



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.

Quick Start Guide

Getting Started with the Playground Application

12/15/2025

Getting Started with the Playground Application

Contents

- **1 Getting Started with the Playground Application**
 - 1.1 Installing Genesys Web Engagement
 - 1.2 Running the Playground Application

The fastest way to get started with Genesys Web Engagement is to use the Playground application, which is included in the Genesys Web Engagement installation package. After you install GWE, you can find the GWE application in the ***GWE_installation_directory\apps\playground*** folder. This application contains all necessary resources to work with the Playground site. The site itself is found in the ***GWE_installation_directory\tools\playground*** folder.

All you need to do is start GWE Server(s), deploy the *playground* GWE application by using the command ***GWE_installation_directory\deploy.bat*** (or ***deploy.sh*** on Linux) and start the playground site by using the command ***GWE_installation_directory\tools\playground\playground.bat*** (or ***playground.sh*** on Linux). Now you are ready to work with the features of Genesys Web Engagement.

Keep reading to learn more about the application, or go directly to "Installing Genesys Web Engagement" for details about installing and running the Playground application.

- **Read more — Playground application**

The Playground application not only includes a Web Engagement application, but also a functioning sample website you can use to test and demonstrate Web Engagement's functional capabilities. This allows you to see Genesys Web Engagement at work on a website that is designed to highlight the GWE features and show you how to use them.

English | Francais

PlayGround

Start page

Change instrumentation script ?≡

Widgets customisation ?≡

Reactive engagement

Get pacing state for chat ?≡

Chat with pacing true ?≡

Chat without pacing ?≡

System events

Page Entered ?≡

Page Exited ?≡

Proactive engagement

Singleton	Engage!
Timeout 30	Engage!
Sequence	Click here first → Click here to
Sequence Ads	Click here first → Click here to
Set	Click ↔
Counter (3)	Count to Engage
Simple search	<input type="text" value="proactive engagement"/>
Search with category	<input type="text" value="What is"/>

The Playground application website.

Browser Support

The Playground website was created using Bootstrap and works on all web browsers that support Bootstrap. See the Bootstrap documentation for details: <http://getbootstrap.com/getting-started/#support>

- **Read more — GWE components**

Genesys Web Engagement Components

You should be familiar with the following Genesys Web Engagement components before you dive into the Playground application:

- [DSL File](#)
- [Categories](#)
- [Instrumentation Script](#)
- [Rules Files](#)
- [Strategies](#)
- [Widgets](#)
- [Configuration Options](#)
- [Monitoring Agent API](#)

DSL File

The DSL file is used to manage and customize business events. 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 (read about the structure of these events [here](#)) 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 id="" name="">
    </event>
    -->

