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.

Genesys Knowledge Center Developer's Guide

Knowledge Center 8.5.3

2/11/2022

Table of Contents

Genesys Knowledge Center Developer's Guide

Welcome to the Genesys Knowledge Center Developer’s Guide. This document provides information about how you can integrate Knowledge Center for your website and environment. See the summary of chapters below:

Simple Integration for Pre-Production Environment

Overview

This chapter describes the integration steps that allows you to add Knowledge Center functionality to your site without modifying any code. To configure the integration you need to use Proxy shipped with Genesys Web Engagement product.

The GWM Proxy is a development tool that you use to add new functionality to a website without directly modifying that site. Once you have configured this proxy, you can use the Genesys Knowledge Center Sample UI from any of your websites. In a few clicks, without modifying your website, the Knowledge Center Sample UI features shows up on a set of web pages, according to the rules and categories that you created. There are two proxy tools available in the Web Engagement installation, the Simple tool and the Advanced tool. Within this instruction you need to use the Advanced GWM Proxy. For more information regarding proxy please refer to the Genesys Web Engagement documentation.

> **Important**
>
> GWE Proxy provides support for easy integration into existing sites within the pre-production environment. It is not recommended to use it in a production environment. Please use similar tools available on the market.

Configuring the Advanced GWM Proxy

The Advanced GWM Proxy is based on the OWASP Zed Attack Proxy Project (ZAProxy). In addition to acting as a proxy, the Advanced GWM Proxy also provides a UI and validates vulnerabilities in your website at the same time.

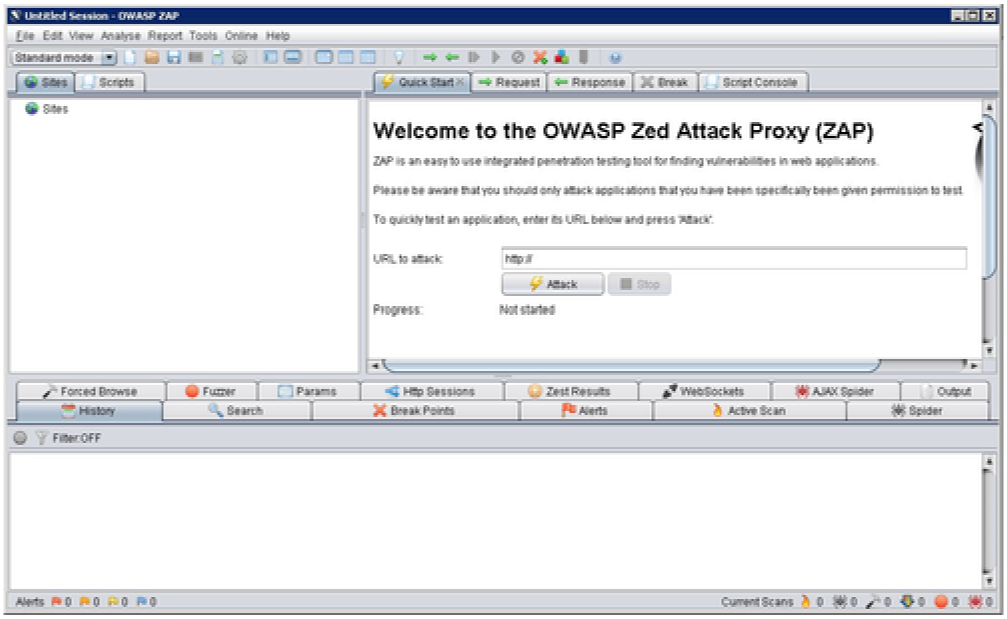
> **Important**
>
> The Advanced GWM Proxy requires JDK 1.7 or higher.

Before you start using the Advanced GWM Proxy, you need to carry out a few configuration procedures.

Starting the Proxy

Navigate to your Web Engagement installation directory and launch either *servers\proxy2\zap.bat* (on Windows) or *servers\proxy2\zap.sh* (on Linux).

The proxy starts.



Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: http://

Progress: Not started

The Advanced GWM Proxy

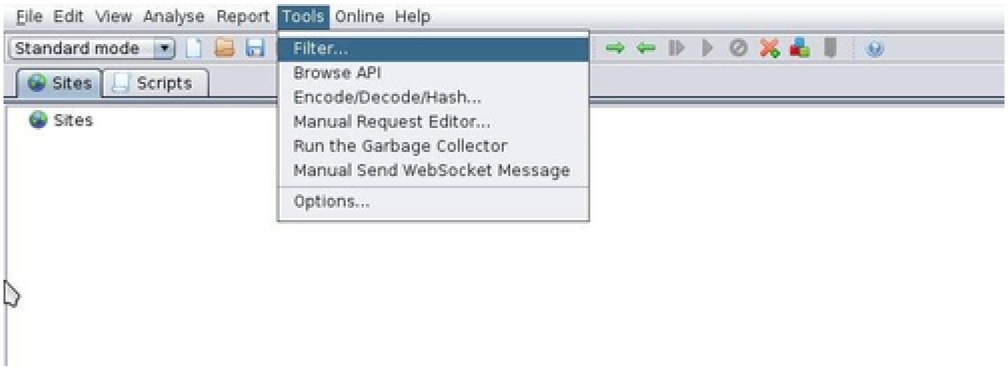
Configuring the Proxy

> **Important**
>
> Below please find an example of integrating the Knowledge Widget via Proxy. The sample source of the widget can be found here: <knowledge_center_server_root>\server\tools\integrations\knowledgewidget

Once the proxy is running, you can configure it using the GUI.

Start

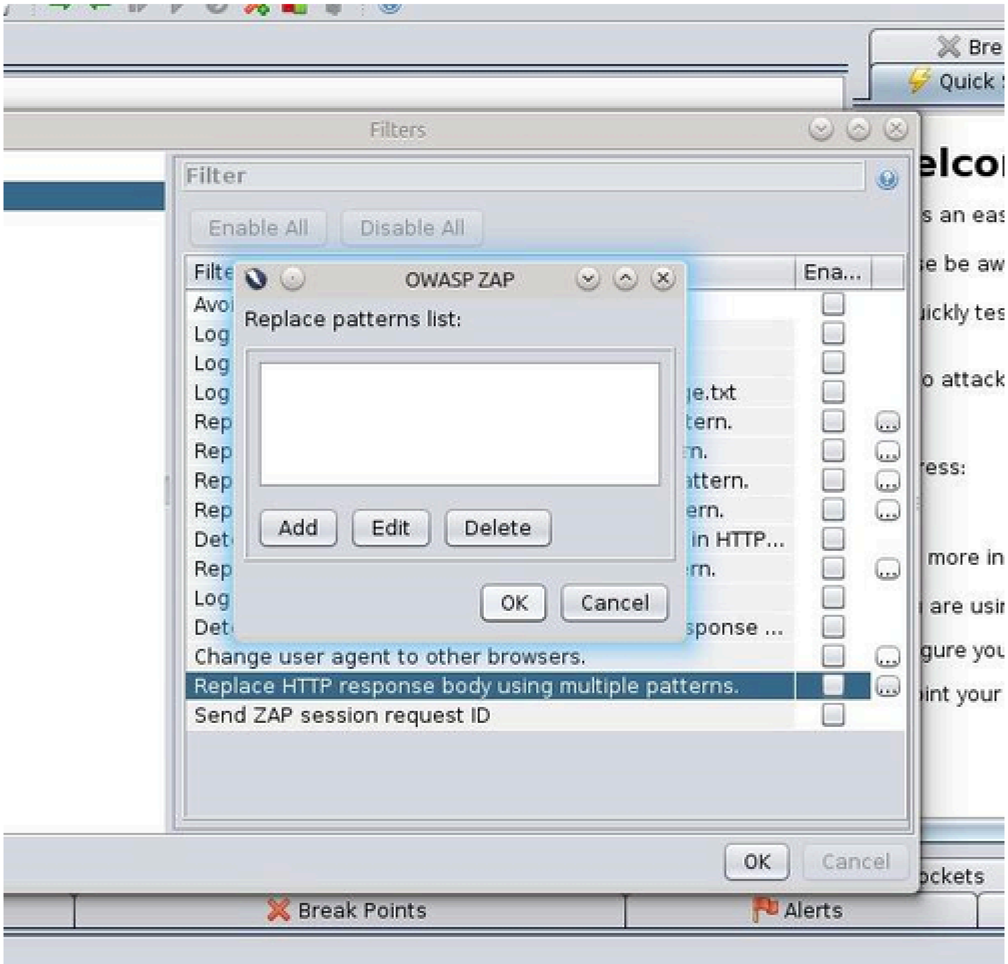
1. Open **Tools > Filter...**.



Configuring the Proxy Filter

Select the Filter menu item.

2. In the list of filters, select **Replace HTTP response body using multiple patterns** and click **...** to edit the filter.



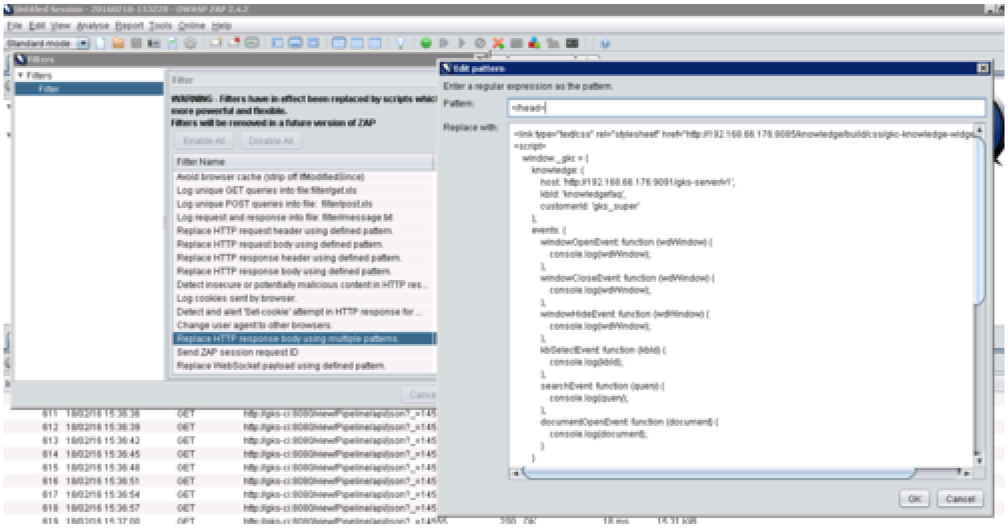
List of Proxy Filters

Select the filter.

3. Click **Add**.
4. Enter **</head>** in the **Pattern:** field of the resulting dialog box.
5. Enter the following code in the **Replace with:** field.

```
<link type="text/css" rel="stylesheet" href="http://
<link_to_resources>/gkc-knowledge-widget.min.css">
<script>
    window._gkc = {
        knowledge: {
            host: 'http://<link_to_server>/gks-server/v1',
            kbId: 'knowledgefaq',
            customerId: 'gks_super'
        },
        events: {
            windowOpenEvent: function (wdWindow) {
                console.log(wdWindow);
            },
            windowCloseEvent: function (wdWindow) {
                console.log(wdWindow);
            },
            windowHideEvent: function (wdWindow) {
                console.log(wdWindow);
            },
            kbSelectEvent: function (kbId) {
                console.log(kbId);
            },
            searchEvent: function (query) {
                console.log(query);
            },
            documentOpenEvent: function (document) {
                console.log(document);
            }
        }
    };

    window._gkcLocalization = {
        title: 'Ask',
        inputPlaceholder: 'Ask a questions',
        trendingMessage: 'Trending questions',
        loading: 'Loading...',
        noResultFound: 'No relevant results found',
        feedback: {
            question: 'Was this helpful?',
            defaultAnswer: 'Thank you for your vote',
            noCommentAnswer: 'Thank you for your vote',
            submitAnswer: 'Thanks, your feedback has been submitted',
            commentPlaceholder: 'Why wasn\'t this helpful?',
            buttons: {
                yes: 'Yes',
                no: 'No',
                submit: 'Submit',
                noComment: 'No comment'
            }
        }
    };
</script>
<script src="http://<link_to_resources>/
gkc-knowledge-widget.min.js"></script>
```

6. Click **OK** to save the pattern.


Untitled Session - 20160218-153228 - OWASP ZAP 2.4.2
File Edit View Analyse Report Tools Online Help
Standard mode
Filters
Filters
Filter
Filter
Enable All
Disable All
Cancel
Edit pattern
OK
Cancel

WARNING - Filters have in effect been replaced by scripts which are more powerful and flexible.
Filters will be removed in a future version of ZAP

Filter Name
Avoid browser cache (strip off IfModifiedSince)
Log unique GET queries into file:filter/get.xls
Log unique POST queries into file: filter/post.xls
Log request and response into file: filter/message.txt
Replace HTTP request header using defined pattern.
Replace HTTP request body using defined pattern.
Replace HTTP response header using defined pattern.
Replace HTTP response body using defined pattern.
Detect insecure or potentially malicious content in HTTP res...
Log cookies sent by browser.
Detect and alert 'Set-cookie' attempt in HTTP response for ...
Change user agent to other browsers.
Replace HTTP response body using multiple patterns.
Send ZAP session request ID
Replace WebSocket payload using defined pattern.

Enter a regular expression as the pattern.

Pattern: <head>

Replace with:

```
<link type="text/css" rel="stylesheet" href="http://192.168.66.176:9095/knowledge/build/css/gkc-knowledge-widge
<script>
  window._gkc = {
    knowledge: {
      host: 'http://192.168.66.176:9091/gks-server/v1',
      kbId: 'knowledgefaq',
      customerId: 'gks_super'
    },
    events: {
      windowOpenEvent: function (wdWindow) {
        console.log(wdWindow);
      },
      windowCloseEvent: function (wdWindow) {
        console.log(wdWindow);
      },
      windowHideEvent: function (wdWindow) {
        console.log(wdWindow);
      },
      kbSelectEvent: function (kbId) {
        console.log(kbId);
      },
      searchEvent: function (query) {
        console.log(query);
      },
      documentOpenEvent: function (document) {
        console.log(document);
      }
    }
```

```
611  18/02/16 15:36:36  GET  http://gks-ci:8080/newPipeline/api/json?_=145
612  18/02/16 15:36:39  GET  http://gks-ci:8080/newPipeline/api/json?_=145
613  18/02/16 15:36:42  GET  http://gks-ci:8080/newPipeline/api/json?_=145
614  18/02/16 15:36:45  GET  http://gks-ci:8080/newPipeline/api/json?_=145
615  18/02/16 15:36:48  GET  http://gks-ci:8080/newPipeline/api/json?_=145
616  18/02/16 15:36:51  GET  http://gks-ci:8080/newPipeline/api/json?_=145
617  18/02/16 15:36:54  GET  http://gks-ci:8080/newPipeline/api/json?_=145
618  18/02/16 15:36:57  GET  http://gks-ci:8080/newPipeline/api/json?_=145
619  18/02/16 15:37:00  GET  http://gks-ci:8080/newPipeline/api/json?_=14555  200  OK  18 ms  15.11 KiB
```

