



# Oracle Knowledge for RightNow Integration Guide

---

*Integrating Oracle Knowledge with RightNow  
Agent Desktop and Customer Portal*

---

Release 8.5.1  
Document Number OKIC-RNI85-01  
May, 2013

---

## COPYRIGHT INFORMATION

Copyright © 2002, 2013, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

### U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are “commercial computer software” or “commercial technical data” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. Other names may be trademarks of their respective owners.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

---

# Contents

---

<b>Preface About This Guide</b>	<b>1</b>
In This Guide	1
Examples of Product Screens and Text	1
Operating System Variations in Examples and Procedures	1
References to Web Content	2
	2
<b>Getting Started with Oracle Knowledge for Right Now</b>	<b>3</b>
Oracle Knowledge for RightNow Agent Desktop	3
Prerequisites for Integrating Oracle Knowledge with Agent Desktop and Customer Portal	3
Downloading and Extracting the RightNow Installation Files	4
Oracle Knowledge for RightNow Customer Portal	4
Oracle Knowledge for RightNow Customer Portal Operational Requirements	5
Third-Party Cookie Enablement	5
Extracting the Customer Portal Installation Files	5
Components of Oracle Knowledge for RightNow	5
<b>Adding Oracle Knowledge to Customer Portal</b>	<b>6</b>
Editing the Template	6
Adding Widgets for Oracle Knowledge	7
Integrating Oracle Knowledge	8
Deploying Customer Portal	10
<b>Adding Oracle Knowledge to Agent Desktop</b>	<b>11</b>
Adding Oracle Knowledge to Agent Desktop	11
Installing the Oracle Knowledge Add-In	11

**11**

Granting Access to Applicable User Profiles .....	12
Adding Oracle Knowledge to Incident Workspaces .....	12
Testing the Integration .....	13
Configuring the Oracle Knowledge Add-In Options .....	15

**Customizing Applications with Oracle Knowledge Widgets ..... 18**

Getting Started with Widgets .....	19
Building an HTML Page .....	19
Adding Widget Dependencies .....	20
Adding Widget Events and Interaction .....	21
Extending Widget Dependencies .....	21
Widget Details .....	21
OKBase .....	22
LocaleUtility .....	22
OKUtility .....	22
AnswerView .....	24
AnswerList .....	25
Category .....	28
CategoryExplorer .....	30
Channel .....	32
Facets .....	33
Search .....	34
Result .....	39
CaseDeflection .....	42
Exposed DOM Elements .....	45
CSS .....	45

# About This Guide

This guide provides detailed instructions and supporting information for installing, configuring, and deploying Oracle Knowledge for RightNow for use with an Oracle Knowledge application. It is intended for application developers and administrators who need to plan for and perform integration of Oracle Knowledge applications with Oracle Knowledge RightNow Agent Desktop and Oracle Knowledge Customer Portal.

This preface includes information on:

- **In This Guide**
- **Examples of Product Screens and Text**
- **Operating System Variations in Examples and Procedures**
- **References to Web Content**

## In This Guide

The Oracle Knowledge for RightNow Integration Guide is divided into the following sections:

<b>Chapter 1, “Getting Started with Oracle Knowledge for Right Now”</b>	This section describes Oracle Knowledge for RightNow Agent Desktop and Customer Portal and their integration requirements.
<b>Chapter 2, “Adding Oracle Knowledge to Customer Portal”</b>	This section describes how to integrate Oracle Knowledge with RightNow Customer Portal.
<b>Chapter 3, “Adding Oracle Knowledge to Agent Desktop”</b>	This section describes how to integrate Oracle Knowledge with incident workspaces, create workspaces, and configure options..
<b>Chapter 4, “Customizing Applications with Oracle Knowledge Widgets”</b>	This section explains how to use Oracle Knowledge widgets to embed knowledge within applications, using your web application development skills, such as HTML, CSS, and JavaScript.

