Genesys Knowledge Center Developer's Guide

Integrating with Genesys Web Engagement

5/3/2025

Integrating with Genesys Web Engagement

Overview

When you integrate Knowledge Center with Genesys Web Engagement, you are giving your agents access to important proactive engagement capabilities. Knowledge Center (and the way you interact with) it allows you to better understand your customer needs and intentions. For example, monitoring customer activities with Knowledge Center on the corporate web site allows you to find the right moment to propose agent help when the customer appears to be lost. When such an interaction appears on an Agent workspace, all the customer requests and browsing history are made available. This is one of the many reasons why you might want to integrate Knowledge Center with Genesys Web Engagement in your environment.

Tight integration between Knowledge Center and Web Engagement allows you to monitor customer activities on your web site (both browsing and working with knowledge). It also defines customer behavior patterns and actions that should take place when patterns occur (including both immediate contact with an agent or postponed processing of the activity).

Here are some examples of the patterns you could look for and suggested reactions:

- Customer indicates that they cannot find the answer to the question. A suggested reaction for this event is the chat option with the agent (how to configure such integration is shown in the example below).
- A Premium customer has left negative feedback on one of the documents he viewed. A suggested reaction for this event is a follow-up call to maintain the relationship with the customer.
- While browsing throughout the site a customer has expressed interest in establishing a new service with the company. A suggested reaction for this event is to do a follow-up and check whether or not the customer has successfully set-up the new service and then send a note of thanks for being a loyal customer.

To integrate products in your environment you need to add Knowledge Center-specific events into the Web Engagement DSL file which describes business events for a given website. All other steps are standard for installation of Genesys Knowledge Center and Genesys Web Engagement.

Sample DSL

KnowledgeCenter.DSL provides a basic set of events that are used in your integration. Events are based on the Sample UI GUI shipped with the product.

DSL file contains following events:

- Open a category in browsing

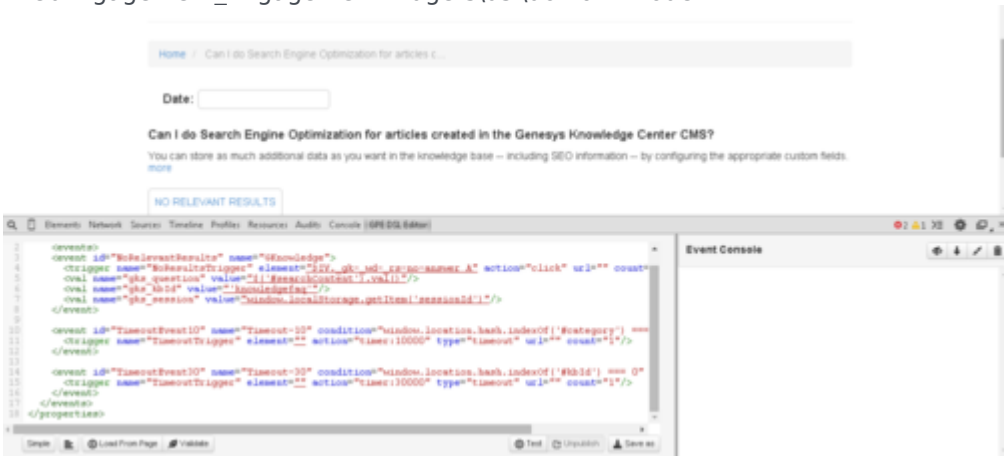
- Viewing of search results
- Open document for viewing content
- Leaving positive and negative feedback
- Requesting additional help (no aster found)

Engaging chat with agent when no answer found

Follow the instructions below to configure this integration.

Start

1. Install and properly configure Genesys Web Engagement, using the GWE Deployment Guide.
2. Create a Knowledge Center application in GWE.
3. Create a DSL file that describes your site's business logic. You can either use the **Intool** provided with GWE or use the standard DSL for the Sample UI that is provided with Knowledge Center. Replace the standard GWE content by the new DSL that is included at *GWE root folder*\apps\gks_composer-project\WebEngagement_EngagementWidgets\dsl\domain-model.xml.