Entering a Filter Pattern

7. Click **OK** to save the pattern.

End

Configuring the URL Filter

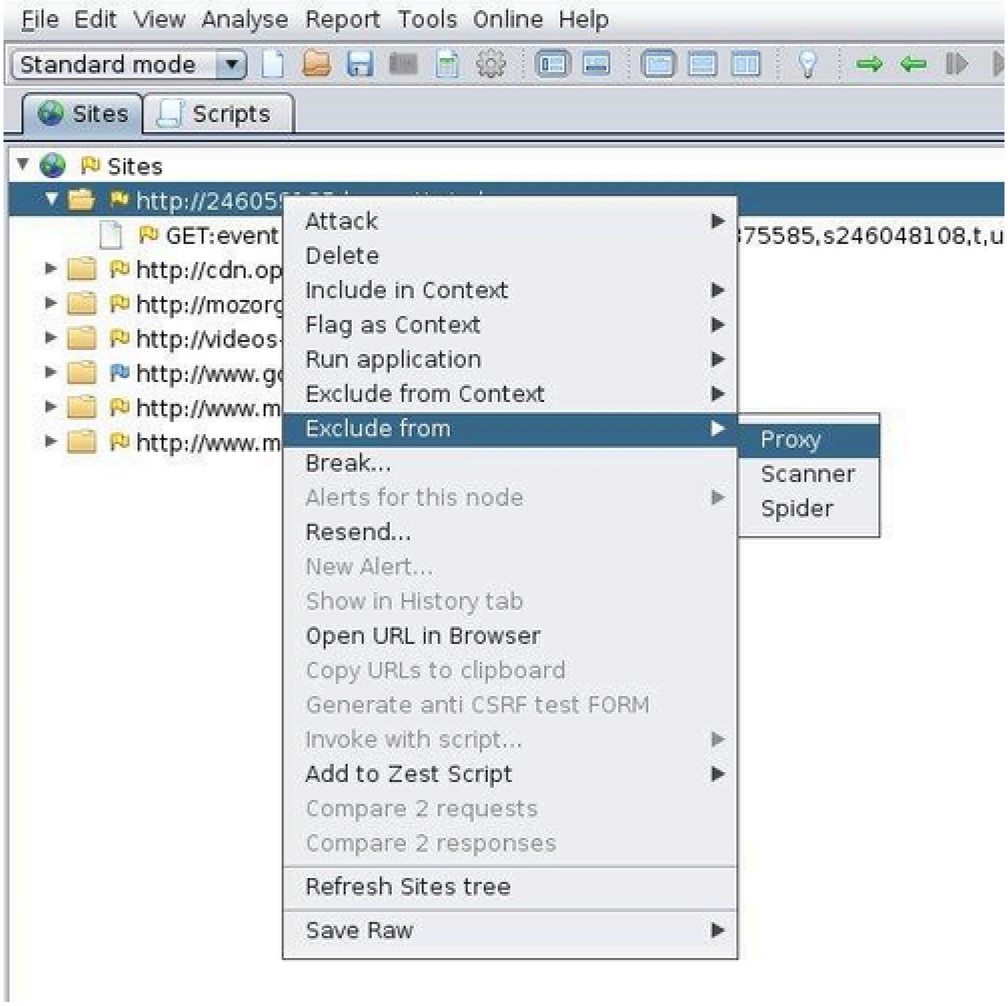
Complete this procedure to use the GUI to configure URLs that the proxy should ignore.

Start

You can exclude a site in one of two ways:

Using the Sites Tab

1. In the **Sites** tab, right-click a site and select **Exclude from > Proxy**.



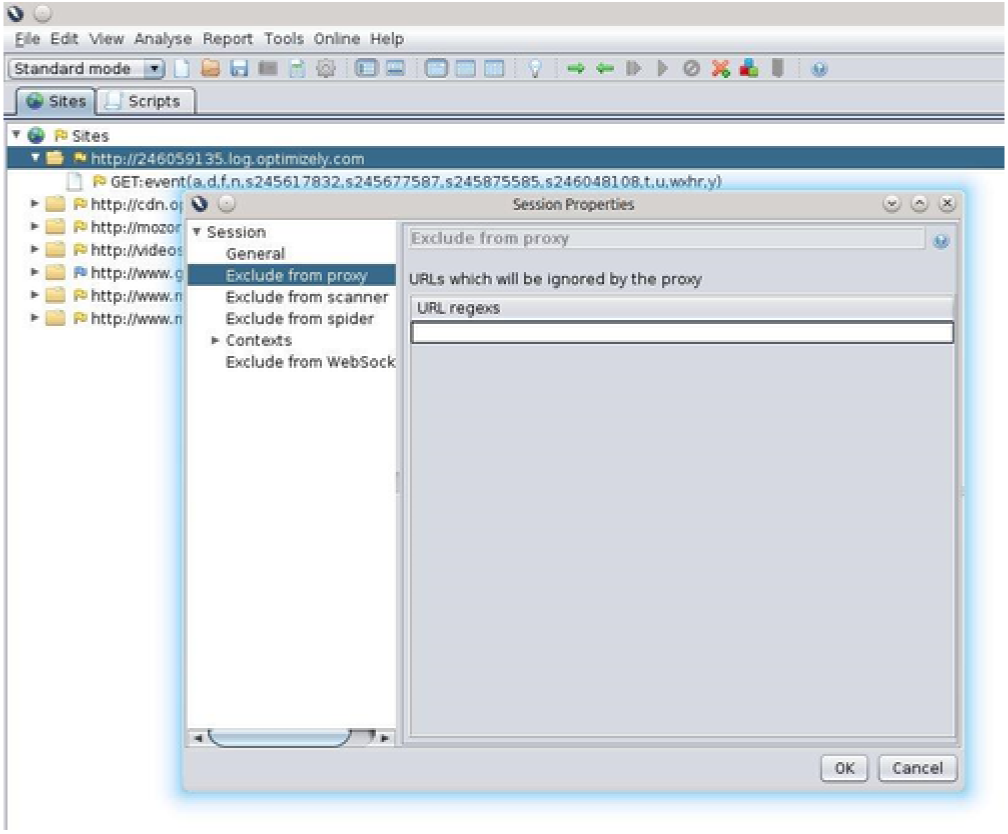
Excluding a Site from the Proxy Filter

2. Select a site to exclude.

Using the File Menu

1. Select **File > Properties**. In the **Session Properties** window, select **Exclude from proxy**, add your URL regular expression, and click **OK**. For example, to have the proxy include only the **google.com** website, use this regular expression:

```
^((?!google.com).)*$
```



Adding a Regular Expression for Ignoring Sites

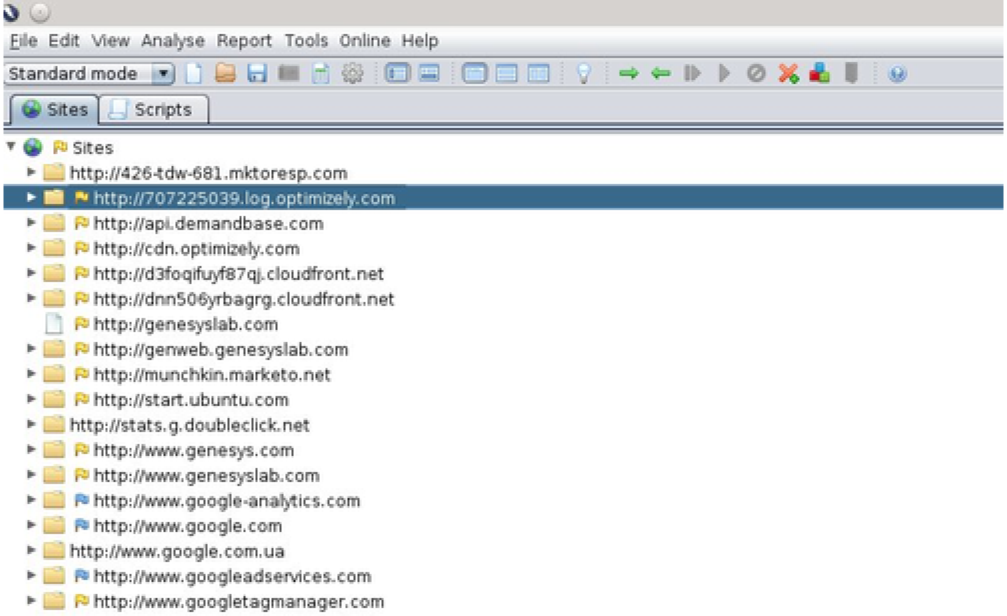
2. Enter a URL to exclude.

If you want the proxy to remember the excluded URLs beyond the current session, select **File > Persist session...** and select a file to save your session to.

End

Working with the Proxy

After you have configured the proxy, keep it open and configure the connection to the network via the Proxy inside you browser. Now you can browse through the web pages that are instrumented with the Knowledge Center Sample UI, and they will be displayed in the **Sites** tab of the proxy GUI, as shown here:



```
Sites
  http://426-tdw-681.mktoresp.com
  http://707225039.log.optimizely.com
  http://api.demandbase.com
  http://cdn.optimizely.com
  http://d3foqifuyf87qj.cloudfront.net
  http://dnn506yrbagrg.cloudfront.net
  http://genesyslab.com
  http://genweb.genesyslab.com
  http://munchkin.marketo.net
  http://start.ubuntu.com
  http://stats.g.doubleclick.net
  http://www.genesys.com
  http://www.genesyslab.com
  http://www.google-analytics.com
  http://www.google.com
  http://www.google.com.ua
  http://www.googleadservices.com
  http://www.googletagmanager.com
```

Browsing Your Proxy Sites

For more information about working with ZAProxy, see https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.

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

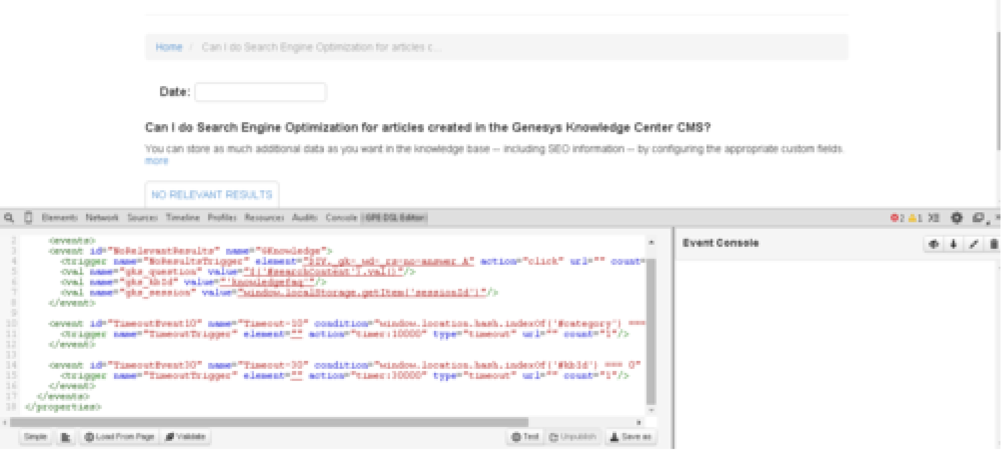
- Viewing of search results
- Open document for viewing content
- Leaving positive and negative feedback
- Requesting additional help (no aster found)

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.


Home / Can I do Search Engine Optimization for articles c...

Date:

Can I do Search Engine Optimization for articles created in the Genesys Knowledge Center CMS?

You can store as much additional data as you want in the knowledge base -- including SEO information -- by configuring the appropriate custom fields.
more

NO RELEVANT RESULTS

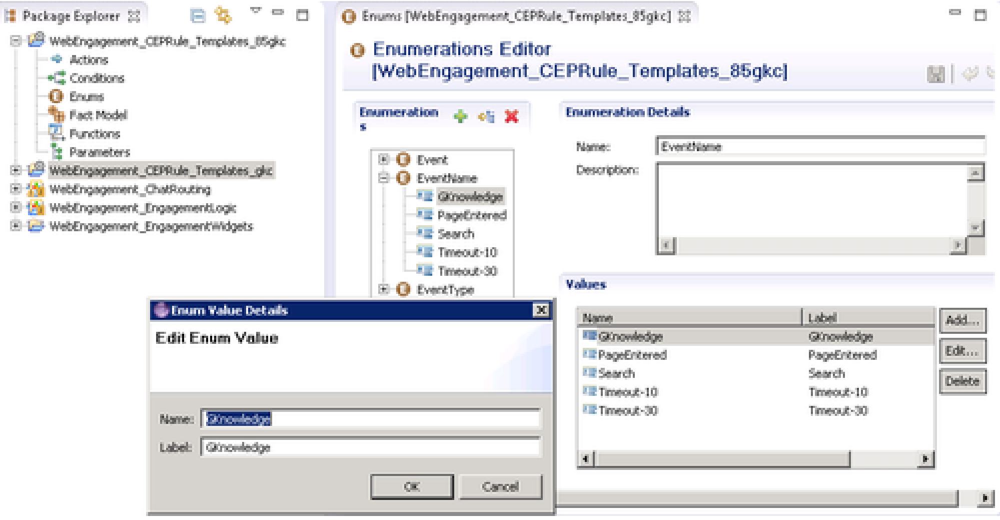
```
    <events>
    <event id="NoRelevantResults" name="GKnowledge">
      <trigger name="NoResultsTrigger" element="DIV._gk-_wd-_rs-no-answer A" action="click" url="" count=
      <val name="gks_question" value="$('#searchContent').val()"/>
      <val name="gks_kbId" value="'knowledgefaq'"/>
      <val name="gks_session" value="window.localStorage.getItem('sessionId')"/>
    </event>

    <event id="TimeoutEvent10" name="Timeout-10" condition="window.location.hash.indexOf('#category') ===
      <trigger name="TimeoutTrigger" element="" action="timer:10000" type="timeout" url="" count="1"/>
    </event>

    <event id="TimeoutEvent30" name="Timeout-30" condition="window.location.hash.indexOf('#kbId') === 0"
      <trigger name="TimeoutTrigger" element="" action="timer:30000" type="timeout" url="" count="1"/>
    </event>
  </events>
</properties>
```

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*.


Package Explorer
WebEngagement_CEPRule_Templates_85gkc
Actions
Conditions
Enums
Fact Model
Functions
Parameters
WebEngagement_CEPRule_Templates_gkc
WebEngagement_ChatRouting
WebEngagement_EngagementLogic
WebEngagement_EngagementWidgets
Enums [WebEngagement_CEPRule_Templates_85gkc]
Enumerations Editor [WebEngagement_CEPRule_Templates_85gkc]
Enumerations
Event
EventName
GKnowledge
PageEntered
Search
Timeout-10
Timeout-30
EventType
Enumeration Details
Values
Add...
Edit...
Delete
Enum Value Details
Edit Enum Value
OK
Cancel

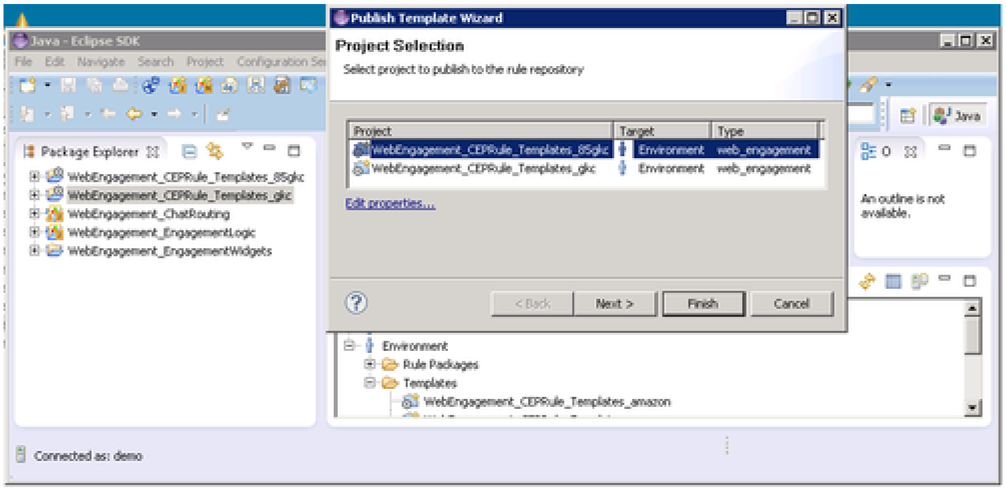
Name: EventName
Description:

Name	Label
GKnowledge	GKnowledge
PageEntered	PageEntered
Search	Search
Timeout-10	Timeout-10
Timeout-30	Timeout-30