## Examples of Product Screens and Text

The product screens, screen text, and file contents depicted in the documentation are examples. We attempt to convey the product's appearance and functionality as accurately as possible; however, the actual product contents and displays may differ from the published examples.

## Operating System Variations in Examples and Procedures

We generally use Linux screen displays and naming conventions in our examples and procedures. We include other operating system-specific procedures or steps as noted in section headings, or within topics, as appropriate.

We present command syntax, program output, and screen displays:

- in Linux format first
- in other Unix-specific variants only when necessary for proper operation or to clarify functional differences
- in Windows format only when necessary for clarity

## References to Web Content

For your convenience, this guide refers to Uniform Resource Locators (URLs) for resources published on the World Wide Web, when appropriate. We attempt to provide accurate information; however, these resources are controlled by their respective owners and are therefore subject to change at any time.

# Getting Started with Oracle Knowledge for Right Now

Oracle Knowledge for RightNow provides a complete intelligent search interface that enables contact center agents end users to quickly and easily find accurate answers to customer inquiries from within the RightNow Agent Desktop and Customer Portal.

Oracle Knowledge for RightNow leverages Oracle Knowledge's patented Intelligent Search technology to find exact answers to inquiries based on their meaning, and to search unstructured content, structured data sources and transactional business applications in parallel. Oracle Knowledge technology can automatically incorporate customer context, call context, and CRM contextual information in the search for answers to customer inquiries.

## Oracle Knowledge for RightNow Agent Desktop

The Oracle Advanced Knowledge for RightNow Agent Desktop integration embeds Oracle Knowledge search functionality directly within the RightNow Agent Desktop to maximize agent productivity and minimize keystrokes, and improve call resolution rates. The Oracle Knowledge search results include relevant excerpts that have a high probability of answering an inquiry based on its intent.

Oracle Advanced Knowledge for RightNow significantly streamlines the call wrap-up process by automatically providing embedded links to associate the right enterprise knowledge with each service request resolution task.

You can configure each incident workspace with Oracle Knowledge options according to the profiles assigned to the workspace.

## Prerequisites for Integrating Oracle Knowledge with Agent Desktop and Customer Portal

Before you can integrate Oracle Knowledge with Agent Desktop and Customer Portal, you must have RightNow environment permissions.

Administrators need the following permissions:

- Upload add-ins
- Create and edit profiles
- Workspace designer
- Object designer
- Configuration

- Session authentication
- Open custom report *OK – LinkedAnswersReport* that was imported previously

End users need the following permissions:

- Session authentication
- Read, Add/Edit, and Response for Incidents
- Create and Read for custom objects
- Open custom report *OK – LinkedAnswersReport* that was imported previously

## Downloading and Extracting the RightNow Installation Files

Extract the following Zip files in any convenient location:

- 1 Download the *RightNowOKiConnectArtifacts.zip* file located in your installation directory.
- 2 Unzip the *RightNowOKiConnectArtifacts.zip* file.  
The extraction creates two directories, AgentDesktop and CustomerPortal. For this part of the installation, use only the files in the AgentDesktop directory.
- 3 Sign in to your RightNow site as administrator. For information on signing in to RightNow, refer to *Click Once Deployment*.
- 4 Unzip the *LinkedAnswersCustomObject.zip* file (located in the AgentDesktop directory). This file contains XML that adds the LinkedAnswers custom object as the Linked Answers tab in the workspace. For information on editing, importing, and deploying custom objects, refer to *Custom Objects*.
- 5 Unzip the *LinkedAnswersReport.zip* file (located in the AgentDesktop directory) to extract the *LinkedAnswersReport.xml*. This XML file contains the report definition for the report used by the Oracle Knowledge add-in.
- 6 Use the Reports explorer to import the *LinkedAnswersReport.xml* file extracted previously and save the report with the name “OK - Linked Answers Report” in the Public Reports folder. For more information, refer to *Reports Explorer* and *Importing Reports*.
- 7 Follow the installation instructions in [Chapter 3, Adding Oracle Knowledge to Agent Desktop](#).