    <!-- This is template for your search event -->
    <!--
    <event id="SearchEvent" name="Search">
      <trigger name="SearchTrigger" element="" action="click" url="" count="1" />
      <val name="searchString" value="" />
    </event>
    -->
    <event id="TimeoutEvent30" name="Timeout-30" condition=""
postcondition="document.hasFocus() === true">
      <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 id="SearchEventClick" 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 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>
```

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 id="TimeoutEvent10" 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.

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](#).

Categories

You can use the GWE Plug-in for Genesys Administrator Extension to define, in a few clicks, Web Engagement categories that contain business information related to URL or web page titles. These categories are used in the CEP rule templates, which provide rules that define when to submit actionable events to Web Engagement — this is what starts the engagement process.

For example, let's look at Solutions on the Genesys website. In this scenario, you can define a Solution category associated with the <http://www.genesys.com/solutions> page and several or all solution sub-pages, such as <http://www.genesys.com/solutions/cloud> or <http://www.genesys.com/solutions/enterprise-workload-management>.

- To associate the category with all the pages containing the "solutions" string in the URL, you can create the "solutions" tag. This tag defines the "solutions" string as a plain text expression to search in the events triggered by the visitor browsers.
- To set up a specific list of sub-pages for the Solutions category, you can create a tag for each sub-page:
 - The "cloud" tag, which defines the "cloud" string as the plain text expression to search in the events triggered by the visitor browsers.
 - The "enterprise-workload" tag, which defines the "enterprise-workload-management" string as the plain text expression to search in the events triggered by the visitor browsers.

Now your rules can use this category to match solution-related pages.

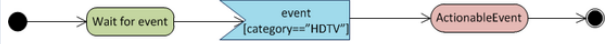
Instrumentation Script



The Tracker Application instrumentation script is a small piece of JavaScript code that you paste into your website to enable Web Engagement functionality. You can copy the [basic instrumentation script](#) from the documentation and then modify it according to the [additional options](#) described in the documentation.