Name: GKnowledge
Label: GKnowledge

Editing an Enum Value

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


Java - Eclipse SDK
File Edit Navigate Search Project Configuration Se
Package Explorer
WebEngagement_CEPRule_Templates_85gkc
WebEngagement_CEPRule_Templates_gkc
WebEngagement_ChatRouting
WebEngagement_EngagementLogic
WebEngagement_EngagementWidgets
Publish Template Wizard
Project Selection
Select project to publish to the rule repository
Edit properties...
< Back
Next >
Finish
Cancel
Java
An outline is not available.
Environment
Rule Packages
Templates
WebEngagement_CEPRule_Templates_amazon
Connected as: demo

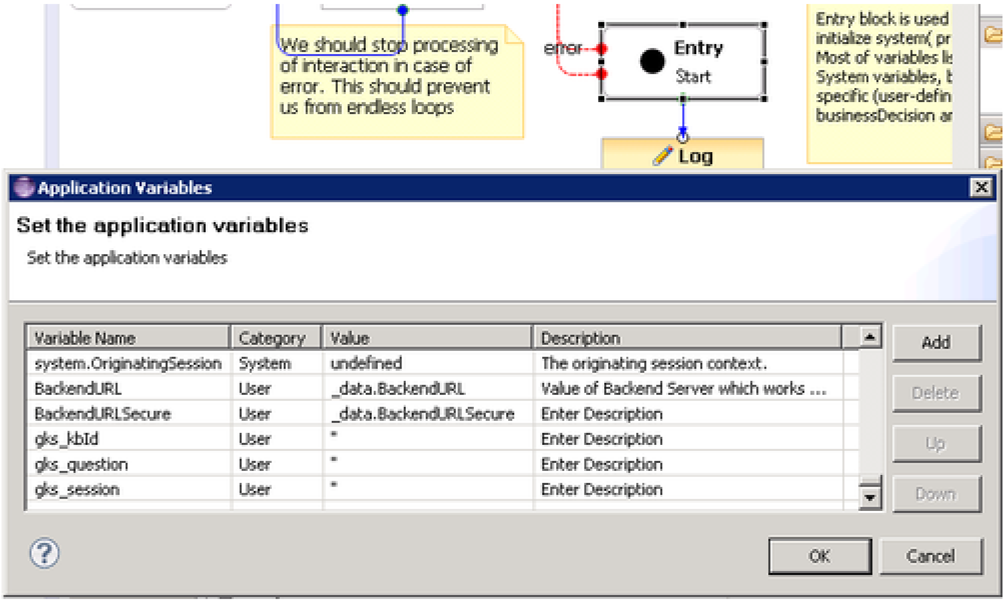
Project	Target	Type
WebEngagement_CEPRule_Templates_85gkc	Environment	web_engagement
WebEngagement_CEPRule_Templates_gkc	Environment	web_engagement

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:


We should stop processing of interaction in case of error. This should prevent us from endless loops
error
Entry
Start
Log
Entry block is used initialize system(pr Most of variables li System variables, b specific (user-defin businessDecision ar

Application Variables

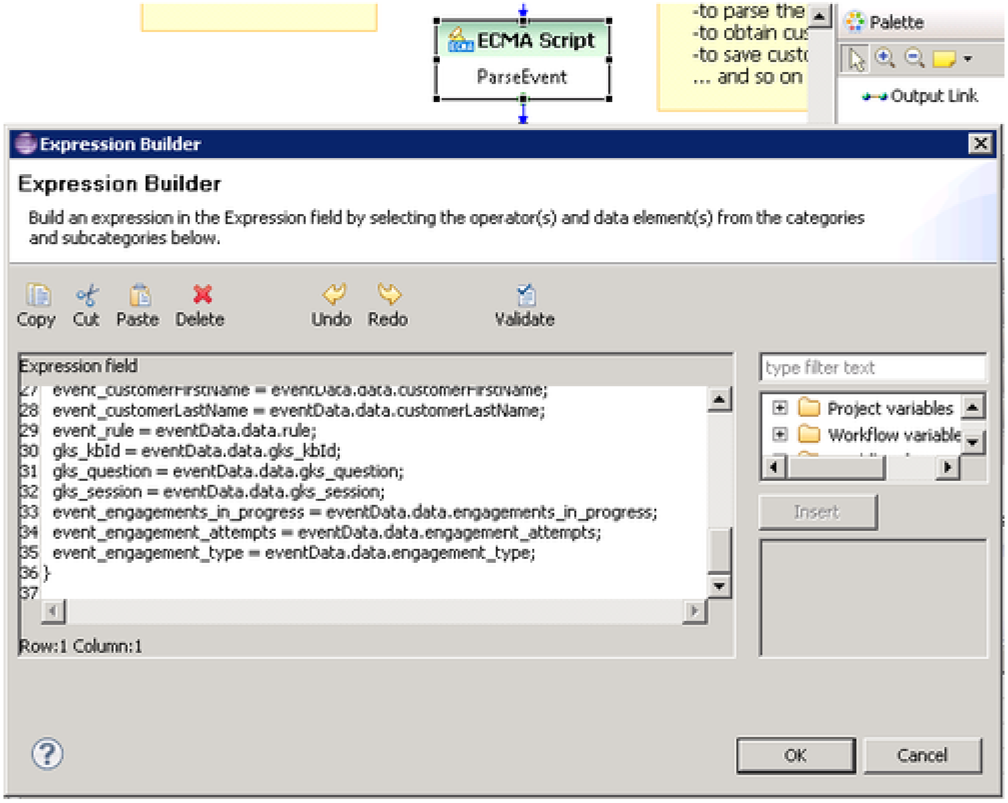
Set the application variables

Set the application variables

Variable Name	Category	Value	Description
system.OriginatingSession	System	undefined	The originating session context.
BackendURL	User	_data.BackendURL	Value of Backend Server which works ...
BackendURLSecure	User	_data.BackendURLSecure	Enter Description
gks_kbId	User	"	Enter Description
gks_question	User	"	Enter Description
gks_session	User	"	Enter Description

Adding New Variables

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



Expression Builder

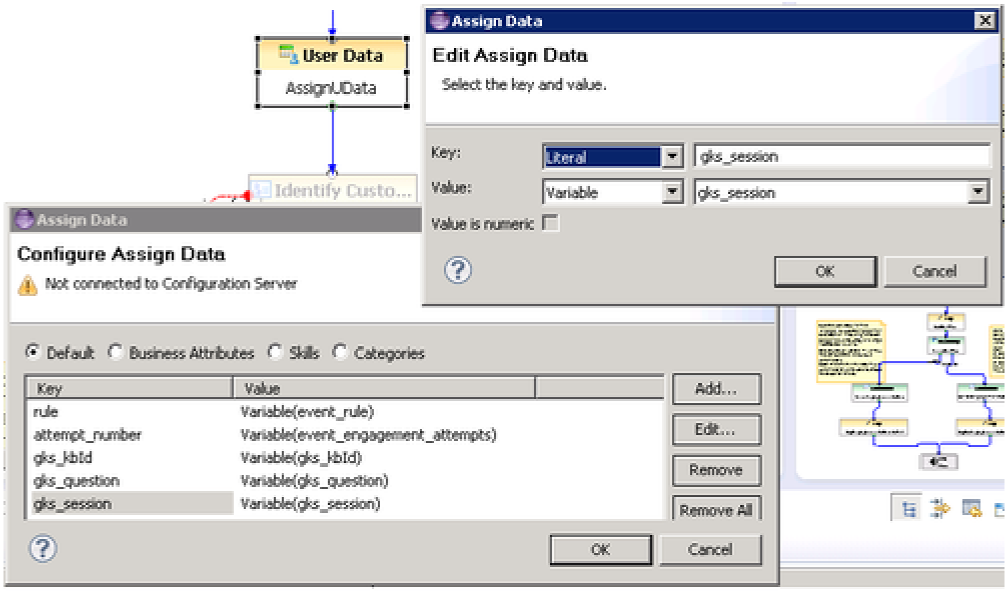
Build an expression in the Expression field by selecting the operator(s) and data element(s) from the categories and subcategories below.

```
  event_customerFirstName = eventData.data.customerFirstName;
  event_customerLastName = eventData.data.customerLastName;
  event_rule = eventData.data.rule;
  gks_kbId = eventData.data.gks_kbId;
  gks_question = eventData.data.gks_question;
  gks_session = eventData.data.gks_session;
  event_engagements_in_progress = eventData.data.engagements_in_progress;
  event_engagement_attempts = eventData.data.engagement_attempts;
  event_engagement_type = eventData.data.engagement_type;
}

```

Row:1 Column:1

ECMA Script for Event Parsing

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



Edit Assign Data

Select the key and value.

Key: Literal | gks_session

Value: Variable | gks_session

Value is numeric ☐

Configure Assign Data

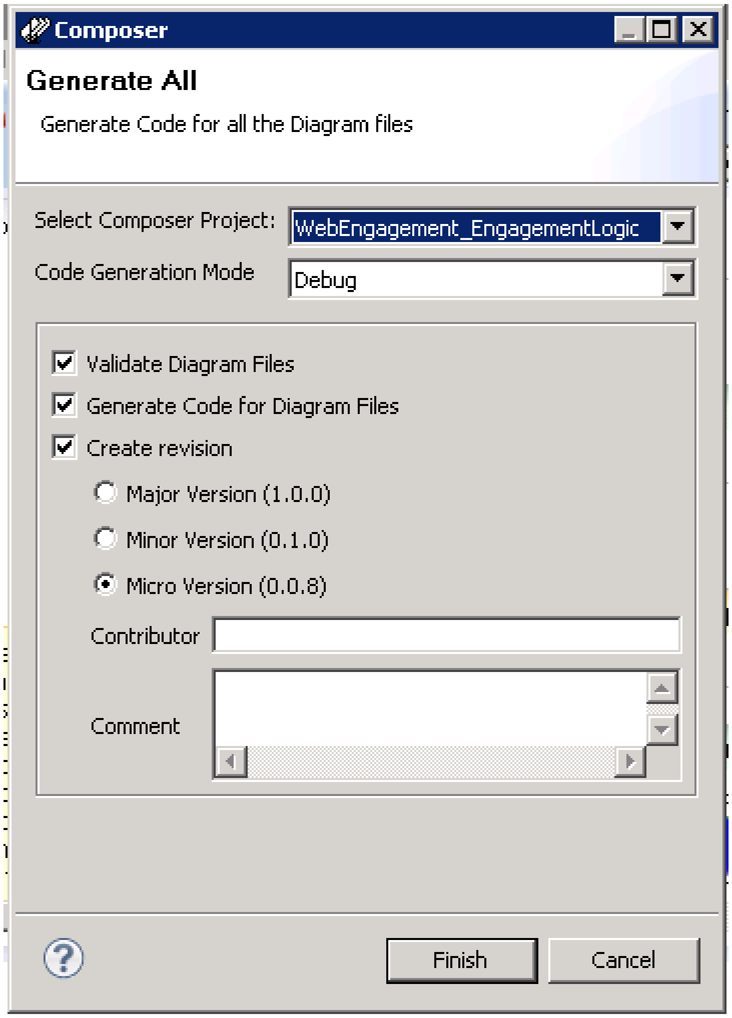
Not connected to Configuration Server

● Default ○ Business Attributes ○ Skills ○ Categories

Key	Value
rule	Variable(event_rule)
attempt_number	Variable(event_engagement_attempts)
gks_kbId	Variable(gks_kbId)
gks_question	Variable(gks_question)
gks_session	Variable(gks_session)

Add Parsed Data to Interaction

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



Generate All

Generate Code for all the Diagram files

Select Composer Project: WebEngagement_EngagementLogic

Code Generation Mode: Debug

☑ Validate Diagram Files

☑ Generate Code for Diagram Files

☑ Create revision

○ Major Version (1.0.0)

○ Minor Version (0.1.0)

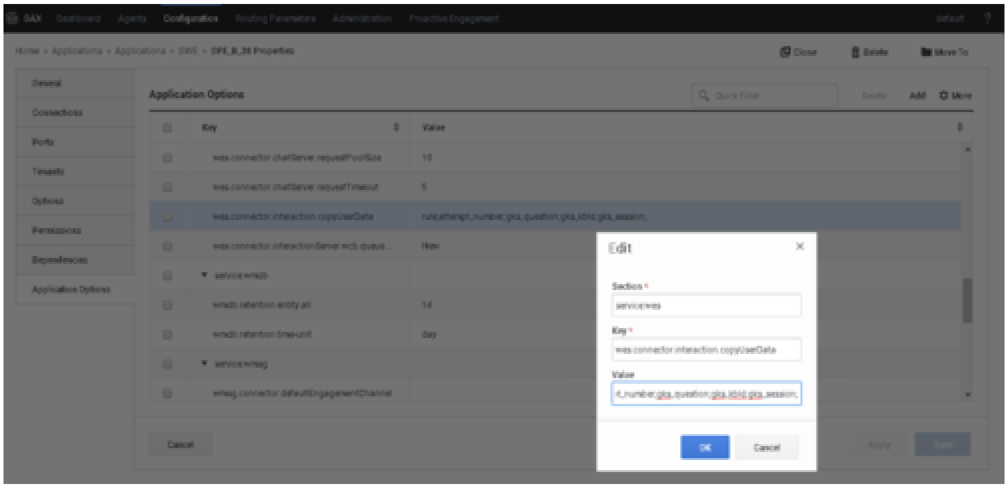
● Micro Version (0.0.8)

Contributor

Comment

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.



Application Options

Key	Value
wes.connector.chatServer.requestPoolSize	10
wes.connector.chatServer.requestTimeout	5
wes.connector.interaction.copyUserData	rule,attempt_number,gks_question,gks_kbId,gks_session,
wes.connector.interactionServer.wcb.queue...	New
▾ service:wmdb	
wmdb.retention.entity.all	14
wmdb.retention.time-unit	day
▾ service:wmag	
wmag.connector.defaultEngagementChannel	

Section *: service:wes

Key *: wes.connector.interaction.copyUserData

Value: t_number,gks_question,gks_kbId,gks_session,

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.

Sample UI

Overview

The Sample UI is based on backbone.js and divided into three parts:

- Knowledge Agent — low level mapper that covers Knowledge API and encapsulate Knowledge session management.

