



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

API Reference

Monitoring DSL

Monitoring DSL

Contents

- **1 Monitoring DSL**
 - **1.1 Description**
 - **1.2 <properties> (mandatory)**
 - **1.3 <events>**
 - **1.4 <event>**
 - **1.5 <trigger> (mandatory child element)**
 - **1.6 <val>**

Description

The Web Engagement Monitoring Agent DSL Reference lists all the DSL elements that you can use to define Business Events in the Monitoring Agent Component. For details about implementation, read [Creating Business Events](#). The DSL specifies the document elements to monitor, the events to send to the Frontend Server, and the data to include with those events.

```
<?xmlversion="1.0"encoding="utf-8"?>
<properties debug="false">
  <events>
    <event id="AddToCartEvent" name="AddToCart">
      <trigger name="AddToCartTrigger" element="img.bdt-addToCart" action="click"
url="http://www.MySite.com/" count="1">
        <val name="productName"
value="$(event.target).parents('div.hproduct').find('h3.name a').text()"/>
        <val name="productModel"
value="$(event.target).parents('div.hproduct').find('span.model')"/>
        <val name="productSKU"
value="$(event.target).parents('div.hproduct').find('span.sku').text()"/>
        <val name="productPrice"
value="$(event.target).parents('div.hproduct').find('h4.price').text()"/>
      </trigger>
    </event>

    <event id="SearchEvent" name="Search">
      <trigger name="GoSearchClick" element="td#gobutton input" action="click"
url="http://www.MySite.com" count="1"/>
        <val name="searchString" value="$( 'input.searchfield:text' ).val();"/>
      </trigger>
    </event>
  </events>
</properties>
```

<properties> (mandatory)

The <properties> element is the main root element of the DSL file. It has an optional debug attribute and a mandatory <events> child.

debug (optional)

The debug attribute enables debugging in the browser by setting its value to the JavaScript Boolean true. The debugging information opens a pop-up window and shows the JSON serialized event data for the business events before they are sent to the Frontend Server.

Note: In some browsers, using the debug attribute can affect the performance of the Frontend Server by delaying the event dispatch.

<events>

The <events> element contains a list of all the business events that can be generated during monitoring. These business events are captured in the <event> child element.

<event>

The <event> element contains mandatory id and name attributes, and an optional condition attribute. An <event> must also have one or more <trigger> children, which define the conditions that must be matched to generate an event.

Note: If the <trigger> child is omitted, the event will never be generated.

id (mandatory)

The id is the internal identifier for the event. This must be unique across the DSL file, to make a distinction between the events sent by the browser.

name (mandatory)

The name is sent to the Frontend Server. A DSL file may contain several <event> elements with identical values for name, but with different values for id. For example, if your website includes a search form, you can submit this form by clicking on the 'search' button or by pressing the 'enter' key. Inside the browser, the click and key press events are clearly distinct, but are not relevant for the Frontend Server.

The following example shows how to create two business events which return the same event name to the Frontend Server:

```
<?xmlversion="1.0"encoding="utf-8"?>
<properties debug="false">
  <events>
    <event id="SearchEvent" name="Search">
      ...
    </event>
    <event id="SearchKeyDownEvent" name="Search">
      ...
    </event>
  </events>
</properties>
```

condition (optional)

The condition attribute is a JavaScript Boolean expression. If it is present, the event's triggers will be installed in the page if the condition evaluates to true.

The following example creates a business event with a timer which can be triggered only if the text inside the <h1> tag on the page is "Compare":

```
<event id="InactivityTimeout4CompareProductsEvent" name="InactivityTimeout4CompareProducts"
condition="$('h1').text() == 'Compare'">
```

```
<trigger name="InactivityTimeout4CompareProductsTrigger" element="" action="timer:10000"
type="timeout"
  url="http://www.MySite.com/site/olspage.jsp" count="1"/>
...
</event>
```

Since the event (in this case 'InactivityTimeout4CompareProductsEvent') will never be generated if its triggers are not installed, the `condition` attribute allows you to place conditions on any feature of the environment that can be tested by a JavaScript Boolean expression, in order to monitor and generate events.

<trigger> (mandatory child element)

The `<trigger>` element defines the conditions that must be matched to generate business events, as well as the data to be included with the event. If several triggers are part of the event definition, they must all match to raise the business event. If each trigger matches a different DOM event in the browser, then the set of triggers specifies a series of web events that must occur before the parent business event is submitted to the Frontend Server.

The `<trigger>` element has mandatory `name`, `element`, and `action` attributes, and optional `url` and `count` attributes. It can have and 0 or more `<val>` children.

name (mandatory)

This attribute specifies the name of the trigger. It must be unique in the parent `<event>` element. If an `<event>` element has multiple triggers, they must all have different names.

element (mandatory)

The `element` attribute specifies the document's DOM element to which the trigger should be attached. The value of `element` should be a jQuery selector. For details on jQuery selectors, see <http://api.jquery.com/category/selectors/>. If the `action` attribute is set to `timer:nnn`, then `element` can be set to null.

action (mandatory)

The `action` specifies the DOM event to track. The trigger is matched if this DOM event is targeted to the DOM element specified by the `element` attribute.