Rules Files

Genesys Rules System (GRS) provides the ability to develop, author, and evaluate business rules. A business rule is a piece of logic defined by a business analyst. These rules are evaluated in a Rules Engine based upon requests received from client applications such as Genesys Web Engagement. GRS implements the CEP (Complex Event Processing) template for GPE. This template type enables rule developers to build templates that rule authors then use to create rules and packages. These rules use customized event types and rule conditions and actions. Each rule condition and action includes the plain-language label that the business rules author will see, as well as the rule language mapping that defines how the underlying data will be retrieved or updated.

The Playground application has a pre-built rule file, called **rule.com.playground.drl**, located at **GWE_installation_directory\apps\playground\resources\drl**. The rules included in this file demonstrate the basic concepts of how Genesys rule files are used.

Singleton	
"Rule-101 singleton"	<p>The rule receives each single event as a formal parameter. If the event's value matches the right category, then the actionable event is sent to the Web Engagement Server.</p> 
Expression Example	<p>When page transition event occurs that belongs to category \$category</p> <p>Then generate actionable event</p>

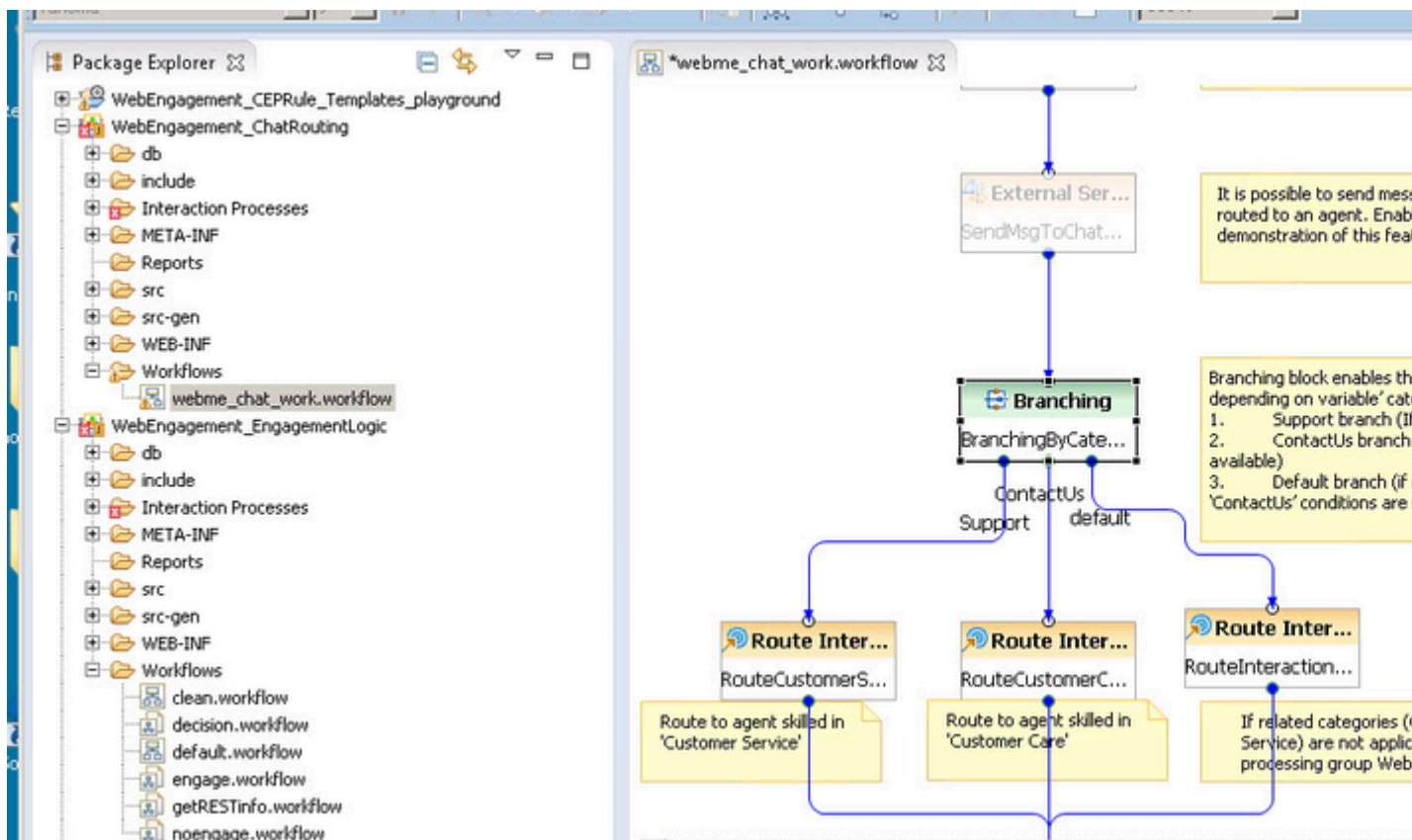
Sequence	
"Rule-108 sequence"	<p>This rule analyses the event stream received from the categorization engine and builds the sequence of events by category values. As soon as the event sequence is completed, the actionable event is submitted. Note that the event sequence must follow a specific order.</p> 
Expression Example	<p>When page transition event occurs that belongs to category \$category1 save as \$event1</p> <p>and event following \$event1 with category \$category2 save as \$event2</p> <p>(...) and event following \$eventⁿ⁻¹ with category \$categoryⁿ save as \$eventⁿ</p> <p>Then generate actionable event based on \$eventⁿ</p>
Set	
"Rule-136 Set"	<p>This rule analyses the event stream received from the categorization engine and collects the events by category values. As soon as the event set is completed, the actionable event is submitted. If you use this rule, the event order is not taken into account.</p> 
Expressions	<p>When page transition event occurs that belongs to category \$category1</p> <p>or page transition event occurs that belongs to category \$category1</p> <p>(...) or page transition event occurs that belongs to category \$categoryⁿ</p> <p>Then generate actionable event</p>