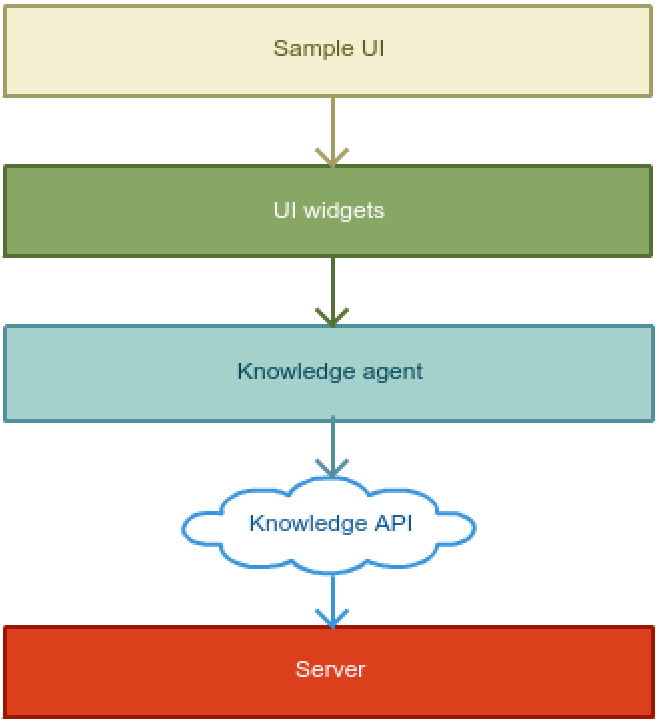
location: gks-sample-ui.war/modules/knowledge_agent/

- UI Widgets — atomic modules that responsible for key UI elements (such as: search panel, search result view, document view).

location: gks-sample-ui.war/modules/widgets/

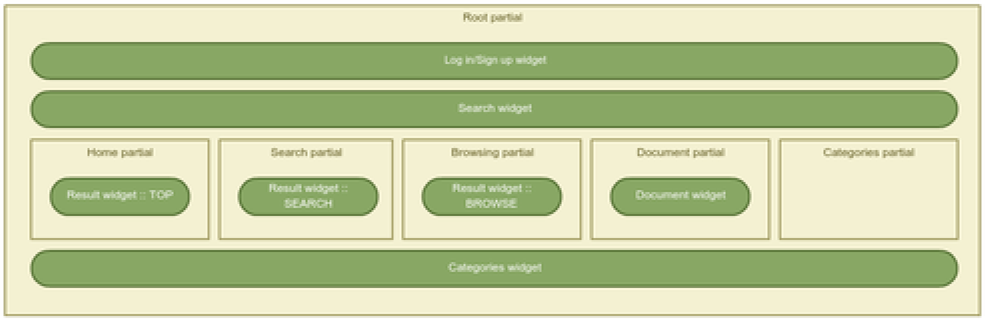
- The Sample-UI itself — combination of the first two and the logic of their interactions.

location: gks-sample-ui.war/



Sample UI

Templates Hierarchy

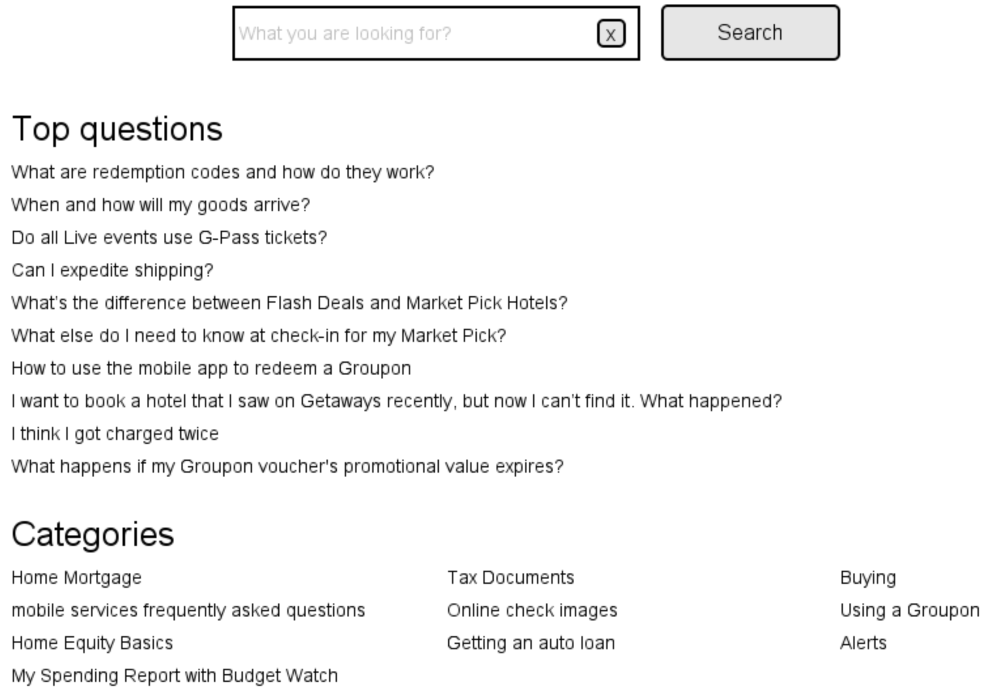


Templates hierarchy and available widgets

Page Descriptions

According to Templates hierarchy there are five general page templates defined:

1. **Home page** — welcome page of Sample UI.
   Displays list of Top questions and associated categories.
   routing: #
   [+] Click here to expand sample



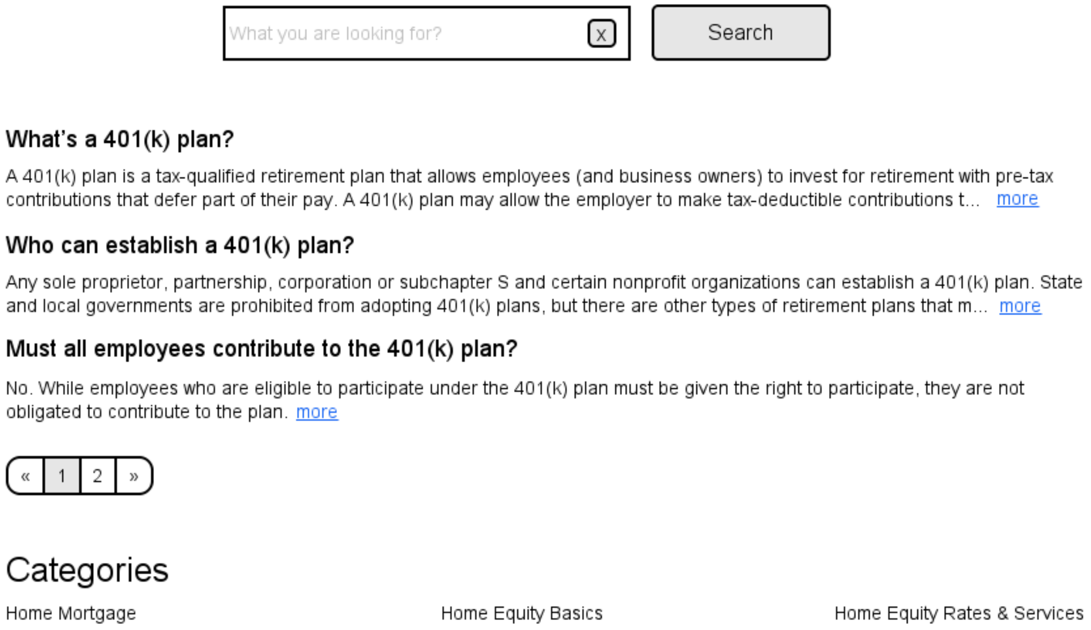
Top questions

What are redemption codes and how do they work?
When and how will my goods arrive?
Do all Live events use G-Pass tickets?
Can I expedite shipping?
What's the difference between Flash Deals and Market Pick Hotels?
What else do I need to know at check-in for my Market Pick?
How to use the mobile app to redeem a Groupon
I want to book a hotel that I saw on Getaways recently, but now I can't find it. What happened?
I think I got charged twice
What happens if my Groupon voucher's promotional value expires?

Categories

Home Mortgage
mobile services frequently asked questions
Home Equity Basics
My Spending Report with Budget Watch
Tax Documents
Online check images
Getting an auto loan
Buying
Using a Groupon
Alerts

2. **Search result page** — result of searching.
   Displays relevant documents based on search query and associated to them categories.
   routing: #category/:categoryId/search/:searchQuery
   [+] Click here to expand sample



What's a 401(k) plan?

A 401(k) plan is a tax-qualified retirement plan that allows employees (and business owners) to invest for retirement with pre-tax contributions that defer part of their pay. A 401(k) plan may allow the employer to make tax-deductible contributions t... more

Who can establish a 401(k) plan?

Any sole proprietor, partnership, corporation or subchapter S and certain nonprofit organizations can establish a 401(k) plan. State and local governments are prohibited from adopting 401(k) plans, but there are other types of retirement plans that m... more

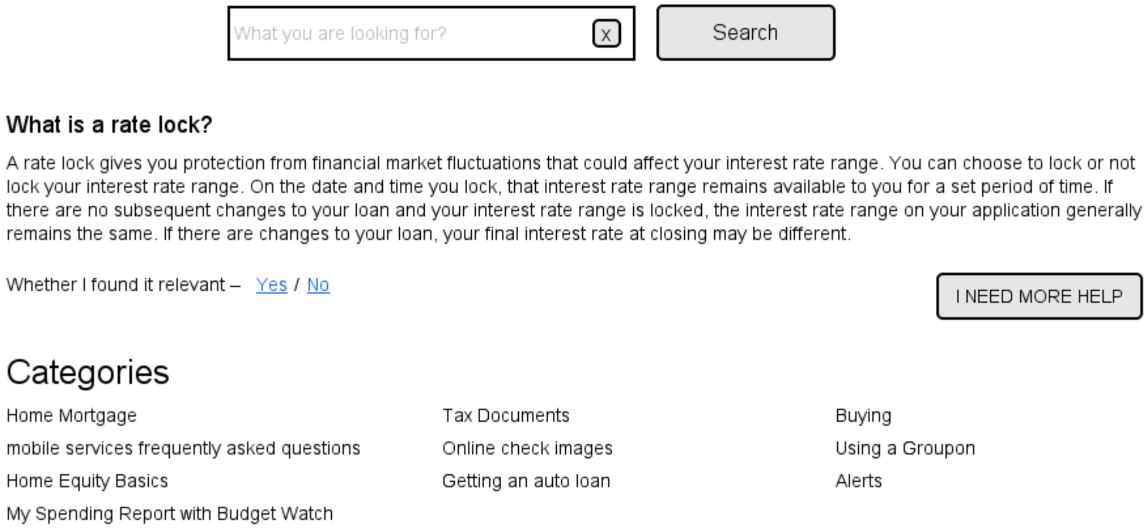
Must all employees contribute to the 401(k) plan?

No. While employees who are eligible to participate under the 401(k) plan must be given the right to participate, they are not obligated to contribute to the plan. more

Categories

Home Mortgage Home Equity Basics Home Equity Rates & Services

3. **Browsing page** — categories browsing.
   Displays documents in specific category and associated to them categories.
   routing: #category/:categoryId/search/
4. **Document page** — full document info.
   Displays full content of the specific document. In case of click directly after search, this page also contains voting area and help button.
   routing: #kbId/:kbId/document/:documentId
   [+] Click here to expand sample



What is a rate lock?

A rate lock gives you protection from financial market fluctuations that could affect your interest rate range. You can choose to lock or not lock your interest rate range. On the date and time you lock, that interest rate range remains available to you for a set period of time. If there are no subsequent changes to your loan and your interest rate range is locked, the interest rate range on your application generally remains the same. If there are changes to your loan, your final interest rate at closing may be different.

Whether I found it relevant – Yes / No

Categories

Home Mortgage
mobile services frequently asked questions
Home Equity Basics
My Spending Report with Budget Watch
Tax Documents
Online check images
Getting an auto loan
Buying
Using a Groupon
Alerts

5. **Categories page** — list of available categories.
   Displays all available categories and provides browsing capability for them.
   routing: #categories

Knowledge Agent

Overview

Knowledge agent is an AMD-based module that can be used with RequireJS. It exports the _gka variable into the local context where it is accessed.

Configuration

_gka.initialize(options)	Examples

Description: Configure the Knowledge agent.
• **options**
Type: PlainObject
A set of key/value pairs that configure the Agent.
• **host**
Type: String
A network host where Knowledge API is stored.
• **kbId**
Type: String
Knowledgebase default identifier to be used.
• **knowledgebases**
Type: String[]
Knowledgebases identifiers to be used
• **lang**
Type: String
Default language to be used.
• **agentId**
Type: String
Agent ID
• **auth**
Type: String
Agent ID
• **customerId**
Type: String
	`_gka.initialize({`
	`    host: 'http://localhost:8080',`
	`    knowledgebases: ['knowledgefaq', 'knowledgearticles']`
	`})`

_gka.initialize(options)	Examples
Customer ID - **authorization** Type: String Basic authorization token - **media** (default: 'chat') Type: String Default search filter by media channel type - **apiClientId** (default: 'web') Type: String A web client identifier - **apiClientMediaType** (default: 'selfservice') Type: String A web client media type	

Knowledge Agent API

Once configuration is complete, the Agent can receive data from the Knowledge API. The _gka variable contains the following interfaces:

Method	Description
Knowledge Base Operations	
.getKnowledgeBases()	Retrieves list of knowledge bases supported
.getKnowledgeBaseInfo()	Retrieves information about the particular knowledge base
.getCategories()	Retrieves list of categories
.getFullContent()	Retrieves full content of particular document
FAQ retrival	
.search()	Executes search for the answer for the given query
.getDocumentsByCategory()	Retrieves documents associated to a specific category
.getTrending()	Retrieves trending documents
.suggestions()	Provides autocomplete functionality
Feedback	
.noAnswer()	Marks query as the one that do not have valid answer in knowledge base
.vote()	Records user rating for the document within query
.advancedVote()	Advanced version of .vote()
.visit()	Registers viewing the document
.rating()	Registers 5-star rating for the document
.addRating()	Adds rating feedback to an existing vote

Knowledge Base Operations

> **Important**
>
> For additional information, please refer to the Knowledge API page.

_gka.getKnowledgeBases(): Promise

Example

Description: Retrieves information about the particular knowledge base.

```
 _gkn.getKnowledgeBases().done(function(response, status) {
    console.log(response)
}).fail(function(error, status) {
    console.log(error)
});
```

Response

- **knowledgebases**

  Type: PlainObject[]
  List of supported knowledgebases

  - **name**
    Type: String
    Name of knowledgebase
  - **languages**
    Type: String[]
    List of supported languages

```
{
    "knowledgebases": [{
        "name": "knowledgefaq",
        "languages": [
            "en"
        ]
    }, {
        "name": "the_bank",
        "languages": [
            "en"
        ]
    }]
}
```

_gka.getKnowledgeBaseInfo([options]): Promise

Example

Description: Retrieves information about the knowledge base.

- **options**
  Type: PlainObject
  A set of key/value pairs that contains arguments for the RESTful API.
  - **kbId** (default: stored _gka.kbId)
    Type: String
    Particular knowledge base identifier.

```
_gkn.getKnowledgeBaseInfo().done(function(response, status) {
    console.log(response)
}).fail(function(error, status) {
    console.log(error)
});
```

Response

_gka.getKnowledgeBaseInfo([options]): Promise	**Example**

- **languages**
Type: String[]
List of supported languages for given knowledgebase
	`{`
	`    languages: [`
	`        'en',`
	`        'fr',`
	`        'de'`
	`    ]`
	`}`

_gka.getCategories(): Promise	**Example**

Description: Retrieves list of categories.	`_gkn.getCategories().done(function(response, status) {`
	`    console.log(response)`
	`}).fail(function(error, status) {`
	`    console.log(error)`
	`});`

Response	

- **categories**
Type: PlainObject[]
List of supported categories
- **id**
Type: String
Category identifier
- **name**
Type: String
Category name
- **count**
Type: Number
Total count of documents in category
	`{`
	`    "categories": [{`
	`        "id": "Home Mortgage",`
	`        "name": "Home Mortgage",`
	`        "count": 78`
	`    }, {`
	`        "id": "Home Equity Basics",`
	`        "name": "Home Equity Basics",`
	`        "count": 78`
	`    }]`
	`}`

_gka.getFullContent(options): Promise

Description: Retrieves full content of particular document.

- **options**
  Type: PlainObject
  A set of key/value pairs that contains arguments for the RESTful API.
  - **kbId**
    Type: String
    Knowledge base identifier
    - **docId**
      Type: String
      Particular document identifier.

Example

```
 _gkn.getFullContent({
    docId: 'knowledge'
}).done(function(response, status) {
    console.log(response)
}).fail(function(error, status) {
    console.log(error)
});
```

Response

- **id**

  Type: String
- **language**
  Type: String
- **typeName**
  Type: String
- **kbId**
  Type: String
- **categories**
  Type: String[]
- **created**
  Type: Number
- **modified**
  Type: Number
- **snippet**

```
{
    "id":"knowledge",
    "language":"en",
    "typeName":"qna_document_en",
    "kbId":"knowledge",
    "categories":[
        "Booking trips"
    ],
    "created":1402573911163,
    "modified":1402573911163,
    "snippet":"",
    "fields":{
        "id":"knowledge",
        "created":1402573911163,
        "answer":"Every travel option we offer",
        "categories":[
            "Booking trips"
        ],
        "question":"What’s the difference between",
        "modified":1402573911163
    }
}
```

_gka.getFullContent(options): Promise	Example
Type: String • **fields** • **id** Type: String • **created** Type: Number • **answer** Type: String • **categories** Type: String[] • **question** Type: String • **modified** Type: Number	

FAQ retrival

<table>
<tr><th>_gka.search([options]): Promise</th><th>Example</th></tr>
<tr><td>

Description: Executes search for the answer for the given query.

