...

<script type="text/javascript">
document.addEventListener("DOMContentLoaded", function (event) {
  dataLayer.push({
    "ecommerce": {
      "detail" : {
        "actionField": {"list": "Gallery"},
        "products": [<%# GetGtmProductJson() %>]
      }
    }
  });
});
</script>
```

Checkout steps

To log progress of visitors through the steps of your website's checkout process:

1. Open the **Pages** application.
2. Locate the pages that define the steps of your site's checkout process.
3. Add the **Javascript** web part to the first checkout step where modifications of the shopping cart content are no longer allowed.
4. Run a script that pushes the required checkout data into the data layer, including the shopping cart's product data. Use the **GetGtmShoppingCartItemsJson** method to get the JSON data of the products in the current shopping cart. For example, configure the web part in the following way:
   - **In-line script**:

     ```
     document.addEventListener("DOMContentLoaded", function(event) {
       dataLayer.push({
         "event": "checkout",
         "ecommerce": {
           "checkout": {
             "actionField": {"step": 1},
             "products": {% GetGtmShoppingCartItemsJson() %}
           }
         }
       });
     });
     ```

   - **In-line script page location**: Startup script *(renders the script tag near the end of the page's <body> content)*
   - **Generate script tags**: yes (checked)
5. Add the **Javascript** web part to the remaining checkout steps.
6. For each step, run a script that pushes the required checkout data into the data layer, but without including the *products* field. For example:

```
document.addEventListener("DOMContentLoaded", function(event) {
  dataLayer.push({
    "event": "checkout",
    "ecommerce": {
      "checkout": {
        "actionField": {
          "step": 2,
          "option" : "{% ECommerceContext.CurrentShoppingCart.ShippingOption.
ShippingOptionDisplayName %}"
        }
      }
    }
  });
});
```

Purchases

To log purchases (i.e. completed orders) on your website:

1. Open the **Pages** application.
2. Select the page that you display to customers after they go through your website's checkout process (i.e. the page specified by the **Final step URL** property of your site's **Page wizard manager** web part, typically a payment page or "thank you" page).
3. Ensure that the page has the context of the completed order after a customer makes a purchase. The system provides this context using the **o** parameter in the URL query string (the parameter's value is a hash, not the actual ID of the related order).
    - If you log the purchase directly on the page specified as the *Final step URL* of your site's checkout process, you do not need to make any adjustments.
    - If you want to log the purchase at a later point, you need to propagate the *o* query string parameter to the given page. For example, if your *Final step URL* leads to a payment page containing the **Payment form** web part, and you want to log the purchase on the page specified in the web part's **Thank you page URL** property, you can include the *o* parameter by adding the following macro to the URL value:

      Example - Thank you page URL

      ```
      ~/Order-Completed{% if (!String.IsNullOrEmpty(Querystring.o)) {"?o=" +
      Querystring.o} %}
      ```

4. Add the **Javascript** web part onto the page where you want to log the purchase.
5. Expand the **Behavior** category of the web part's properties and click **Edit value** (▶) next to the **Enabled** property.
6. Add the following macro condition into the value:

```
{% !String.IsNullOrEmpty(QueryString.o) %}
```

The condition prevents logging of empty purchases by ensuring that the JavaScript runs only if the page is viewed after completing an order (i.e. the URL query string contains the *o* parameter).

7. Run a script that pushes the completed order's JSON data into the data layer. Use the **GetGtmPurchaseJson** macro method to get all data required for the *purchase* field. For example, configure the web part in the following way:

    - **In-line script**:

```
document.addEventListener("DOMContentLoaded", function(event) {
  dataLayer.push({
    "ecommerce": {
      "purchase": {% GetGtmPurchaseJson() %}
    }
  });
});
```

- **In-line script page location**: Startup script *(renders the script tag near the end of the page's <body> content)*
- **Generate script tags**: yes (checked)