Counter	

<p>"Rule-122 counter"</p>	<p>This rule analyses the event stream received from the categorization engine and counts events which occur for a given category. As soon as the counter is reached, the actionable event is submitted.</p> 
<p>Expressions</p>	<p>When Category \$category counts \$count times</p> <p>Then generate actionable event</p>

Strategies

When you create your application, Genesys Web Engagement also creates default chat routing and engagement logic strategies in the `\apps\application_name\resources\composer-projects\` folder. Orchestration Server (ORS) uses these strategies to decide whether and when to make a proactive offer, taking into account resources availability (pacing feature). You can modify these strategies by importing them into Composer.

The following shows the Chat Routing workflow, where interactions are routed to agents with "Customer Service" or "Customer Care" skills:



A Chat Routing workflow example.

When you alter the strategies, you must save your changes, generate the code, and redeploy your application on a running

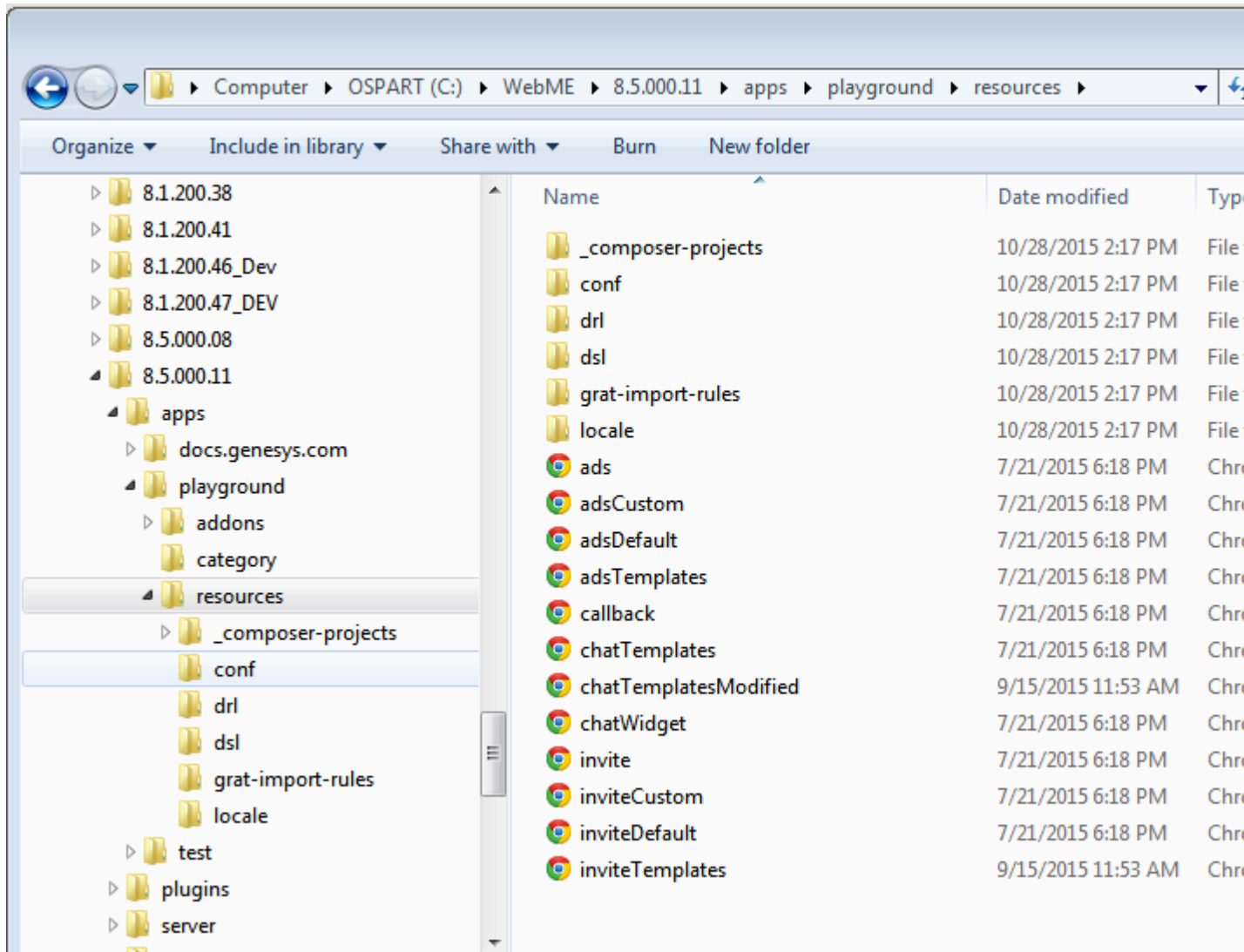
instance of Genesys Web Engagement server to apply the changes.

Widgets

Genesys Web Engagement includes three pre-integrated *legacy* Web Engagement Browser Tier widgets:

- Invitation Widget
- Chat Widget
- Advertisement Widget

These widgets are based on HTML, CSS, and Javascript, and can be customized to suit the look and feel of your website. They are located in the folder ***GWE_installation_directory*\apps\application_name\resources**.



The Browser Tier Widgets.

You can customize the widgets using the HTML files in this directory, CSS, or the [Notification Service API](#). You can also build your own version of a particular widget with the help of the [Web Engagement APIs](#).

Configuration Options

For a full list of configuration options for the Web Engagement servers, see [Configuration Options](#).

Monitoring Agent API

The Monitoring Agent API is available through the JavaScript libraries provided by the Web Engagement Server at runtime. By using this API in your web pages, you can submit events to the Genesys Web Engagement Server, independently from the set of events and conditions defined in the DSL files loaded by the browser's Monitoring Agents. For more information, see the [Monitoring Agent API documentation](#).

Installing Genesys Web Engagement