- **options**
  Type: PlainObject
  A set of key/value pairs that contains arguments for the RESTful API.
  - **from** (default: 0)
    Type: Number
    Pagination offset.
  - **size** (default: 10)
    Type: Number
    Pagination page size
  - **query** (default: '')
    Type: String
    User typed query string
  - **categories** (default: [])
    Type: String[]
    List of categories that is used as a context for the current query
  - **filters**
    Type: String[]
    List of filters

</td><td>

```
_gkn.search({
    from: 0,
    size: 10,
    query: '',
    categories: []
}).done(function(response, status) {
    console.log(response)
}).fail(function(error, status) {
    console.log(error)
});
```

</td></tr>
<tr><td colspan="2">

Response

</td></tr>
<tr><td>

- **count**

  Type: Number
- **page**
  Type: PlainObject
  - **from**
    Type: Number

</td><td>

```
{
    "count": 78,
    "page": {
        "from": 0,
        "size": 10
    },
    "documents": [
        {
            "id": "knowledge",
```

</td></tr>
</table>

<table>
<tr><th>_gka.search([options]): Promise</th><th>Example</th></tr>
<tr><td>
<ul>
<li><ul><li><b>size</b>
Type: Number</li></ul></li>
<li><b>documents</b>
Type: PlainObject[]
<ul>
<li><b>id</b>
Type: String</li>
<li><b>language</b>
Type: String</li>
<li><b>typeName</b>
Type: String</li>
<li><b>kbId</b>
Type: String</li>
<li><b>categories</b>
Type: String[]</li>
<li><b>created</b>
Type: Number</li>
<li><b>modified</b>
Type: Number</li>
<li><b>snippet</b>
Type: String</li>
<li><b>score</b>
Type: Number</li>
<li><b>fields</b>
Type: PlainObject
<ul>
<li><b>question</b>
Type: String</li>
</ul></li>
</ul></li>
</ul>
</td><td>
<pre>
            "language": "en",
            "typeName": "qna_document_en",
            "kbId": "knowledge",
            "categories": [
                "Home Equity Basics",
                "Home Mortgage"
            ],
            "created": 1404823845989,
            "modified": 1404823845989,
            "snippet": "What is the difference\n",
            "score": 4.20981,
            "fields": {
                "question": "What is the difference",
                "answer": "Locking ensures that you"
            },
            "morelikethis": [

            ],
            "confidence": 1.0
        },
        {
            "id": "knowledge",
            "language": "en",
            "typeName": "qna_document_en",
            "kbId": "knowledge",
            "categories": [
                "Home Equity Basics",
                "Home Mortgage"
            ],
            "created": 1404823845989,
            "modified": 1404823845989,
            "snippet": "How long do I have to pay",
            "score": 4.20981,
            "fields": {
                "question": "How long do I have",
                "answer": "If you obtained your"
            },
            "morelikethis": [
</pre>
</td></tr>
</table>

_gka.search([options]): Promise

        - **answer**
          Type: String
    - **morelikethis**
      Type: String[]
    - **confidence**
      Type: Number
- **categories**
  Type: PlainObject[]
    - **id**
      Type: String
    - **name**
      Type: String
    - **count**
      Type: String

Example

```
            ],
            "confidence": 1.0
        }
    ],
    "categories": [
        {
            "id": "Home Equity Basics",
            "name": "Home Equity Basics",
            "count": 78
        },
        {
            "id": "Home Equity Rates & Services",
            "name": "Home Equity Rates & Services",
            "count": 2
        }
    ]
}
```

_gka.getDocumentsByCategory(): Promise

Description: Retrieves documents associated to a specific category.

- **options**
  Type: PlainObject
  A set of key/value pairs that contains arguments for the RESTful API.
    - **kbId**
      Type: String
      Knowledge base identifier
        - **catId**
          Type: Number
          Category ID.

Example

```
_gka.getDocumentsByCategory({
 catId: options.categories[0]
}).done(function(reponse) {
 console.log(response)
}).fail(function (error, status) {
 console.warn(error)
});
```

_gka.getDocumentsByCategory(): Promise	**Example**
- **from** (default:0) Type: Number Pagination offset. - **size** (default: 10) Type: Number Pagination page size	
Response	

- **count**
Type: Number[]
- **page**
Type: PlainObject
  - **from**
  Type: Number
  - **size**
  Type: Number
- **documents**
Type: PlainObject[]
  - **id**
  Type: String
  - **language**
  Type: String
  - **typeName**
  Type: String
  - **kbId**
  Type: String
	`{`
	`    "count": 78,`
	`    "page": {`
	`        "from": 0,`
	`        "size": 10`
	`    },`
	`    "documents": [`
	`        {`
	`            "id": "GBank_458",`
	`            "language": "en",`
	`            "typeName": "qna_document_en",`
	`            "kbId": "GBank",`
	`            "categories": [`
	`                "Home Equity Basics",`
	`                "Home Mortgage"`
	`            ],`
	`            "created": 1404823845989,`
	`            "modified": 1404823845989,`
	`            "snippet": "What is the difference\n",`
	`            "score": 4.20981,`
	`            "fields": {`
	`                "question": "What is the difference",`
	`                "answer": "Locking ensures that you"`
	`            },`
	`            "morelikethis": [`
	
	`            ],`

_gka.getDocumentsByCategory(): Promise	Example

  - **categories**
    Type: String[]
  - **created**
    Type: Number
  - **modified**
    Type: Number
  - **snippet**
    Type: String
  - **score**
    Type: Number
  - **fields**
    Type: PlainObject
    - **question**
      Type: String
    - **answer**
      Type: String
  - **morelikethis**
    Type: String[]
  - **confidence**
    Type: Number
- **categories**
  Type: PlainObject[]
  - **id**
    Type: String
  - **name**
    Type: String

```
                "confidence": 1.0
            },
            {
                "id": "GBank_477",
                "language": "en",
                "typeName": "qna_document_en",
                "kbId": "GBank",
                "categories": [
                    "Home Equity Basics",
                    "Home Mortgage"
                ],
                "created": 1404823845989,
                "modified": 1404823845989,
                "snippet": "How long do I have to pay",
                "score": 4.20981,
                "fields": {
                    "question": "How long do I have",
                    "answer": "If you obtained your"
                },
                "morelikethis": [

                ],
                "confidence": 1.0
            }
        ],
        "categories": [
            {
                "id": "Home Equity Basics",
                "name": "Home Equity Basics",
                "count": 78
            },
            {
                "id": "Home Equity Rates & Services",
                "name": "Home Equity Rates & Services",
                "count": 2
            }
        ]
}
```

_gka.getDocumentsByCategory(): Promise	**Example**
• **count** Type: String For additional information, please refer to the Knowledge API page.	

_gka.getTrending([options]): Promise	**Example**

Description: Retrieves trending documents.
• **options**
Type: PlainObject
A set of key/value pairs that contains arguments for the RESTful API.
• **size** (default: 10)
Type: Number
Pagination page size
	_gka.getTrending().done(function(reponse) {
	 console.log(response)
	}).fail(function (error, status) {
	 console.warn(error)
	});

Response	

• **count**
Type: Number[]
• **documents**
Type: PlainObject
• **id**
Type: String
• **language**
Type: String
• **typeName**
Type: String
• **kbId**
Type: String
	{
	    "count": 78,
	    "documents": [
	        {
	            "id": "GBank_458",
	            "language": "en",
	            "typeName": "qna_document_en",
	            "kbId": "GBank",
	            "categories": [
	                "Home Equity Basics",
	                "Home Mortgage"
	            ],
	            "created": 1404823845989,
	            "modified": 1404823845989,
	            "snippet": "What is the difference\n",
	            "score": 4.20981,
	            "fields": {

_gka.getTrending([options]): Promise	Example

• **categories**
Type: String[]
• **created**
Type: Number
• **modified**
Type: Number
• **snippet**
Type: String
• **score**
Type: Number
• **fields**
Type: PlainObject
  • **question**
  Type: String
  • **answer**
  Type: String
• **morelikethis**
Type: String[]
• **confidence**
Type: Number
• **categories**
Type: PlainObject[]
• **id**
Type: String
• **name**
Type: String
	"question": "What is the difference",
	"answer": "Locking ensures that you"
	},
	"morelikethis": [
	
	],
	"confidence": 1.0
	},
	{
	"id": "GBank_477",
	"language": "en",
	"typeName": "qna_document_en",
	"kbId": "GBank",
	"categories": [
	"Home Equity Basics",
	"Home Mortgage"
	],
	"created": 1404823845989,
	"modified": 1404823845989,
	"snippet": "How long do I have to pay",
	"score": 4.20981,
	"fields": {
	"question": "How long do I have",
	"answer": "If you obtained your"
	},
	"morelikethis": [
	
	],
	"confidence": 1.0
	}
	],
	"categories": [
	{
	"id": "Home Equity Basics",
	"name": "Home Equity Basics",
	"count": 78
	},
	{
	"id": "Home Equity Rates & Services",

_gka.getTrending([options]): Promise	Example

• **count**
Type: String
For additional information, please refer to the Knowledge API page.
	`            "name": "Home Equity Rates & Services",`
	`            "count": 2`
	`        }`
	`    ]`
	`}`

_gka.suggestions(options): Promise	Example

Description: Provides autocomplete functionality.
• **options**
Type: PlainObject
A set of key/value pairs that contains arguments for the RESTful API.
• **query**
Type: String
User typed query string.
• **categories**
Type: String
List of categories that are used as context for the query.
	`_gkn.suggestions({`
	`    query: 'ipad'`
	`}).done(function(response, status) {`
	`    console.log(response)`
	`}).fail(function(error, status) {`
	`    console.log(error)`
	`});`

Response	

• **suggestions**
Type: String[]
For additional information, please refer to the Knowledge API page.
	`{`
	`    "suggestions":[`
	`        "What else do I need to know at check-in",`
	`        "What is a Non-Sufficient Funds fee",`
	`        "What’s a 401(k) plan?\n",`
	`    ]`
	`}`

Feedback

_gka.noAnswer(options): Promise	**Example**

Description: Marks query as the one that do not have valid answer in knowledge base.
- **options**
Type: PlainObject
A set of key/value pairs that contains arguments for the RESTful API.
- **from**
Type: Number
Pagination offset.
- **size**
Type: Number
Pagination page size
- **query**
Type: String
User typed query string
- **categories**
Type: String[]
List of categories that are used as context for the current query
For additional information, please refer to the Knowledge API page.
	_gkn.noAnswer ({
	    from: 0,
	    size: 10,
	    query: '',
	    categories: []
	}).fail(function(error, status) {
	    console.log(error)
	})

_gka.vote(options): Promise	**Example**

Description: Records user ratings for the document within a query.
- **options**
Type: PlainObject
A set of key/value pairs that contains arguments for the RESTful API.
- **kbId**
Type: String
Knowledge base identifier
- **docId**
	_gkn.like({
	    docId: ''groupon_22'',
	    relevant: false
	}).fail(function(error, status) {
	    console.log(error)
	});

_gka.vote(options): Promise	Example
Type: String Particular document identifier. • **relevant** (default: true) Type: Boolean Whether the search result was relevant. • **query** Type: String User typed query string. • **categories** Type: String List of categories that are used as context for the query. • **filters** Type: String[] List of filters.	
Response	
• **recordId** Type: String[] Created vote ID	

_gka.advancedVote(options): Promise	Example

Description: Marks queries that do not have a valid answer in knowledge base.
• **options**
Type: PlainObject
A set of key/value pairs that contains arguments for the RESTful API.
	`_gka.advancedVote({`
	`    likeDocId: 'id2', selection: ['id1', 'id2', id3']`
	`});`

_gka.advancedVote(options): Promise	**Example**
• **kbId** Type: String Knowledge base identifier • **likeDocId** Type: String Particular document identifier. • **selection** Type: String[] An array of document ID's in search result. • **request** (default: '') Type: PlainObject Request for the associated search. • **query** Type: String User typed query string. • **categories** Type: String[] List of categories that are used as context for the current query • **filters** Type: String[] List of filters. For additional information, please refer to the Knowledge API page.	
Response	
• **recordId** Type: String Created vote ID	

_gka.visit(options): Promise	**Example**

Description: Registers document views.
- **options**
Type: PlainObject
A set of key/value pairs that contains arguments for the RESTful API.
- **kbId**
Type: String
Knowledge base identifier
- **docId**
Type: String
Particular document identifier.
- **query**
Type: String
User typed query string.
- **categoreis**
Type: String[]
List of categories that are used as context for the current query.
- **filter**
Type: String[]
List of filters.
	`_gkn.visit({`
	`    docId: "knowledge",`
	`}).fail(function(error, status) {`
	`    console.log(error)`
	`});`

_gka.rating(options):Promise	**Example**

Description: Registers 5-star rating for the document
- **options**
Type: PlainObject
A set of key/value pairs that contains arguments for the RESTful API.
- **kbId**
Type: String
Knowledge base identifier
	`_gka.rating({`
	` kbId: 'knowledgefaq',`
	` docId: '550e8400-e29b-41d4-a716-446655440000',`
	` comment: 'This document was very helpful',`
	` rating: 5`
	`});`

_gka.rating(options):Promise	Example
• **docId** Type: String Particular document identifier. • **comment** Type: String Text comment • **rating** Type: String enum (1, 2, 3, 4, 5) Rating for the document	
Response	
• **recordId** Type: String Created vote ID	

_gka.addRating(options): Promise	Example

Description: Add rating & comment to an existing vote.
• **options**
Type: PlainObject
A set of key/value pairs that contains arguments for the RESTful API.
• **voteId**
Type: String
Particular vote identifier
• **comment**
Type: String
Text comment
	`_gka.addRating({`
	`voteId: 'vote_id',`
	`comment: 'some comment',`
	`rating: 5`
	`});`

_gka.addRating(options): Promise	**Example**
- **rating** Type: String enum (1, 2, 3, 4, 5) Rating for the document	
Response	
Not Provided	

_gka.addComment(options): Promise	**Example**

Description: Registers document views.
- **options**
Type: PlainObject
A set of key/value pairs that contains arguments for the RESTful API.
- **voteId**
Type: String
Particular vote identifier.
- **comment**
Type: String[]
Comment body
	`_gka.addComment({`
	`    voteId: 'vote_id', comment: 'some comment'`
	`});`

Response	
- **recordId** Type: String Created vote ID	

Promise Object

The object returned by all methods of _gka variable implements the Promise interface, giving them all the properties, methods, and behaviour of a Promise (see jQuery Deferred for more information). It represents a value that may not be available yet.

promise.done(doneCallbacks [, doneCallbacks]): Promise	Example

Description: Add handlers to be called when the Deferred object is resolved.
• **doneCallbacks**
Type: Function(response, status)
A function, or array of functions, that are called when the Deferred is resolved.
• **doneCallbacks**
Type: Function(response, status)
Optional additional functions, or arrays of functions, that are called when the Deferred is resolved.
	`promise.done(function(response, status) {`
	`    alert('ajax call succeeded');`
	`    console.log(response);`
	`    console.log(status)`
	`});`

promise.fail(failCallbacks [, failCallbacks])): Promise	Example

Description: Add handlers to be called when the Deferred object is rejected.
• **doneCallbacks**
Type: Function(error, status)
A function, or array of functions, that are called when the Deferred is rejected.
• **doneCallbacks**
Type: Function(error, status)
Optional additional functions, or arrays of functions, that are called when the Deferred is rejected.
	`promise.done(function() {`
	`    alert('ajax call succeeded');`
	`}).fail(function(error, status) {`
	`    alert('ajax call failed');`
	`    console.log(error);`
	`    console.log(status)`
	`});`

promise.always(alwaysCallbacks [, alwaysCallbacks]): Promise	Example

Description: Add handlers to be called when the Deferred object is either resolved or rejected.
• **doneCallbacks**
Type: Function(response|error, status)
A function, or array of functions, that is called when the Deferred is
	`promise.always(function(responseOrError, status) {`
	`    alert('ajax call completed with success or error callback arguments');`
	`    console.log(responseOrError);`
	`    console.log(status)`

promise.always(alwaysCallbacks [, alwaysCallbacks]): Promise	Example
resolved or rejected. • **doneCallbacks** Type: Function(response|error, status) Optional additional functions, or arrays of functions, that are called when the Deferred is resolved or rejected.	`});`

promise.state(): String
Description: Determine the current state of a linked Deferred object. The .state() method returns a string representing the current state of the Deferred object. The Deferred object can be in one of three states: • **pending**: The Deferred object is not yet in a completed state (neither "rejected" nor "resolved"). • **resolved**: The Deferred object is in the resolved state, meaning that the doneCallbacks have been called (or are in the process of being called). • **rejected**: The Deferred object is in the rejected state, meaning that the failCallbacks have been called (or are in the process of being called).