## Oracle Knowledge for RightNow Customer Portal

Customer Portal is the client-facing site of RightNow that administrators can use to modify customer service web pages.

Oracle Knowledge brings advanced knowledge capabilities into Customer Portal by embedding Oracle Knowledge-powered knowledge artifacts into Customer Portal. The self-sustained knowledge artifacts deliver advanced knowledge to customers. The artifacts also embed Oracle Knowledge's search functionality directly within the RightNow Customer Portal to deliver intent-based, natural language powered search. Oracle Knowledge search results include relevant excerpts that have a high probability of answering customers' questions based on its intent.



Unlike typical self-service solutions that merely “parse the words,” Oracle Knowledge uses natural language processing to understand the true intent of each inquiry. Then, using other search enhancement features such as industry-specific libraries, it finds the best-possible answer. Oracle Knowledge for RightNow Customer Portal transforms customers’ self-service experience—increases online resolution rates, reduces calls into the contact center, and substantially reduces support costs.

## Oracle Knowledge for RightNow Customer Portal Operational Requirements

### Third-Party Cookie Enablement

Oracle Knowledge for RightNow Customer Portal requires the use of browsers with third-party cookies enabled. Some browsers disable third-party cookies by default; Oracle Knowledge for RightNow Customer Portal customers need to ensure that end-users access the Oracle Knowledge functionality using browsers that have third-party cookies enabled.

## Extracting the Customer Portal Installation Files

Extract the following Zip files in any convenient location:

- 1 Unzip the *RightNowOKiConnectArtifacts.zip* file located in your installation directory. The extraction creates two directories, AgentDesktop and CustomerPortal. For this part of the installation, use only the files in the CustomerPortal directory.
- 2 Unzip file *cp.zip*.
- 3 Follow the installation instructions in [Chapter 2, Adding Oracle Knowledge to Customer Portal](#).

## Components of Oracle Knowledge for RightNow

Oracle Knowledge for Customer Portal contains the following components:

- template
- home page
- Ask a Question page
- supporting files
- Browse page
- custom widget

Oracle Knowledge for Agent Desktop also includes the following options:

- The SearchImmediately option provides the ability to determine whether a search runs automatically when a user selects the Oracle Knowledge workspace tab.
- The iConnectProperties option provides the ability to apply additional incident information to further filter the Oracle Knowledge search results.

# Adding Oracle Knowledge to Customer Portal

This document describes the procedures for adding Oracle Knowledge to your customer portal. It includes steps for editing the template, home page, and Ask a Question page, as well as editing the supporting files, adding a browse page, and adding the custom widgets.

The following placeholder is used in the sample code in this document. Before you begin, be sure you know the equivalent values for your site so you can use them to replace the placeholder. The bold italic text in code examples indicates where you need to add information specific to your site.

**<your\_site>**-This is the URL your customers use to access your organization's customer portal. It will resemble `https://your-company-name.custhelp.com/`.

The procedures in this document also assume that you have received the source files necessary for implementing Oracle Knowledge and that they are available for you to copy. When unzipped, the source file contains two folders: development (which includes the views and widgets subfolders) and assets.

## Editing the Template

Because your changes to the template will be extensive, we recommend that you back up the *standard.php* template found in the reference implementation before making your changes.

The template changes accomplish the following tasks:

- Add a Browse navigation tab to the template to replace the reference implementation's Answer tab.
- Replace the Search field in the template sidebar with code for Oracle Knowledge searching.