You can specify **standard DOM events** for the action:

- Browser Events
- Document Loading
- Keyboard Events
- Mouse Events
- Form Events

In addition to the standard DOM events, the DSL supports the following two values: `timer` and `enterpress`.

The `enterpress` event signals that the user has pressed the 'enter' key. It is more specific than the standard DOM 'keypress' event, which is raised when any key is pressed.

The following example adds a value named "searchString" when the "Search" event is generated and sent to the Frontend Server. The value is the string entered in the "searchfield" textbox.

```
<event id="SearchKeyDownEvent" name="Search">
  <trigger name="SearchKeyDown" element="input.searchfield:text" action="enterpress"
    url="http://www.MySite.com" count="1"/>
  <val name="searchString" value="$('input.searchfield:text').val()"/>
</event>
```

Setting the action attribute to `timer` allows you to specify the time interval, in milliseconds. For example, `action="timer:10000"` specifies a 10-second timer. If you set the value of `action="timer:nnn"`, you must also include the type attribute to specify how the timer works.

type (mandatory when action="timer:nnn")

The type attribute is mandatory when `action="timer:nnn"`. The type can have a value of either `timeout` or `nomove`, which specifies how the timer action works.

If `type="timeout"`, the timer interval begins after the page is loaded. If `type="nomove"`, the timer resets each time the user moves the mouse.

In the following example, the "InactivityTimeout" event is generated after the user has been inactive for 10 seconds.

```
<event id="InactivityTimeout4CompareProductsEvent" name="InactivityTimeout"
  condition="$('h1').text() == 'Compare'">
  <trigger name="InactivityTimeout" element="" action="timer:10000" type="nomove"
    url="http://www.MySite.com/site/olspage.jsp" count="1"/>
  <val name="products" value="..." />
</event>
```

If `type="timeout"` was specified instead, the event would be generated 10 seconds after the page was loaded.

url (optional)

The url attribute defines the URL of a specific page which raises the business event. The business event is not submitted if the current document's URL does not match the URL parameter.

count (optional)

The count attribute specifies how many times the trigger needs to be matched before the event is generated and sent to the Frontend Server.

<val>

The `<val>` element can be used to add data to the business event. You can have 0 or more `<val>` elements; each instance adds a field to the business event. If `<val>` is a child of `<trigger>`, it can also have access to the DOM event matched by the trigger.

name (mandatory)

The name attribute is the name of the value in the generated business event. The name of each val must be unique inside a parent event. The name will be added to the generated business event's data, along with the corresponding value attribute.

The following example adds a value named "searchString" when the "Search" event is generated and sent to the Frontend Server.

```
<event id="SearchEvent" name="Search">
  <trigger name="GoSearchClick" element="td#gobutton input" action="click" url=" "
count="1"/>
  <val name="searchString" value="$('input.searchfield:text').val();"/>
</event>
```

The following output is an example of an event submitted to the Frontend Server when the visitor enters "my search string" in the search box and clicks the search button. The "eventName" parameter is the name attribute of the <event> element. The "searchString" parameter is the name attribute of the <val> element and "my search string" is its value. The additional fields are generated automatically by the DSL code:

```
{
  "data":{
    "searchString":"my search string"
  },
  "eventType":"BUSINESS",
  "eventName":"Search",
  "eventID":"D88B2FF5A9C24095837CF105FB6D5CF9",
  "pageID":"A9D1E9265D444351876C13D6C5FA5FAD",
  "timestamp":1309962580226,
  "globalVisitID":"7E67BA9701124F738CAC80DDFEA1D705",
  "visitID":"4608DD210B034AC18C65C2C2275CD8B6",
  "userID":"",
  "url":"http://www.MySite.com/site/",
  "category":""
}
```

value (optional)

The value attribute specifies the value to associate with the name attribute in the field of the generated event. Its value can be any JavaScript code which returns a serializable object.

The following example tracks search events and includes the search string in the event when it is sent to the Frontend Server. In this example, there is only one search input box on the page:

```
<event id="SearchEvent" name="Search">
  <trigger name="GoSearchClick" element="td#gobutton input" action="click"
url="http://www.MySite.com" count="1" />
  <val name="searchString" value="$('input.searchfield:text').val();"/>
</event>
```

In the following example, the "AddToCart" event is tracked, including information about the product that was added: name, model, SKU, and price. Tracking by clicking on the "add to cart" button does not provide information about which button was clicked and which product was added to the cart. To get this information, you need to use the DOM event object: "event.target" identifies the clicked button, which can provide information related to the product.

```
<?xml version="1.0"encoding="UTF-8"?>
```

```
<event id="AddToCartEvent" name="AddToCart">
  <trigger name="AddToCartTrigger" element="div.info-side img.bdt-addToCart" action="click"
url="http://www.MySite.com" count="1">
    <val name="productName" value="$(event.target).parents('div.hproduct').find('h3.name
a').text()"/>
    <val name="productModel"
value="$(event.target).parents('div.hproduct').find('span.model').text()"/>
    <val name="productSKU"
value="$(event.target).parents('div.hproduct').find('span.sku').text()"/>
    <val name="productPrice"
value="$(event.target).parents('div.hproduct').find('h4.price').text().replace('Sale:', )"/>
  </trigger>
</event>
```