UI Widgets

Overview

The Sample UI is based on independent and easily configurable components. Each component is a **View** in term of Backbone.js and may depend on Knowledge Agent and other 3rd party libraries. All the dependencies are managed by RequireJS in AMD-style.

When using widgets, the path to their file must be written using the **.define** function, which exports a constructor function to the current context. The last step is to create object **(new Constructor(options))** based on this constructor and call **.render()** method. Some widgets (in particular : Categories, Result and Document widgets) fetch data from the server and, in this case, **.fire()** method must be called before **.render()** method. Furthermore, each widget has public object settings, with stored widget configuration, based on constructor arguments.

You can find a Basic API on the Backbone documentation page.

Components

Search Widget

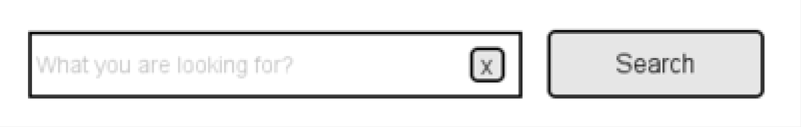
The Search widget displays the standard Search bar and has a custom placeholder and optional Clear button.

SearchWidget([options])

Description: constructor for search widget.

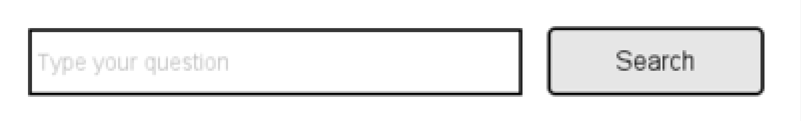
- **el** (default: '#_gk-_wd-_sr')
  Type: String
  Selector for DOM element in which the current widget will be inserted
- **placeholder** (default: *)*
  Type: String
  Placeholder for search input
- **buttonText** (default: 'Search')
  Type: String
  The text in search button
- **showClearButton** (default: true)
  Type: Boolean
  Whether to show or not Clear button
- **query** (default: *)*
  Type: String
  Typed text for search input
- **categories** (defult: [])
  Type: String[]
  Context categories for autocomplete
- **searchButtonClickEventListeners**
  Type: Function(Element element, PlainObject options)[]
  Functions to be called on Search button click
  - **element**
    Type: Element
    An element in the Document Object Model (DOM)
  - **options**
    Type: PlainObject
    A set of key/value pairs that contains data for listeners.
    - **query**

Example



Search Widget Interface

```
new SearchWidget({
    placeholder: 'What are you looking for?',
    showClearButton: true,
    searchButtonClickEventListeners: [function (element, options)
{
        console.log('Search button clicked')
    }]
}).render();
```



Search Widget Interface

```
new SearchWidget({
    placeholder: 'Type your question',
    showClearButton: false
}).render();
```

SearchWidget([options])	Example
Type: String User typed query string. • **clearButtonClickEventListeners** Type: Function(Element element)[] Functions to be called on Clear button click • **element** Type: Element An element in the Document Object Model (DOM)	
.render(): SearchWidget	
Description: renders the view template and updates **this.el** with the new HTML.	

Result Widget

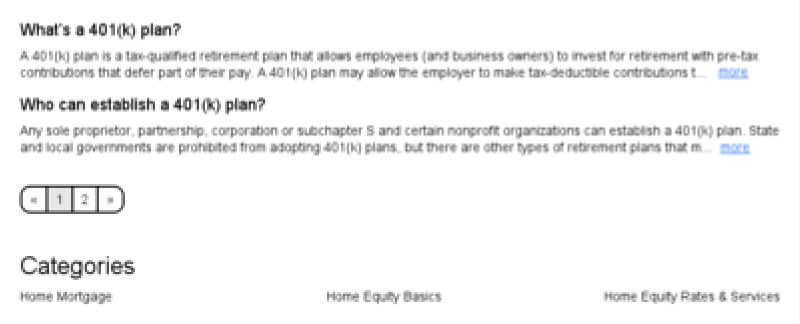
The Result widget displays the search results and has optional pagination and optional embedded blocks of categories. this widget is dependent on the **_gka.search(**) method in Knowledge Agent.

ResultWidget([options])

Description: constructor for results widget.

- **el** (default: '#_gk-_wd-_rs')
  Type: String
  Selector for DOM element in which the current widget will be inserted
- **resultType** (default: 'SEARCH')
  Type: String enum ('SEARCH', 'TOP', 'BROWSE')
  Base type of the widget
- **size** (default: 10)
  Type: Number
  Number of documents in result list
- **showAnswer** (default: true)
  Type: Boolean
  Whether to show or not answers in document
- **charactersInAnswer** (default: 250)
  Type: Number
  Number of character in answer before "more" link appears
- **pagination** (default: true)
  Type: Boolean
  Whether to show or not pagination panel. Works only with resultType= 'BROWSE'
- **filters**
  Type: PlainObject[]
  Array of custom filters for search.
  - **fieldId**
    Type: String
    Identifier of the field which need to be filtered (segment equal "premium")
  - **operation**
    Type: String enum ("equal", "gt", "lt", "range", "regexp", "enum")

Example



What's a 401(k) plan?

A 401(k) plan is a tax-qualified retirement plan that allows employees (and business owners) to invest for retirement with pre-tax contributions that defer part of their pay. A 401(k) plan may allow the employer to make tax-deductible contributions t... more

Who can establish a 401(k) plan?

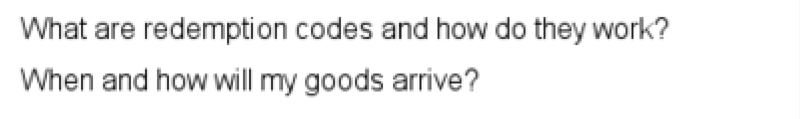
Any sole proprietor, partnership, corporation or subchapter S and certain nonprofit organizations can establish a 401(k) plan. State and local governments are prohibited from adopting 401(k) plans, but there are other types of retirement plans that m... more

Categories

Home Mortgage Home Equity Basics Home Equity Rates & Services

Result Widget Interface

```
new ResultWidget({
    size: 2,
    showAnswer: true,
    charactersInAnswer: 250,
    pagination: true,
    highlighting: false,
    moreLinkClickEventListeners: [function (element, options) {
              console.log('Document selected')
     }]
}).fire().done(function(response, widget) {
 widget.render()
});
```



What are redemption codes and how do they work?

When and how will my goods arrive?

Result Widget Interface

ResultWidget([options])	Example

Expression operator (segment equal "premium")
- **value**
Type: not defined
Value for expression (segment equal "premium")
- **moreLinkText** (default: 'more')
Type: String
Allows to override default text in "more" link
- **showNoAnswerButton**
Type: Boolean
Whether the "No relevant results" button will be shown
- **noAnswerButtonText** (default:"No relevant results")
Type: String
Localizable button label
- **noAnswerClickedText'** (default: "Thank you for your feedback!")
Type: String
Localizable message after the "No relevant results" button has been clicked
- **noMatchesText** (default: "No matches found.")
Type: String
Localizable message if search result contains zero documents
- **documentClickURI** (default: function () { return 'javascript:;' })
Type: Function()
URI after "more" button has been clicked
- **documentClickListeners**
Type: Function(Element element, PlainObject options)[]
Functions to be called on "more" link click
- **element**
Type: Element
An element in the Document Object Model (DOM)
	`new ResultWidget({`
	`    size: 2,`
	`    showAnswer: false,`
	`    showCategories: false`
	`}).fire().done(function(response, widget) {`
	` widget.render()`
	`});`

ResultWidget([options])	Example
• **options** Type: PlainObject A set of key/value pairs that contains data for listeners. • **id** Type: Number Document identifier **question** Type: String Question in document **answer** Type: String Answer in document • **noAnswerClickListeners** Type: Function(Element element)[] Functions to be called on "No relevant results" button click	
.fire(): Promise	
Description: fetch data from the server according to passed options. • **query** (default:) Type: String User typed query string • **categories** (default: []) Type: String[] List of categories that is used as a context for the current query • **filters** (default: []) Type: String[] Filters that will be passed. Works anly with resultType='SEARCH'	
.render(): ResultWidget	
Description: renders the view template from fetched data and updates this.el with the new HTML	

Categories Widget

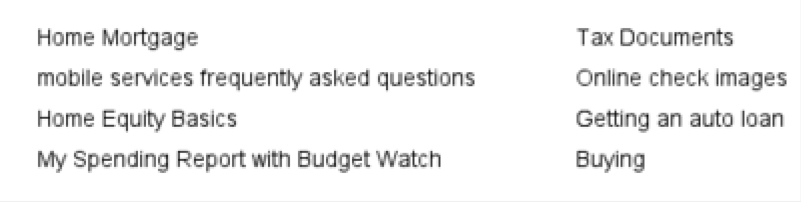
The Categories widget displays categories and is similar to the categories block in the Result widget or the Document widget however, in the Categories widget, only the categories are stored. The Categories widget is dependant on the **_gka.getCategories()** method in Knowledge Agent.

CategoriesWidget([options])

Description: constructor for categories widget.

- **el** (default: '#_gk-_wd-_cat')
  Type: String
  Selector for DOM element in which the current widget will be inserted
- **numberOfColumns** (default: 3)
  Type: Number
  Number of columns in panel categories
- **filteredCategories** (default: [])
  Type: String[]
  Array of id's for categories which should be rendered. Empty array means that all of fetched categories will be rendered.
- **categoryClickURI** (default: function () {return 'javascript:;'})
  Type: Function()
  URI after category has been selected (clicked)
- **categoryClickEventListeners**
  Type: Function(Element element)[]
  Functions to be called after particular category selected
  - **element**
    Type: Element
    An element in the Document Object Model (DOM)
  - **options**
    Type: PlainObject
    A set of key/value pairs that contains data for listeners.
    - **id**
      Type: Number
      Category identifier
    - **name**
      Type: String
      Category name

Example



Home Mortgage
mobile services frequently asked questions
Home Equity Basics
My Spending Report with Budget Watch
Tax Documents
Online check images
Getting an auto loan
Buying

Categories Example

```
new CategoriesWidget({
    numberOfColumns: 2,
    categoryClickEventListeners: [function (element, options) {
        console.log(options.id)
    }]
}).fire().done(function(response, widget) {
    widget.render()
});
```

CategoriesWidget([options])	Example
• **categoriesNotFoundText** (default: "No linked categories found") Type: String Localizable message if no linked categories found	
.fire(): Promise	
Description: fetch data from the server.	
.render(): CategoriesWidget	
Description: renders the view template from fetched data and updates this.el with the new HTML.	

Document Widget

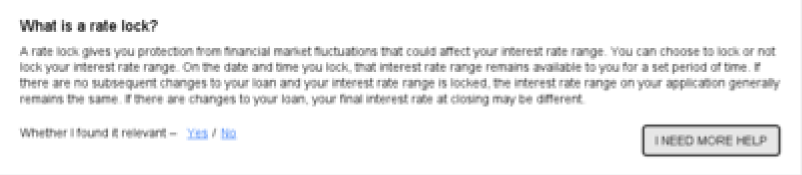
The Document widget displays documents and can have an optional feedback block, a help button, and an embedded block of categories that are associated to the document. The Document widget is dependant on **_gka.getFullContent()** and **_gka.like()** methods in Knowledge Agent.

DocumentWidget([options])

Description: constructor for document widget.

- **el** (default: '#_gk-_wd-_doc')
  Type: String
  Selector for DOM element in which the current widget will be inserted
- **feedbackType** (default: 'BINARY')
  Type: String enum ('NONE', "BINARY")
  Style of rendering the feedback block
- **commentTrigger** (default: 'negative')
  Type: String enum ('negative', 'positive', both)
  When comment block should appear (for example when 'negative', the comment block will be shown just after a negative vote)
- **feedback**
  Type: PlainObject
  - **modified** (default: 'Modified')
    Type: String
    Localizable message for modification date tooltip
  - **views** (default: 'Views')
    Type: String
    Localizable message for views number tooltip
  - **rating** (default: 'Rating')
    Type: String
    Localizable message for rating tooltip
  - **question** (default: *Was this helpful?*)
    Type: String
    Localizable message
  - **defaultAnswer** (default: Thank you for your vote.')
    Type: String
    Localizable message
  - **noCommentAnswer** (default: 'Thank you for your vote.')

Example



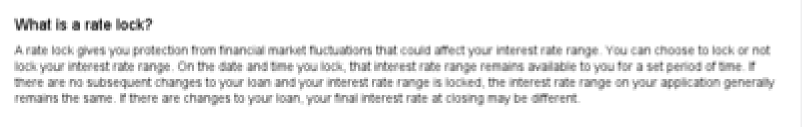
What is a rate lock?

A rate lock gives you protection from financial market fluctuations that could affect your interest rate range. You can choose to lock or not lock your interest rate range. On the date and time you lock, that interest rate range remains available to you for a set period of time. If there are no subsequent changes to your loan and your interest rate range is locked, the interest rate range on your application generally remains the same. If there are changes to your loan, your final interest rate at closing may be different.

Whether I found it relevant – Yes / No

Document Widget Example 1

```
new DocumentWidget({
    documentId: 'knowledgefaq_4',
    feedbackType: 'BINARY',
    feedbackText: 'Whether I found it relevant',
    showHelpButton: true,
    helpButtonText: 'I need more help',
    feedbackSelectEventListeners: [function (element, options) {
        console.log(options.feedbackType);
        console.log(options.rate);
    }],
    helpButtonClickEventListeners: [function (element, options) {
        console.log(options.document.id);
    }]
}).fire({
    kbId: 'knowledgefaq',
    docId: 'knowledgefaq_66'
}).done(function(response, widget) {
    widget.render()
})
```



What is a rate lock?

A rate lock gives you protection from financial market fluctuations that could affect your interest rate range. You can choose to lock or not lock your interest rate range. On the date and time you lock, that interest rate range remains available to you for a set period of time. If there are no subsequent changes to your loan and your interest rate range is locked, the interest rate range on your application generally remains the same. If there are changes to your loan, your final interest rate at closing may be different.