To edit the template for Oracle Knowledge, use the following procedure:

- 1 Using a WebDAV client, open the *standard.php* file in your site's `/cp/development/views/templates` folder. Save the original file with a new name, for example, *standard\_original.php*, and then close it.
- 2 Open *standard.php* to edit as your new standard template.
- 3 Locate the following line of code:

```
<link rel="icon" href="images/favicon.png" type="image/png"/>
```

- 4 Add the following line of code immediately above the code in step 3.

```
<link type="text/css" rel="stylesheet" href="/euf/assets/css/oracleKnowledge_ltr.css">
```

- 5 Remove the Answers tab and Community tab by deleting the following lines of code:

```
<li><rn:widget path="navigation/NavigationTab" label_tab="#rn:msg:ANSWERS_HDG#"
link="/app/answers/list" pages="answers/list, answers/detail, answers/intent"/></li>
```

```
<rn:condition config_check="COMMUNITY_ENABLED == true">
<li><rn:widget path="navigation/NavigationTab" label_tab="#rn:msg:COMMUNITY_LBL#"
link="#rn:config:COMMUNITY_HOME_URL:RNW##rn:community_token:?" external="true"/></li>
</rn:condition>
```

- 6 Add the following code to create a Browse tab to replace the code that you just deleted for the Answers tab and Community tab in step 5.

**Note:** The Browse page referenced in the code has not yet been added to the page set.

```
<li><rn:widget path="navigation/NavigationTab" label_tab="#rn:msg:BROWSE_LBL#" link=
"/app/browse" pages="browse" /></li>
```

- 7 To replace the Search field in the sidebar, locate and delete the following line of code:

```
<rn:widget path="search/SimpleSearch"/>
```

- 8 Add the following code below the code that you located in step 7. Add the information for your site where you see the bold italic text placeholders. Also check that the directories indicated in bold italics match your infocenter file structure.

```
<rn:widget path="oracleKnowledge/SidebarSearch" base_url="http://
infocenterhost:infocenterport/<infocenterappname>/" widget_dir="http://
infocenterhost:infocenterport/<infocenterappname>/apps/infocenter/resources/js/
OKWidgets/" />
```

- 9 Save *standard.php*.

## Adding Widgets for Oracle Knowledge

The source files include files for the following widgets:

Widget	Displays
Browse	Oracle Knowledge information on the Browse page of the customer portal
Home	Oracle Knowledge searching and information on the Home page of the customer portal
OKDeflectionFormSubmit	Oracle Knowledge information on the Ask a Question page of the customer portal
SidebarSearch	Oracle Knowledge searching on the side bar of the customer portal

To install the Oracle Knowledge widgets, use the following procedure:

- 1 Using the unzipped source file, copy the oracleKnowledge folder from the *development/widgets/custom/* source folder, and paste the folder into your customer portal file structure at */cp/development/widgets/custom*. The file structure in your customer portal will be as follows:

```
/cp/development/widgets/custom/oracleKnowledge/Browse/1.0
/cp/development/widgets/custom/oracleKnowledge/Home/1.0
/cp/development/widgets/custom/oracleKnowledge/
```

```
OKDeflectionFormSubmit/1.0
/cp/development/widgets/custom/oracleKnowledge/SidebarSearch/1.0.
```

## 2 Activate the Oracle Knowledge widgets.

- a Go to `https://<your_site>/ci/admin/versions/manage`.
- b In the left column, locate and select the **oracleKnowledge/Browse** widget.  
Custom widgets are listed at the top, and the widgets you just created are dimmed to indicate that they are not yet active.
- c Click **Start Using This Version** in the upper right corner of the page.  
The widget is now activated and can be used on any page of your customer portal.
- d Repeat steps b and c for the other widgets:
  - oracleKnowledge/Home
  - oracleKnowledge/OKDeflectionFormSubmit
  - oracleKnowledge/SidebarSearch

For more information about widgets for Oracle Knowledge, see “Customizing Applications with Oracle Knowledge Widgets”.

# Integrating Oracle Knowledge

The following procedures describe how to import the Oracle Knowledge files into your customer portal files.

To integrate Oracle Knowledge files, use the following procedure:

- 1 Copy the following Oracle Knowledge CSS file from the source folder *assets/css* to your */cp/assets/css*.  
*oracleKnowledge\_ltr.css*- This CSS file is for languages that use left-to-right writing.
- 2 Copy the *oracleKnowledge* folder from the *assets/images* folder of the source files to */cp/assets/images*. These images are required by the CSS files for Oracle Knowledge.
- 3 Edit the Support Home, Browse, and Ask a Question pages as described in the following procedures.

