



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.

# User's Guide

## Managing Business Events

# Managing Business Events

## Contents

- **1 Managing Business Events**
  - **1.1 Overview**
  - **1.2 Creating Business Events by Customizing the DSL File**
  - **1.3 Creating Business Events by Using the Monitoring Agent API**

## Overview

When you **create an application**, a set of Domain Specific Language (DSL) files that are used by your application is also created. These files are defined in the **apps\Your application name\resources\dsl\** directory. You can use the DSL to define **business events**) that are specific to your solution needs.

### Default domain-model.xml

The **domain-model.xml** is the main default DSL file for your application:

```
<?xml version="1.0" encoding="utf-8" ?>
<properties>
  <events>
    <!-- Add your code here
    <event name="">
    </event>
    -->

    <!-- This is template for your search event -->
    <!--
    <event id="Search" name="Search">
      <trigger name="SearchTrigger" element="" action="click" url="" count="1" />
      <val name="searchString" value="" />
    </event>
    -->

    <event id="Timeout-30" name="Timeout-30">
      <trigger name="TimeoutTrigger" element="" action="timer:30000" type="timeout"
url="" count="1" />
    </event>

  </events>
</properties>
```

By using the **<event>** element, you can create as many business events as you need. These events can be tied to the HTML components of your page and can have the same name, as long as they have different identifiers (these identifiers must be unique across the DSL file, to make a distinction between the events sent by the browser). It can be useful to associate several HTML components with the same event if these HTML components have the same function. For instance, you can define several events associated with a search feature and give all these events the same name: "Search".

For each event, you can define triggers which describe the condition to match in order to submit the event:

- Triggers can implement timeouts.
- Triggers can be associated with DOM events.
- You can define several triggers for the same event (see **<trigger>** for further details).

Each trigger should have an **element** attribute that specifies the document's DOM element to attach the trigger to, and the **action** attribute, which species the DOM event to track.

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 following example generates a "Search" event if the visitor does a site search. The "searchString" value is the string entered in the "INPUT.search-submit" form.

```
<event name="Search">
  <trigger name="SearchTrigger" element="INPUT.search-submit" action="click" url=""
count="1" />
  <val name="searchString" value="INPUT.search-submit" />
</event>
```

If the DSL uses the optional condition attribute, the event's triggers are installed on the page if the condition evaluates to true. The following example creates a Business event with a time that can be triggered only if the text inside the <h1> tag is "Compare":

```
<event 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>
```

If the DSL uses an optional postcondition attribute, this can manage how an event is generated by checking a condition after the actions are completed. The following example creates a Business event timeout by timer if a page is in focus. In this case, the event does not generate if the page is opened in the background:

```
<event name="Timeout-10" condition="" postcondition="document.hasFocus() === true">
  <trigger name="TimeoutTrigger" element="" action="timer:10000" type="timeout" url=""
count="1" />
</event>
```

A DSL trigger can use the type attribute. This can have a value of either timeout or nomove, which specifies how the timer action works. If the type is timeout, then the timer interval begins after the page is loaded. If the type is nomove, then the timer resets each time the user moves the mouse or the browser registers keyboard input for any element on the page.

You can also apply the optional url attribute. This attribute defines the URL of the specific page that raises the Business event. The Business event is not submitted if the current document's URL does not match the URL parameter.

Finally, you can apply the optional count attribute. This attribute specifies how many times the trigger needs to be matched before the event is generated and sent to the Web Engagement Server.

For more information about the DSL elements, see the [Business Events DSL](#).

## Creating Business Events by Customizing the DSL File

You can edit the **apps\Your application name\resources\dsl\domain-model.xml** and add a list of events, with specific conditions, related to your web pages' content.

### Important

Genesys recommends that you use the **InTools application** to help you modify your DSL.

The default **domain-model.xml** file provides the Timeout-30 event and a template for Search events to help you get started with your customizations. The following sections show how you can customize these events for use on your website.

### Using the SearchEvent Template

By default, the **domain-model.xml** file contains commented code that you can implement to trigger a business event when a visitor tries to search for something on your website. Complete the following steps to customize the SearchEvent for your website.

#### Start

1. Remove the comment characters that wrap around the event: `<!--` and `-->`. The event should look like the following:

```
<event name="Search">
  <trigger name="SearchTrigger" element="" action="click" url="" count="1" />
  <val name="searchString" value="" />
</event>
```

2. Set the **element** attribute to the jQuery selector that triggers a search. For example, we have an input (`id="search"`) with a submit button (`id="search-submit"`).

```
<event name="Search">
  <trigger name="SearchTrigger" element="#search-submit" action="click" url=""
count="1" />
  <val name="searchString" value="" />
</event>
```

3. Set the **value** attribute to the script to retrieve the search string. For example, our input id of "search".

```
<event name="Search">
  <trigger name="SearchTrigger" element="#search-submit" action="click" url=""
count="1" />
  <val name="searchString" value=$("#search).val()" />
</event>
```

Now the search event is triggered when a visitor clicks the **search-submit** button.

#### End

### Using the Timeout Events

By default, the **domain-model.xml** file contains the timeout-30 event.

```
<event id="Timeout-30" name="Timeout-30">
  <trigger name="TimeoutTrigger" element="" action="timer:30000" type="timeout" url=""
count="1" />
</event>
```

By default, this event is triggered 30 seconds after the tracking script is initialized on the page.

### Creating Business Events by Using the Monitoring Agent API

You can also use the [Monitoring JS API](#), which allows you to submit events and data from the HTML source code.

In this case, you can use the `_gt.push()` method which allows you to decide when events should be submitted and which data they generate, directly from your web pages. See [Monitoring JS API Reference](#) for further details.

You should also consider using the API when you have more complex logic that can't be handled by DSL alone. For an example, see [How To — Enable a trigger after another trigger](#).