Document Widget Example 2

DocumentWidget([options])	Example

Type: String
Localizable message
- **submitAnswer** (default: 'Thanks, your feedback has been submitted.')
Type: String
Localizable message
- **commentPlaceholder** (default: "Why wasn't this helpful?')
Type: String
Localizable message
- **buttons**
Type: PlainObject
- **yes** (default: 'Yes')
Type: String
Localizable message
- **no** (default 'No')
Type: String
Localizable message
- **submit** (default: 'Submit')
Type: String
Localizable message
- **noComment** (default: 'No Comment')
Type: String
Localizable message
- **showHelpButton** (default: true)
Type: Boolean
Whether or not to display help button
- **helpButtonText** (default: 'I need more help.')
Type: String
Text in help button
	`new DocumentWidget({`
	`    documentId: 'knowledgefaq_4',`
	`    feedbackType: 'NONE',`
	`    showHelpButton: false`
	`}).fire({`
	`    kbId: 'knowledgefaq',`
	`    docId: 'knowledgefaq_66'`
	`}).done(function(response, widget) {`
	`    widget.render()`<br>`})
	`<br>`[[File:Document3.png|thumb|center|400px|`
	`Document Widget Example 3]]<br>`<br>`new DocumentWidget({`
	`    documentId: 'knowledgefaq_4',`
	`    feedbackType: 'RATING',`
	`    feedbackText: '',`
	`    showHelpButton: true,`
	`    helpButtonText: 'Help me'`
	`}).fire({`
	`    kbId: 'knowledgefaq',`
	`    docId: 'knowledgefaq_66'`
	`}).done(function(response, widget) {`
	`    widget.render()`<br>`})`

DocumentWidget([options])	Example
• **showHelpAttachments** (default: true) Type: Boolean Whether or not to display document attachments • **feedbackClickListeners** Type: Function(Element element, options) [] Functions to be called after feedback selected • **element** Type: Element An element in the Document Object Model (DOM) • **options** Type: PlainObject A set of key/value pairs that contains data for listeners. • **document** Type: PlainObject Current document • **rate** Type: Number Chosen rate • **feedbackType** Type: String Current feedbackType • **helpButtonClickListeners** Type: Function(Element element, options)[] Functions to be called after help button clicked • **element** Type: Element An element in the Document Object Model (DOM) • **options** Type: PlainObject	

DocumentWidget([options])	Example
A set of key/value pairs that contains data for listeners. • **document** Type: PlainObject Current document • **localLinkClickEventListeners** Type: Function(Element element)[] Functions to be called after help local link in the document clicked • **element** Type: Element An element in the Document Object Model (DOM)	
.fire(options): Promise	
Description: fetch data from the server according to passed options. • **kbId** Type: String Knowledge base identifier • **docId** Type: String Document identifier	
.render(): DocumentWidget	
Description: renders the view template from fetched data and updates **this.el** with the new HTML.	

Filter Widget

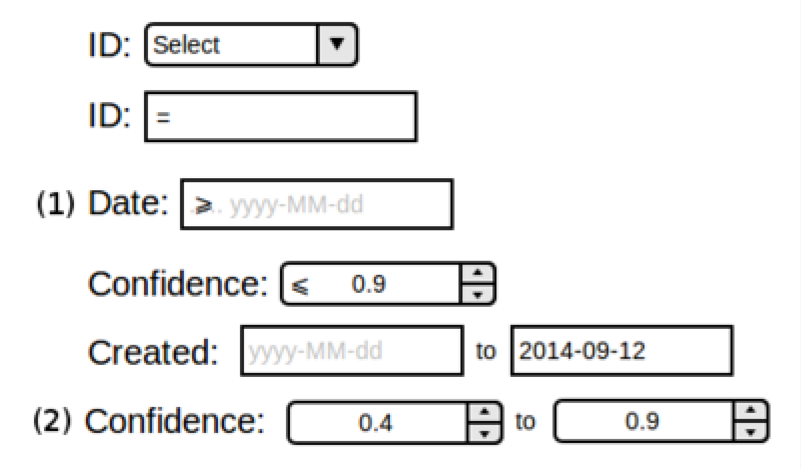
The filter widget displays one single filter item depending on the passed configuration options and can be configured for visualization (timepicker, string input, number input).

FilterWidget(options)

Description: constructor for filter widget.

- **el** (default: '#_gk-_wd-_fl')
  Type: String
  Selector for DOM element in which the current widget will be inserted
- **type** (default: 'INPUT')
  Type: String enum ('INPUT', 'DROPDOWN', 'BETWEEN')
  Basic filter type
- **field** (default: 'id')
  Type: String
  Document field on which the result will be filtered
- **fieldAlias** (default: field)
  Type: String
  The text that will be shown near the input(s)
- **options**
  Type: PlainObject
  A set of key/value pairs which contain additional configuration of the widget
  Widgets with different type expect different set
- **valueChangeEventListeners** (default: [])
  Type: Function(Element element, options)[]
  Functions to be called after value changed
  - **element**
    Type: Element
    An element in the Document Object Model (DOM)
  - **options**
    Type: PlainObject
    A set of key/value pairs that contains data for listeners

Example



ID: Select ▾

ID: =

(1) Date: ≥ yyyy-MM-dd

Confidence: ≤ 0.9

Created: yyyy-MM-dd to 2014-09-12

(2) Confidence: 0.4 to 0.9

Filter Widget Example

```
// Example (1)
var filterWidget = new FilterWidget({
    type: 'INPUT',
    field: 'created',
    fieldAlias: 'Date',
    options: {
        inputType: 'DATE',
        operator: 'GREATER_EQUAL',
    },
    valueChangeEventListeners: [function (element) {
        console.log('date changed')
    }]
})

//Example (2)
var filter2 = new FilterWidget({
```

FilterWidget(options)

"INPUT" type options (default)

Rendering based on HTML tag:

<input type="*text* | *number*">

- **inputType** (default: 'STRING')
  Type: String enum ('STRING', 'NUMERIC', 'DATE')
  Determines which basic type wiil be used for <input>
- **operator** (default: 'EQUAL')
  Type: String enum ('LESS', 'LESS_EQUAL', 'GREATER', 'GREATER_EQUAL', 'EQUAL')
- **value** (default: *)*
  Type: String

"DROPDOWN" type options

Rendering based on HTML tag:

<select>

- **header** (default: 'Select')
  Type: String
  First, default, empty-behaviour <option>
- **values** (default: [])
  Type: String[]
  Other, non-empty <options>'s
- **value** (default: *)*
  Type: String
  Current value. If value in an instance of values, particular <option> obtains selected attribute.

Example

```
    type: 'BETWEEN',
    field: 'confidence',
    fieldAlias: 'Confidence',
    options: {
        separator: 'to',
        inputType: 'NUMERIC',
        from: 0.4,
        to: 0.9
    },
    valueChangeEventListeners: [function (element, options) {
        console.log('one of two values changed')
    }]
})
```

<table>
<tr><th>FilterWidget(options)</th><th>Example</th></tr>
<tr><td>
<table>
<tr><th>"BETWEEN" type options</th></tr>
<tr><td>Rendering based on two of the following HTML tags:

<input type="<i>text</i> | <i>number</i>">

• <b>inputType</b> (default: 'DATE')
Type: String enum ('NUMERIC', 'DATE')
Determines which basic type wiil be used for <input>

• <b>separator</b> (deafult: 'to')
Type: String
The text separator between two <input>'s

• <b>from</b> (default: <i>)</i>
Type: String | Number
First value

• <b>to</b> (default: <i>)</i>
Type: String | Number

Second value</td></tr>
</table>
</td><td></td></tr>
<tr><td colspan="2"><b>.render(): FilterWidget</b></td></tr>
<tr><td colspan="2"><b>Description:</b> renders the view template and updates <b>this.el</b> with the new HTML.</td></tr>
<tr><td colspan="2"><b>.filterJSON(): Filter</b></td></tr>
<tr><td colspan="2"><b>Description:</b> returns compiled filter object related to current widget that can be used in /search API.</td></tr>
</table>

CSS Customization

All widgets have some basic CSS defined and built-in, however Styles can be managed though the browser debugger or easily overridden in a custom CSS file. HTML tags separation is based on classes and all classes used in the Sample UI are divided into different namespaces. For example, widget-classes have a **_gks-_wg-** prefix. Non-widget classes only use the **_gks-** prefix.

Filters

The Result widget supports extensions with custom filters and has a filters property, which is an array of standard objects.

Default filters are configurable along with other Knowledge Base information however only one default filter can be configured per language. Filters can be based on the basic fields of the knowledge article and custom fields (properly defined custom fields only).

All filter criteria is applied using AND logic (for example, createddate > 20140101 00:00:00 AND segment = "premium").

> ### Important
> The Result widget only supports the standard syntax and does not know which filters are enabled in the Knowledge Base.

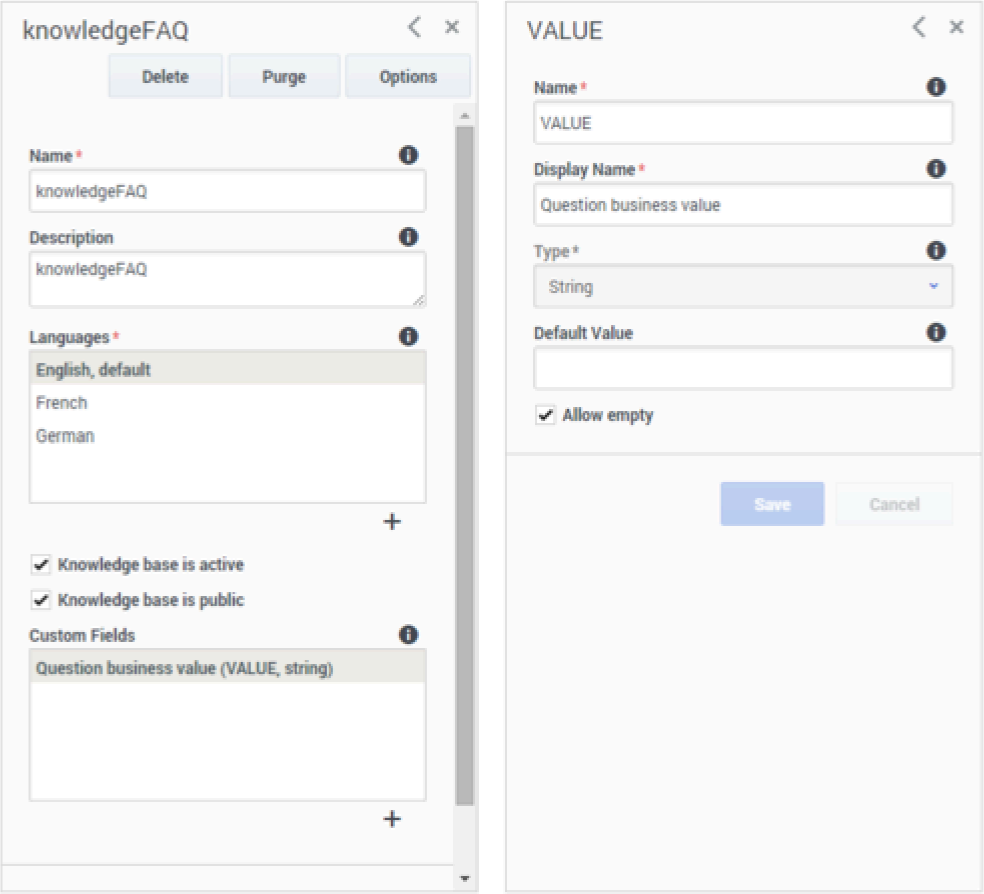
Adding Business Insight

Overview

Some customizations are available when applying different routing strategies to different groups of customers.

Customer categorization in proactive events

1. In **Custom Fields**, create a **Value** for a particular Knowledge Base.



knowledgeFAQ

Name * knowledgeFAQ

Description knowledgeFAQ

Languages *
English, default
French
German

☑ Knowledge base is active
☑ Knowledge base is public

Custom Fields
Question business value (VALUE, string)

VALUE

Name * VALUE

Display Name * Question business value

Type * String

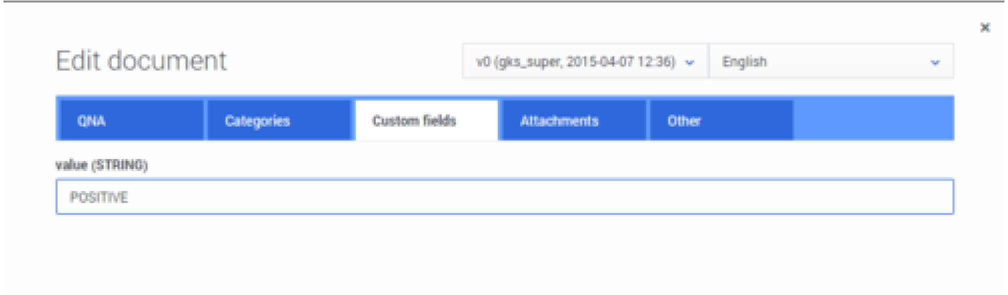
Default Value

☑ Allow empty

Creating a Value

2. Store a business value in **Custom Fields** for each question in Knowledge Base:
   POSITIVE - Customer who needs additional information or help after observing the data. This could be a new client who is looking to purchase a service or request additional services.

NEGATIVE - Customer who searched the info and refused service or found ways to create a claim.
NEUTRAL - No potential positive or negative business impacts.



value (STRING)

POSITIVE

Business Value

3. This Custom Field and its Value is stored in the Knowledge UI page as a hidden attribute.
4. Customize the DLS file to support proactive events on opened documents and attach the value of the Custom Field to the interaction. For example, when clicking the "I need more help" button we can invoke a new event and attach all required information to the interaction:

```
<event id="Help" name="GKnowledge_Help">
    <trigger name="HelpTrigger" element=
"DIV._gk-_wd-_doc-help-bt 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')"/>
    <val name="gks_lang" value="'en'"/>
    <val name="gks_value" value=
"window.localStorage.getItem('businessvalue')"/>
</event>
```

5. Add a new business rule to invoke a new proactive chat on this event:

```
rule "Rule-101 Provide Help"
salience 100000
    agenda-group "level0"
    dialect "mvel"
    when
        $event1: Event(eval($event1.getName().equals('Help')))
    then
        sendEvent($event1, ed, drools);
end
```

6. During parsing, the new variable and its value are obtained from this interaction we can route the interaction using different branches of the business strategy:
NEUTRAL - route via common strategy
NEGATIVE - route with high priority to specific group with escalation specialists
POSITIVE - route with high priority to marketing specialists

Implicit User Feedback

Overview

To improve search quality, gather implicit user feedback via Proactive Engagement integration.

For example, sending additional implicit user feedback automatically in cases such as:

- Customer executed the search, saw search results and left search result page in less then X seconds - > Negative feedback published for all documents in the result set.
- Customer executed the search, saw result, opened the document and left the page in less then X seconds -> Negative feedback published for particular document.
- Customer executed the search, saw result, opened the document and remained on the page for X minutes -> Positive feedback to be published for the document.
- Customer executed the search, saw result, opened the document and opened attachment to the document-> Positive feedback to be published for the document.