In order to use the Playground application, you must first install and configure Genesys Web Engagement and its related components, start Web Engagement Server and then deploy the Playground Application.

For details, see the steps below:

1. Review the [prerequisites](#) and make sure your Genesys environment meets the requirements.
2. [Install the Genesys Web Engagement Servers](#). These procedures describe how to install and configure the Web Engagement Server cluster and nodes.
3. Make sure you have properly configured agents and add agent shortcuts to the Web Engagement Chat agent group: In Genesys Administrator, navigate to **Provisioning > Accounts > Agent Group**, and select Web Engagement Chat. Go to the agent tab and click **Add** to add the agent to the group.
4. [Install the Plug-in for Workspace Desktop Edition](#). You can install this plug-in if you want to enable chat engagement features so you can see Web Engagement from the agent's perspective when you test the Playground Application.
5. Go to the **tools\playground\static\html** folder and open the **instrumentation.html** file in an editor. Modify the following lines to replace `demosrv.genesyslab.com` with the fully qualified domain name of your host:
 - `var serverDefault = 'http://demosrv.genesyslab.com:9081';`
 - `var serverSecure = 'http://demosrv.genesyslab.com:9081';`
 - `domainName: 'demosrv.genesyslab.com',`Save your changes and close the file.
6. [Deploy an Application](#). To complete these procedures, make sure you use the application name **playground**.
7. Make sure the components in your Genesys environment are running:
 - [Configuration Server](#)
 - [Universal Routing Server](#)
 - [Interaction Server](#)
 - [Orchestration Server](#)
 - [Stat Server](#)

- Chat Server
 - Media Server
 - Interaction Workspace
8. Start the Web Engagement Servers.
 9. Start the playground site using the command **playground.bat** (**playground.sh** on Linux) located in the folder **GWE_installation_directory\tools\playground**.
 10. Log in any agents from the Web Engagement Chat agent group.

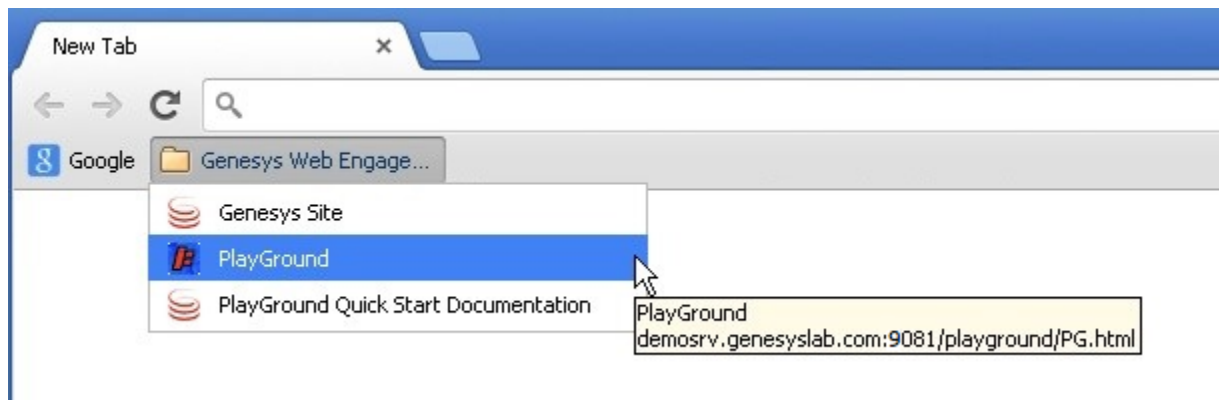
Congratulations! You are ready to run the Playground application.

Running the Playground Application

You can now run the application in any web browser or the **InTools** application — a GWE tool that allows you to work with the DSL and view events.

To launch InTools, navigate to **GWE_installation_directory/tools/intools/**. Take the **intools.crx** file and drop it into the Extensions page of your Google Chrome or Chromium browser. Click the **Genesys Web Engagement** bookmark in the toolbar and select **Playground** to launch the Playground application.