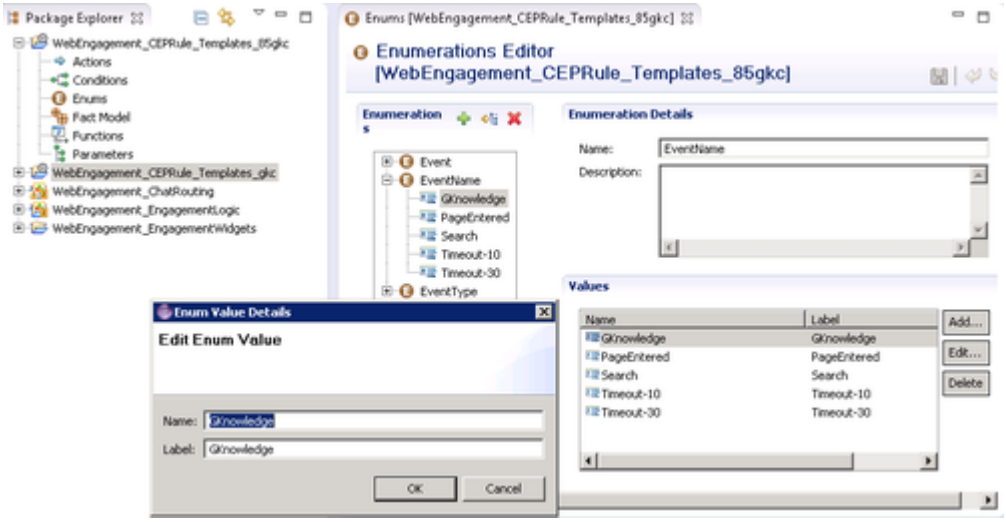


GWE Integration Tool

Here is a sample DSL file:

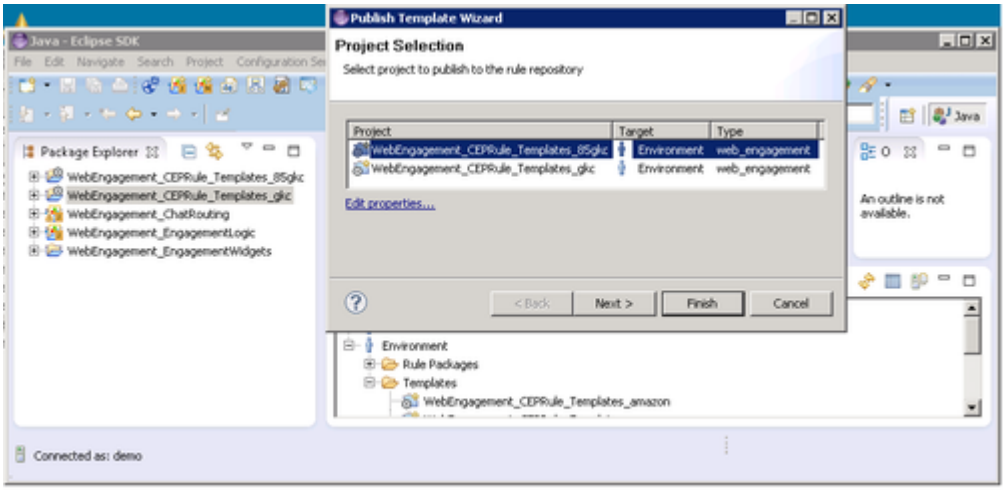
```
<?xml version="1.0" encoding="utf-8"?>
<properties>
     <events>
        <event id="NoRelevantResults" name="GKnowledge">
           <trigger name="NoResultsTrigger" element=
"DIV._gk-_wd-_rs-no-answer A" action="click" url="" count="1"/>
           <val name="gks_question" value="$
('#searchContent').val()"/>
           <val name="gks_kbId" value="'knowledgeFAQ'"/>
           <val name="gks_session" value=
"window.localStorage.getItem('sessionId')"/>
        </event>
     </events>
</properties>
```

4. In Composer, modify the Web Engagement templates, which will be either **WebEngagement_CEPRule_Templates** (if you use GRAT 8.1.3) or **WebEngagement_CEPRule_Templates_85** (if you use GRAT 8.5).
   Add new event names to the Enums. In the above example, we used an event name of *GKnowledge*.