Customizing the Script and reconfiguring the routing strategy

1. Add additional JavaScript code to the Web Monitoring Agent on the front-end page to monitor described events.
   For example:

   1. Create a small JavaScript function to wrap sending feedbacks:

      ```
      function feedbackSender
      (decision, behavior, timeoutSeconds){
          this.decision = decision;//positive,negative
          this.timeoutSeconds = timeoutSeconds;
          this.behavior = behavior;//less,more,equal
          this.openTime = null;
      this.sendPositiveFeedback = function(){
          console.log("Positive feedback");
      };
      this.sendNegativeFeedback = function(){
          console.log("Negative feedback");
      };
      this.sendFeedback = function () {
          switch (decision) {
            case "positive":
            this.sendPositiveFeedback();
              break;
            case "negative":
            this.sendNegativeFeedback();
              break;
            default:
              console.log("Sorry, we are out of "
      + this.decision + ".");
      ```

```
        }
};
}
```

2. Add methods to monitor timeout:

```
this.onOpenPage = function () {
        this.openTime = new Date();
};
this.onLeftPage = function () {
    switch (behavior) {
      case "less":
        if ((this.openTime)&&(((new Date())
- this.openTime) < this.timeoutSeconds * 1000)) {
                this.sendFeedback();
            }
        break;
      case "more":
        if ((this.openTime)&&(((new Date())
- this.openTime) > this.timeoutSeconds * 1000)) {
                this.sendFeedback();
            }
        break;
      case "equals":
        if ((this.openTime)&&(((new Date())
- this.openTime) == this.timeoutSeconds * 1000)) {
                this.sendFeedback();
            }
        break;
      default:
        console.log("Sorry, we are out of "
+ this.behavior + ".");
        }
}
```

3. Implement provided business cases using this function. For example:

   - Case 1: Customer executed the search, saw search results and left search result page in less then X seconds - > Negative feedback published for all documents in the result set.

```
firstCaseFeedback = new feedbackSender
('negative','less',5)
firstCaseFeedback.sendNegativeFeedback=function()
{
 _gt.push(['event', {eventName: 'Feedback',
gks_Reason: 'negative', gks_docIds: strArr,
gks_sessionId: gks_sessionId,
 gks_query: gks_query}]);
}
$(window).on('hashchange', function(e){
 if (window.location.hash.indexOf('/search/')
>= 0) {
 firstCaseFeedback.onOpenPage();
 arr=[];
 var $resultDiv = $('._gk-_wd-_rs');
 $resultDiv.ready(function () {
 $resultDiv.each(
 function(index, a) {
 var basicId = $(a).attr('id');
 arr.push(basicId.substring(18, basicId.length))
 }
 );
 })
 strArr = JSON.stringify(arr);
 gks_query = $('#searchContent').val();
 gks_sessionId = window.localStorage.getItem
('sessionId');
 }
});
$(window).on('hashchange', function(e){
 var oldURL = e.originalEvent.oldURL;
 if (oldURL.indexOf('/search/') >= 0) {
 firstCaseFeedback.onLeftPage();
 }
 }
});
```

- Case 2: Customer executed the search, saw result, opened the document and left the page in less then X seconds -> Negative feedback published for particular document.

```
secondCaseFeedback = new feedbackSender
('negative','less',10)
secondCaseFeedback.sendNegativeFeedback=function()
{
    _gt.push(['event', {eventName: 'Feedback',
gks_Reason: 'negative', gks_docIds:
gks_docId, gks_sessionId: gks_sessionId,
 gks_query: gks_query}]);
}

 $(window).on('hashchange', function(e){
    if (window.location.hash.indexOf('/document/')
>= 0) {
    gks_query = $('#searchContent').val();
    gks_sessionId = window.localStorage.getItem
('sessionId');
    gks_docId = window.location.hash.substr
(window.location.hash.length - 36);
    secondCaseFeedback.onOpenPage();
    }
});

$(window).on('hashchange', function(e){
    var oldURL = e.originalEvent.oldURL;
    if (oldURL.indexOf('/document/') >= 0) {
        secondCaseFeedback.onLeftPage();
    }
});
```

- Case 3: Customer executed the search, saw result, opened the document and remained on the page for X minutes -> Positive feedback to be published for the document.

```
thirdCaseFeedback = new feedbackSender
('positive','more',20)
thirdCaseFeedback.sendPositiveFeedback=function()
{
    _gt.push(['event', {eventName: 'Feedback',
gks_Reason: 'positive', gks_docIds:
gks_docId, gks_sessionId: gks_sessionId,
 gks_query: gks_query}]);
}

thirdCaseFeedback.onOpenPage=function(){
    thirdCaseFeedback.openTime = new Date();
    setTimeout(function()
{thirdCaseFeedback.onLeftPage();},
(thirdCaseFeedback.timeoutSeconds+1)*1000);
}

$(window).on('hashchange', function(e){
    if (window.location.hash.indexOf('/document/')
>= 0) {
    gks_query = $('#searchContent').val();
    gks_sessionId = window.localStorage.getItem
('sessionId');
    gks_docId = window.location.hash.substr
(window.location.hash.length - 36);
    thirdCaseFeedback.onOpenPage();
    }
});
```

2. Create a business rule to invoke interaction on this event:

```
rule "Rule-101 Feedback"

salience 100001

    agenda-group "level0"

    dialect "mvel"

    when

         $event1: Event(eval($event1.getName().equals
('Feedback')))

    then

        sendEvent($event1, ed, drools);

end
```

3. Modify the strategy to route the interaction invoked by these events in a separate branch.
4. Use modules to send REST requests in these branches to publish feedback to the Knowledge server:

```
Positive feedback:

POST
URL: http://<host>:<port>/gks-server/v1/feedback/
knowledgefaq/documents/<docId>/
vote?relevant=true&sessionId=<sessionId>
BODY: {"query":"<query>"}

Negative feedback:

POST
URL: http://<host>:<port>/gks-server/v1/feedback/
knowledgefaq/documents/<docId>/
vote?relevant=false&sessionId=<sessionId>
BODY: {"query":"<query>"}
```

Improving Contact Us Form

Overview

You can utilize Knowledge Center capabilities to assist customers as they fill out forms on your corporate web site (for example, when providing feedback). This integration provides suggested answers to a customer's query by utilizing the **Category** selections, and keywords used in the **Subject** line. This simple guide will show you how Knowledge Center can be easily integrated into a Webform.

Webform integration

Before integration

Integrate the knowledge with a simple feedback form:

HTML

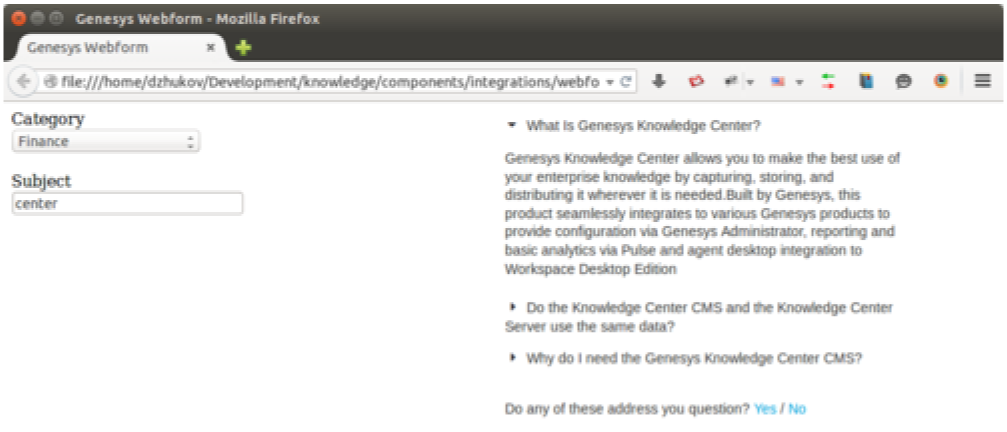
```
<div class="container">
    <div class="main">
        <div>
            <label>
                Category <br>
                <select class="categories"></select>
            </label>
        </div>
        <br>

        <div>
            <label>
                Subject <br>
                <input autocomplete="off" class="search" placeholder=
"What is knowledge Center?" type="text">
            </label>
        </div>
    </div>

    <!-- MAIN DIV -->
    <div class="gkc-webform"></div>
</div>
```

Java Script

```
$(document).ready(function () {
    webform.init({
        host: 'http://%your_server_host%/gks-server/v1',
        categories: {
            'Finance':                  'knowledgefaq',
            'Account':                  'knowledgefaq',
            'Signing in':               'knowledgefaq',
            'Buying':                   'knowledgefaq',
            'Shipping & tracking':      'knowledgefaq',
            'Booking trips ':           'knowledgefaq',
            'Gifts ':                   'knowledgefaq',
```

Java Script

```
            'Mobile ':                  'knowledgefaq',
            'Email subscriptions ':     'knowledgefaq',
            'Restaurant reservations ': 'knowledgefaq'
        }
    });

    webform.markKbsDropdownWithMap('.gkc-kbs');
    webform.markSearchInput('.gk-search');
})
```


Genesys Webform - Mozilla Firefox
Genesys Webform
file:///home/dzhukov/Development/knowledge/components/integrations/webfo

Category: Finance

Subject: center

▾ What Is Genesys Knowledge Center?

Genesys Knowledge Center allows you to make the best use of your enterprise knowledge by capturing, storing, and distributing it wherever it is needed.Built by Genesys, this product seamlessly integrates to various Genesys products to provide configuration via Genesys Administrator, reporting and basic analytics via Pulse and agent desktop integration to Workspace Desktop Edition

▸ Do the Knowledge Center CMS and the Knowledge Center Server use the same data?

▸ Why do I need the Genesys Knowledge Center CMS?

Do any of these address you question? Yes / No

Example of simple integration.

Integration steps

1. Add all files (1 .css and 1 .js) from folder **<knowledge_center_server_root>\server\tools\webform** to site context. Core .js file applies only to rendering results ("Suggestions" window in the above figure).
2. Configure added script through **window.webform** variable:
   - Use **webform.init()** method to pass general options
   - Use **webform.markKbsDropdownWithMap()** to mark a specific <select> tag as a Categories selector.
   - Use **webform.markSearchInput()** to mark a specific <input> tag as to which Knowledge Center is performing the search.

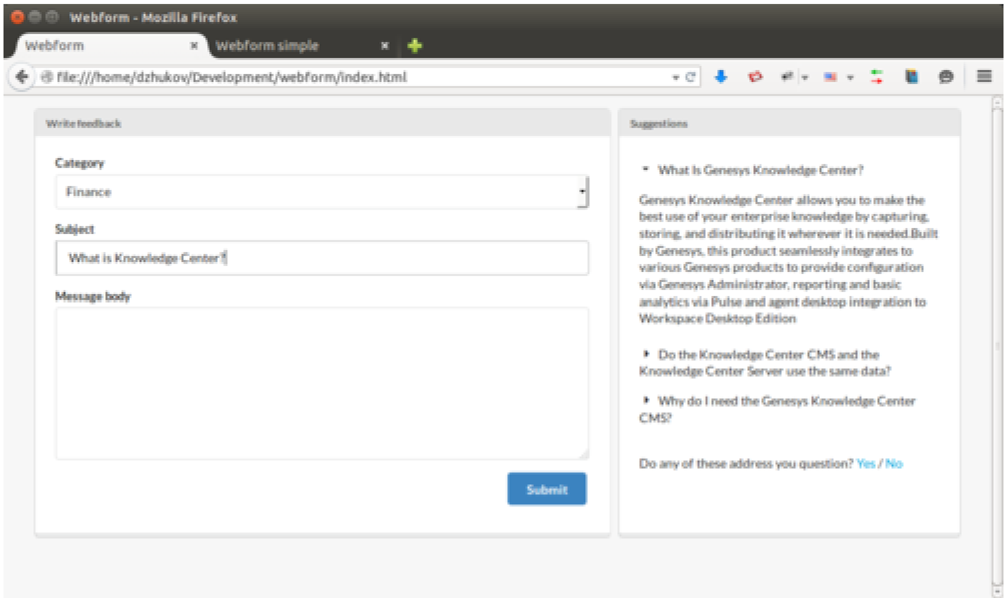
> **Important**
>
> Examples of integrations can be found in <knowledge_center_server_root>/server/tools/webform/example folder.

After integration

As the result of this integration you now have a feedback form that pro-actively looks up the knowledge related to a customer inquiry and displays possible suggestions to the customer.

Important

WebForm can also contain a more complex demo based on Semantic-UI CSS-framework. See a complex integration at <knowledge_center_server_root>/server/tools/example/complex.html


Webform - Mozilla Firefox
Webform
Webform simple
file:///home/dzhukov/Development/webform/index.html
Write feedback
Suggestions
Submit

Category: Finance

Subject: What is Knowledge Center?

Message body

- What Is Genesys Knowledge Center?

Genesys Knowledge Center allows you to make the best use of your enterprise knowledge by capturing, storing, and distributing it wherever it is needed.Built by Genesys, this product seamlessly integrates to various Genesys products to provide configuration via Genesys Administrator, reporting and basic analytics via Pulse and agent desktop integration to Workspace Desktop Edition

- Do the Knowledge Center CMS and the Knowledge Center Server use the same data?
- Why do I need the Genesys Knowledge Center CMS?

Do any of these address you question? Yes / No

Knowledge suggestions displayed

WebForm Widget API

Use the following information to integrate the WebForm Widget API on your html page.

<table>
<tr><th>webform.initialize(options)</th><th>Example</th></tr>
<tr><td>
<ul>
<li><b>Description</b>: Configure the WebForm widget.
<ul>
<li><b>options</b> Type: PlainObject A set of key/value pairs that configure the Agent.
<ul>
<li><b>host</b> Type: string A network host where Knowledge API is stored.</li>
<li><b>categories</b> Type: PlainObject A map of the predefined labels of categories. Keys are a labels and values are knowledgebase IDs.</li>
</ul>
</li>
</ul>
</li>
</ul>
</td><td>
<pre>
webform.init({
    host: 'http://192.168.66.176:9095/gks-server/v1',
    categories: {
        'Finance':   'financefaq',
        'Account':   'accounting',
        'Signing in':   'webfaq',
        'Gifts ':   'knowledgefaq',
        'Mobile ':   'mobilefaq',
        'Email subscriptions ':   'webfaq'
    }
});
</pre>
</td></tr>
</table>

<table>
<tr><th>webform.markKbsDropdownWithMap(selector, callback)</th><th>Example</th></tr>
<tr><td>
<ul>
<li><b>Description</b>: Create widget-dependent dropdown based on passed selector of <select> tag. This method uses <b>categories</b> passed to the <b>.initialize</b> method.
<ul>
<li><b>selector</b> Type: jQuery Selector A string containing a selector expression to match elements against.</li>
<li><b>callback</b> Type: Function() A function to be called after main operations.</li>
</ul>
</li>
</ul>
</td><td>
<pre>
webform.markKbsDropdownWithMap('.gkc-kbs', function () {
    $('.gkc-kbs').dropdown();
});
</pre>
</td></tr>
</table>

<table>
<tr><th>webform.markKbsDropdown(selector, callback)</th><th>Example</th></tr>
<tr><td>
<ul>
<li><b>Description</b>: Create widget-dependent dropdown based on passed selector of <select> tag. This method does not use <b>categories</b> passed to the <b>.initialize method</b>. It will load knowledge bases with labels directly from the Knowledge API.
<ul>
<li><b>selector</b> Type: jQuery Selector A string containing a selector expression to match elements against.</li>
</ul>
</li>
</ul>
</td><td>
<pre>
webform.markKbsDropdown('.gkc-kbs', function () {
    $('.gkc-kbs').dropdown();
});
</pre>
</td></tr>
</table>

webform.markKbsDropdown(selector, callback)	Example
- **callback** Type: Function(kbs) A function to be called after main operations.	

webform.markSearchInput(selector)	Example
- **Description**: Create widget-dependent search input based on passed selector of <input> tag. - **selector** Type: jQuery Selector A string containing a selector expression to match elements against.	`webform.markSearchInput('.gk-search');`

webform.getKbs(callback)	Example

- **Description**: Retrieves knowledge bases from the Knowledge API.
- **callback** Type: Function(kbs) A function to be called after knowledge bases have been loaded.
	`webform.getKbs(function (kb) {`
	`    console.log(kb)`
	`})`

webform.makeSearch(query, callback)	Example

- **Description**: Searches documents from the Knowledge API based on query.
- **query** Type: string User typed query string
- **callback** Type: Function(documents) A function to be called after documents have been loaded.
	`webform.makeSearch('What is Knowledge Center?', function`
	`(documents) {`
	`    console.log(documents)`
	`})`