To edit the Support Home page, use the following procedure:

- 1 Using a WebDAV client, open the *home.php* file in the */cp/development/views/pages* folder and save the original file with a new name, for example, *home\_original.php*, and then close it.
- 2 Open the file *home.php* so you can edit it to be your new home page.
- 3 Delete every line of code in the file except the first one:
 

```
<rn:meta title="#rn:msg:SHP_TITLE_HDG#" template="standard.php" clickstream="home"/>
```
- 4 Add the following code after the first line. Add the information for your site where you see the bold italic text placeholders. Also check that the directories indicated with bold italics match your customer infocenter file structure.

```
<rn:widget path="oracleKnowledge/Home"
base_url="http://infocenterhost:infocenterport/<infocenterappname>/" widget_dir=
"http://infocenterhost:infocenterport/<infocenterappname>/apps/infocenter/
```

```
resources/js/OKWidgets/" />
```

##### 5 Save *home.php*.

To add the Browse page, use the following procedure:

- 1 Using a WebDAV client, copy the *browse.php* file from the *development/views/pages* source folder and paste it into */cp/development/views/pages* in your customer portal file structure.
- 2 Locate the following code and edit it with information for your site.

```
base_url="http://infocenterhost:infocenterport/<infocenterappname>/"
```

- 3 Locate the following code and verify that the directory matches your customer infocenter file structure.

```
widget_dir="http://infocenterhost:infocenterport/<infocenterappname>/apps/  
infocenter/resources/js/OKWidgets/"
```

- 4 Locate the following code and edit it with information for your site.

```
default_channel="FAQ"
```

##### 5 Save *browse.php*.

To edit the Ask a Question page, use the following procedure:

- 1 Using a WebDAV client, open the *ask.php* file in the */cp/development/views/pages* folder and save the original file with a new name, for example, *ask\_original.php*, and then close it.
- 2 Open *ask.php* so you can edit it to be your new Ask a Question page.
- 3 Delete the following lines of code.

```
<rn:widget path="input/FormSubmit" label_button="#rn:msg:CONTINUE_  
ELLIPSIS_CMD#" on_success_url="/app/ask_confirm" error_location=  
"rn_ErrorLocation"/>  
<rn:condition answers_viewed="2" searches_done="1">  
<rn:condition else/>  
<rn:widget path="input/SmartAssistantDialog"/>  
</rn:condition>
```

- 4 Add the following code to replace the code you deleted in step 3. Add the information for your site where you see the bold italic text placeholders. Also check that the directories indicated with bold italics match your customer portal file structure.

```
<rn:widget path="oracleKnowledge/OKDeflectionFormSubmit"  
label_button="#rn:msg:CONTINUE_ELLIPSIS_CMD#"  
base_url="http://infocenterhost:infocenterport/<infocenterappname>/"  
widget_dir="http://infocenterhost:infocenterport/<infocenterappname>/apps/  
infocenter/resources/js/OKWidgets/" on_success_url="/app/ask_confirm"  
error_location="rn_ErrorLocation" />
```

##### 5 Save *ask.php*.

## Deploying Customer Portal

This section contains a brief overview of staging and promoting customer portal after modifying it. If you need more detailed information, refer to *Staging and Promoting the Customer Portal*.

To deploy the modified customer portal, use the following procedure:

- 1 Log directly in to the Deploy page on the Customer Portal Administration site by typing `https://<your_site>/ci/admin/deploy/index` in a web browser.
- 2 Click **Stage** on the Deploy page to open the Select Files page.  
A list of the files you modified in the development area appears. Verify that the following files are shown and that the Action column for each file contains Copy to Staging:
  - /development/views/pages/ask.php
  - /development/views/pages/browse.php
  - /development/views/pages/home.php
  - /development/views/templates/standard.php
  - /development/widgets/custom/oracleKnowledge/Browse/1.0
  - /development/widgets/custom/oracleKnowledge/Home/1.0
  - /development/widgets/custom/oracleKnowledge/OKDeflectionFormSubmit/1.0
  - /development/widgets/custom/oracleKnowledge/SidebarSearch/1.0
- 3 Click **Next**.  
The Select Version Changes page opens.
- 4 Click the drop-down menu for Push All Framework and Widget Version Changes? and select **Yes**.  
The Select Configurations window opens.
- 5 Click **Next**.  
The Stage page opens.
- 6 Click the **Stage** button, and then click **Stage** on the confirmation window.  
When the staging process is complete, a Staging Completed Successfully message appears. Now you can promote your changes to the production site that your customers access.
- 7 Click the **Deploy** tab and select **Promote** on the Deploy page.  
The Promote page opens.
- 8 Click the **Promote** button, and then click **Promote** on the confirmation window.  
When the Promote process is complete, a Promote Completed Successfully message opens.

# Adding Oracle Knowledge to Agent Desktop

This process adds the OK Knowledge tab and the LinkedAnswers tab to incident workspaces. This section describes how to add and configure Agent Desktop in your incident workspaces.

## Adding Oracle Knowledge to Agent Desktop

To allow staff members access to the new components, you must configure staff profiles for incident workspaces.

**Important!** Before you proceed, make sure that you have fulfilled all prerequisites and have run the installation files, as described in “Getting Started with Oracle Knowledge for Right Now”.

## Installing the Oracle Knowledge Add-In

To give access to user profiles, use the following procedure:

- 1 In the left navigation panel, navigate to **Configuration, Site Configuration**, and then **Add-In Manager**.
- 2 Create a new add-in.
- 3 Select *OKiConnectAddin.zip* and complete the upload steps.
- 4 In the Server Configuration Properties section of the Add-In Manager, set the value of the ServerName and ServerPort properties to the correct values for your server, as shown in the following figure.

Server Configuration Properties		
Class Name	Property Name	Value
OKiConnectAddin.OracleKnowledgeiConnectFactory	AppContext	iconnect
OKiConnectAddin.OracleKnowledgeiConnectFactory	protocol	http
OKiConnectAddin.OracleKnowledgeiConnectFactory	ServerName	slc03smc
OKiConnectAddin.OracleKnowledgeiConnectFactory	ServerPort	8226

## Granting Access to Applicable User Profiles

- 1 On the Permissions page on the Profiles editor, for every profile that will use the Oracle Knowledge add-in, allow Create, Read, and Personal Notes permissions for the new LinkedAnswers custom object. For information on editing profile permissions, refer to *Customizing Profiles*.
- 2 In the Profiles Allowed to Access section of the Add-In Manager, grant access for every profile that will use the Oracle Knowledge add-in. If your RightNow site has multiple interfaces, you must grant access to the add-in for each profile and for each interface the profile has access to. Refer to *Using the Add-In Manager*.

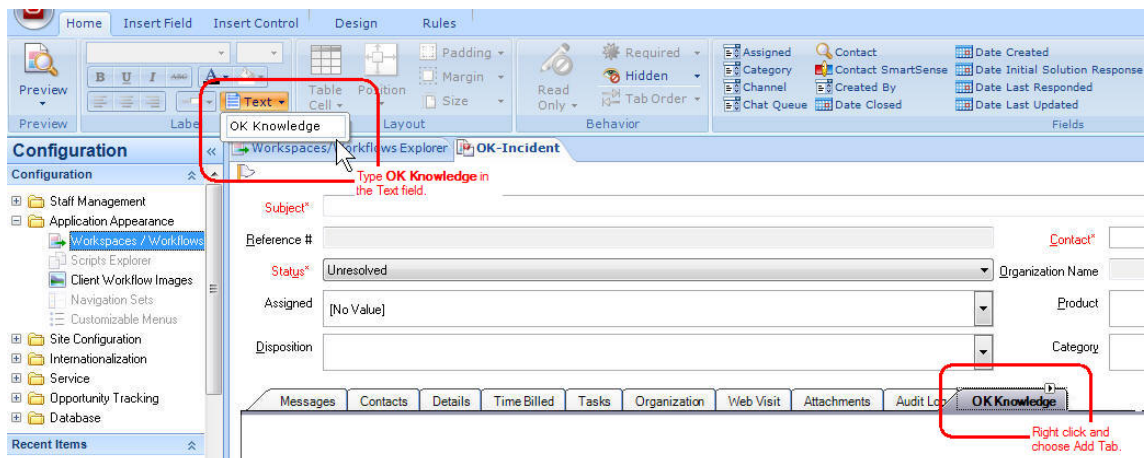
## Adding Oracle Knowledge to Incident Workspaces