Editing an Enum Value

5. Publish **CEPRule_Templates** to the GRS repository.

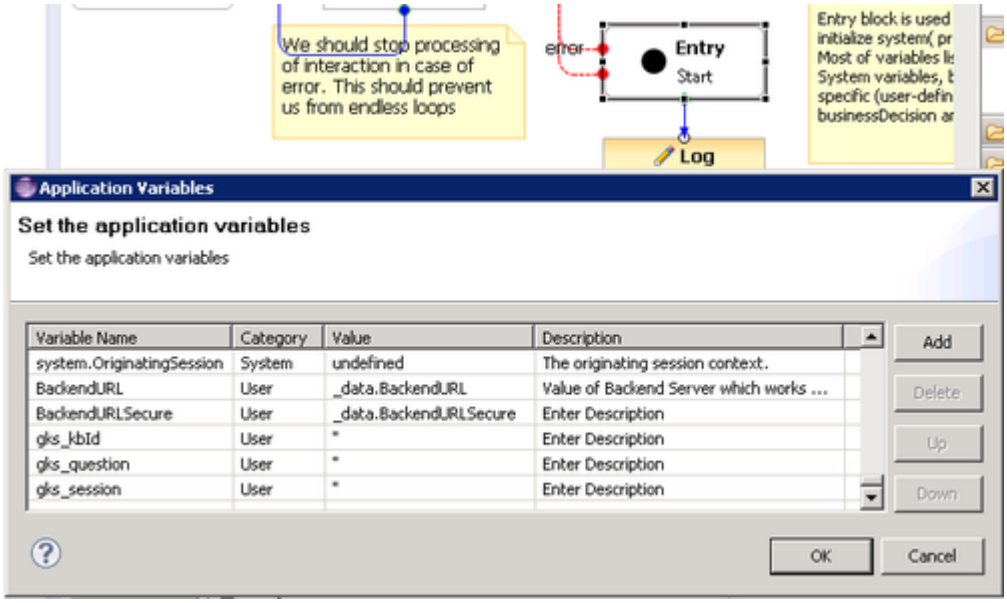


Publishing a Project

6. Create a business rule based on your custom DSL and on **CEPRule_Templates**. For example:

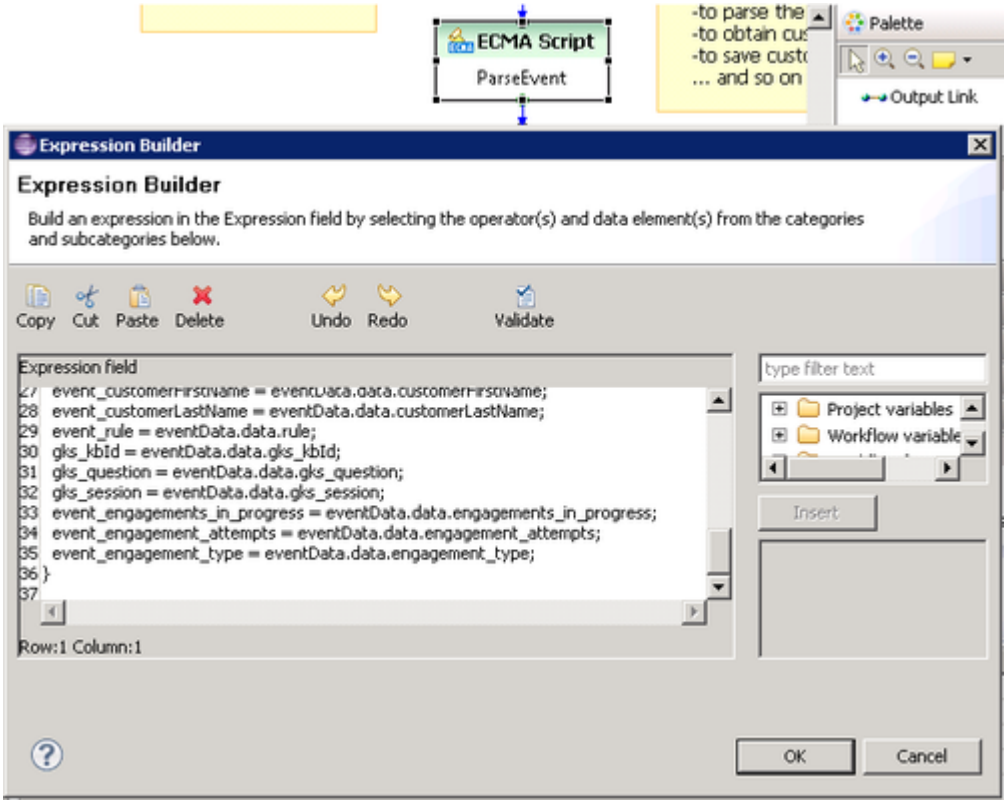
```
rule "Rule-100 No Relevant Results"
salience 100000
    agenda-group "level0"
    dialect "mvel"
    when
        $event1: Event(eval($event1.getName()
.equals('NoRelevantResults')))
    then
        sendEvent($event1, ed, drools);
end
```