IMPORTANT: If your web browser blocks the installation of extensions, you can try the steps described in [InTools installation for Chrome \(Windows\)](#).



You can access the Playground application directly from InTools.

The Playground is an external application located at **GWE_installation_directory/tools/playground/**. You can configure it in the application.properties. For example, to change the port configuration, you would modify the value of the **server.port** parameter (default is 8081).

To open the Playground application from the **GWE_installation_directory/tools/playground/** folder, run the **playground.bat** file.

To open the Playground application from a web browser, navigate to `http://playground_host_name:playground_port/playground/PG.html`.

- *playground_host_name* — The name or IP address of the host where the Playground application is running.
- *playground_port* — The default port of the Playground application (can be configured in `application.properties`).

For example, `http://127.0.0.1:8081/playground/PG.html`

Example: Web Chat End-to-End

Complete the steps below for an example of an end-to-end scenario for proactive chat.

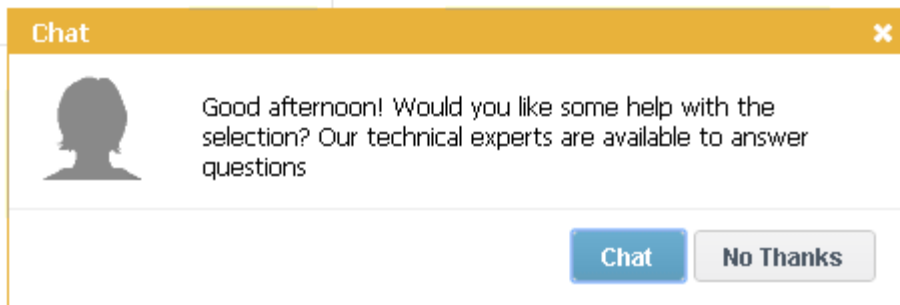
1. Make sure the Playground application is open and you have a logged in and ready agent in the Web Engagement Chat agent group.
2. Click **Engage!** next to the Singleton business rule.

Proactive engagement

Singleton	Engage!
Timeout 30	Engage!
Sequence	Click here first → Click here to Engage
Sequence Ads	Click here first → Click here to Ads
Set	Click ↔ Click
Counter (3)	Count to Engage
Simple search	<input type="text" value="proactive engagement"/> <input type="button" value="Search"/>
Search with category	<input type="text" value="What is"/> <input type="button" value="Search"/>

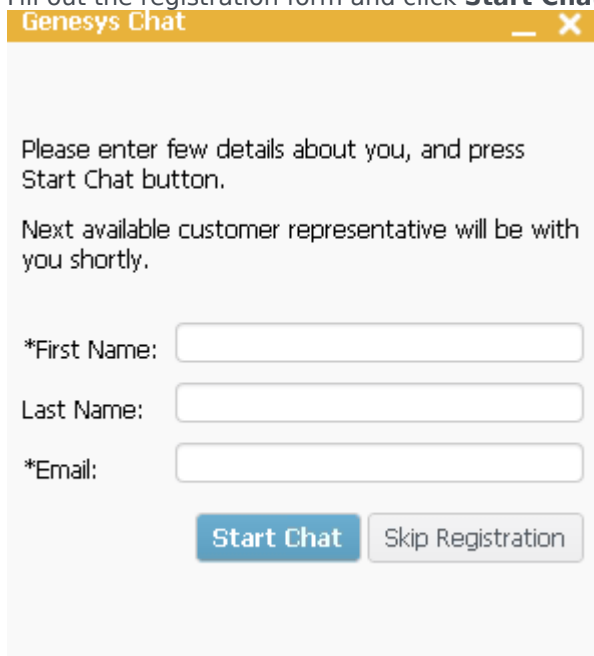
Click **Engage!**

3. The chat pop-up appears. Click **Chat**.



Click **Chat** in the chat pop-up.

4. Fill out the registration form and click **Start Chat**.



Please enter few details about you, and press Start Chat button.

Next available customer representative will be with you shortly.

*First Name:

Last Name:

*Email:

Click **Start Chat** after filling out the registration form.

5. On the agent side, accept the chat. You just walked through your first proactive engagement!

Next Steps

- [Working with the Playground Application](#)