7. Modify **default.workflow** in the **WebEngagement_EngagementLogic** Composer project.
   Add new user variables, **gks_kbId**, **gks_question**, and **gks_session**, to the **Entry (Start)** block:



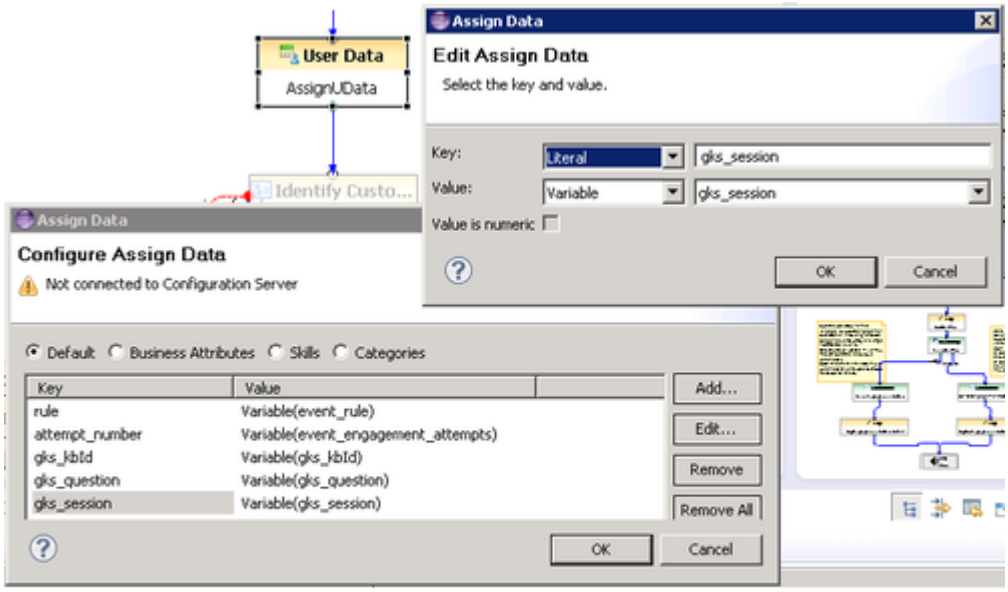
Adding New Variables

8. Add parsing for new variables to the **ECMA Script (ParseEvent)** block:



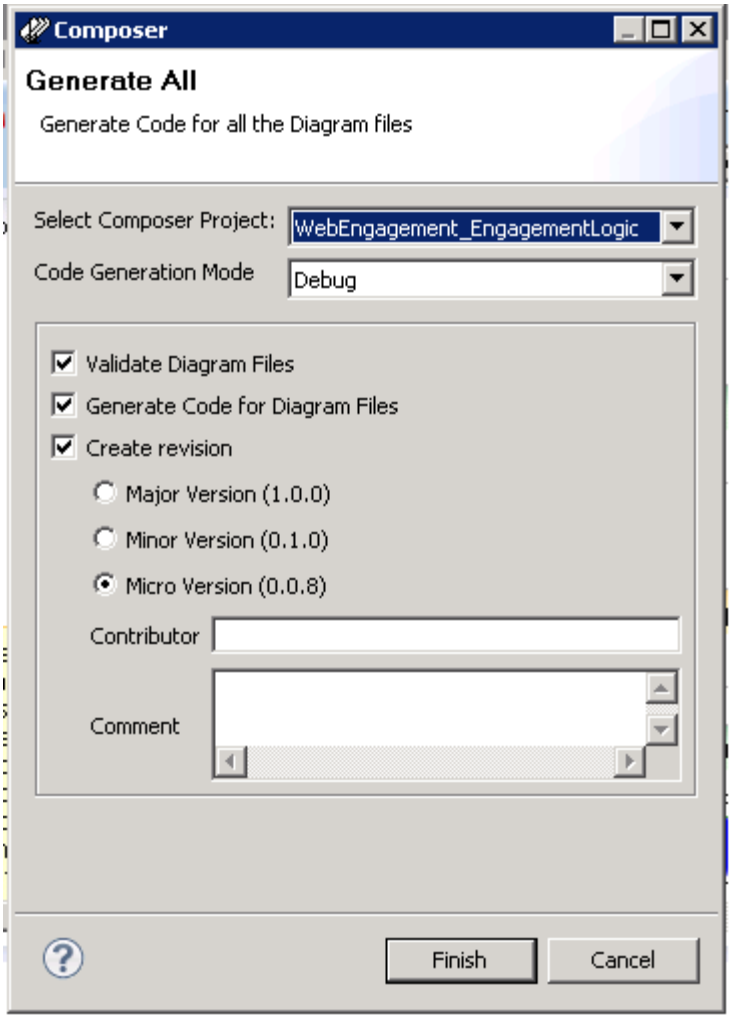
ECMA Script for Event Parsing

9. Add parsed data to the interaction in the **User Data (AssignUData)** block:



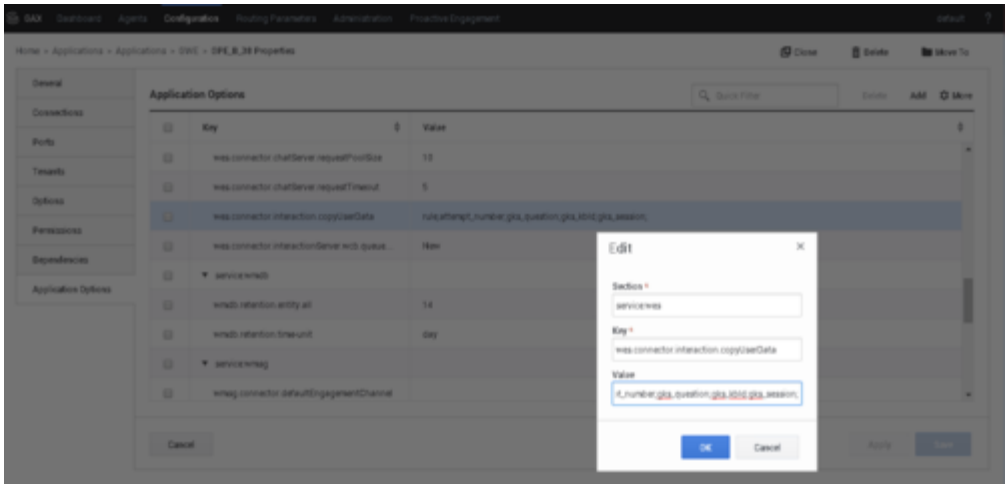
Add Parsed Data to Interaction

10. Save **default.workflow** and generate new SCXML strategies by clicking the **Generate All** button:



Generate SCXML Strategies

11. Build the Knowledge Center Server application (run **build gks**).
12. Deploy the Knowledge Center Server application (run **deploy gks**).
13. Modify the GWE backend Config Server application. Add new variables, **gks_question**, **gks_kbId**, and **gks_session**, to the **wes.connector.interaction.copyUserData** option.



Add Options to GWE Backend Server

14. Deploy the business rule created in Step 6, above, to GWE storage.
15. Run the GWE servers.

End

To allow GWE to access the Knowledge Center UI, you need to modify either your site or the Sample UI by adding a Web Monitoring Agent script similar to the following sample to the source code of your web UI application.

```
<script>
    var _gt = _gt || [];
    _gt.push(['config', {
       dslResource : ('https:' == document.location.protocol
? 'https://<host>:<port1>' : 'http://<host>:<port2>')
+ '/server/resources/dsl/domain-model.xml',
       httpEndpoint : 'http://<host>:<port2>',
       httpsEndpoint : 'https://<host>:<port1>'
    }]);

    var _genesys = {
       chat: {
          serverUrl: 'http://<host>:<port3>/backend/cometd',
          registration: true
       },
      embedded:true,
      onReady: []
    };

    (function(d, s, id, o) {
        var fs = d.getElementsByTagName(s)[0], e;
        if (d.getElementById(id)) return;
        e = d.createElement(s); e.id = id; e.src = o.src;
        e.setAttribute('data-gcb-url', o.cbUrl);
        fs.parentNode.insertBefore(e, fs);
    }) (document, 'script', 'genesys-js', {
       src:
"http://<host>:<port2>/server/resources/js/build/genesys.min.js",
    });
</script>
```

> **Important**
>
> To make the integration work, you need to run both the GWE backend and frontend servers.

For more detailed instructions, refer to the GWE documentation.