In a workspace, you can manage and record the actions taken to resolve an incident. Identify an existing incident workspace where you would like to expose the Oracle Knowledge add-in. If you do not have a workspace, you can create a new one.

Oracle Knowledge searches on the incident Subject name for suggested solutions and inserts the text of or links to the solutions into the incident workspace. The results may include text and links from demos, FAQs, or blogs, and links to discussion topics.

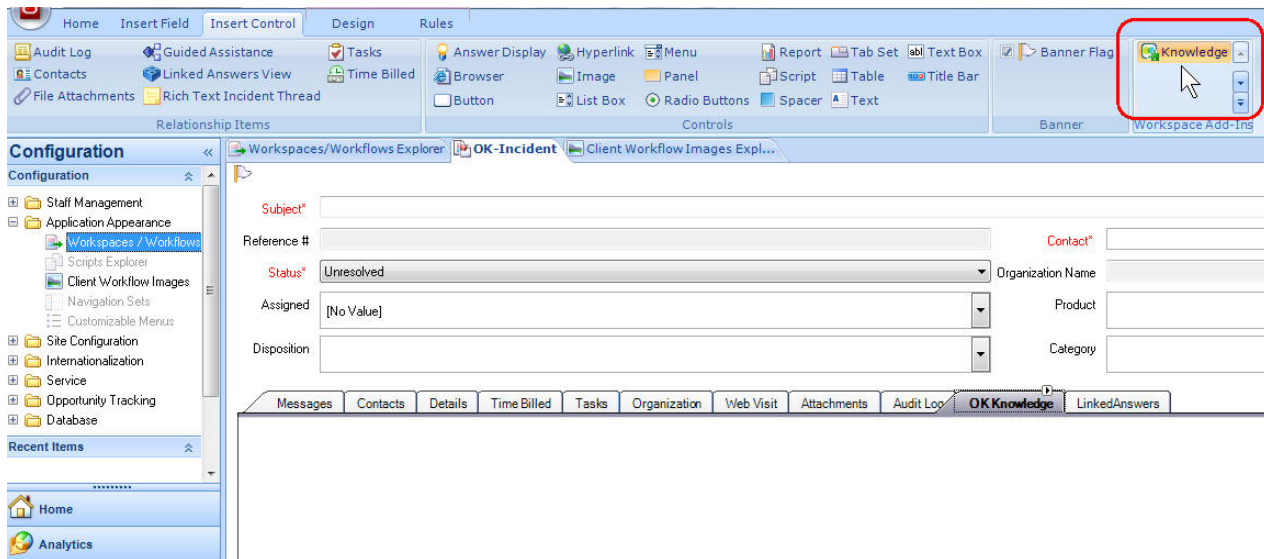
To add Oracle Knowledge to an incident workspace, use the following procedure:

- 1 Add the *OKiConnect AddIn* and *LinkedAnswers* report into the OK – Incident workspace.
- 2 Click the **Design** tab on the toolbar.
- 3 Right click at the row of tabs and select **Add Tab**.
- 4 In the **Text** field on the toolbar ribbon, type “OK Knowledge” to name the new tab.



- 5 Click the **Insert Control** tab on the toolbar.
- 6 Drag the **Knowledge** icon on the toolbar and drop it into the OK Knowledge tab space.



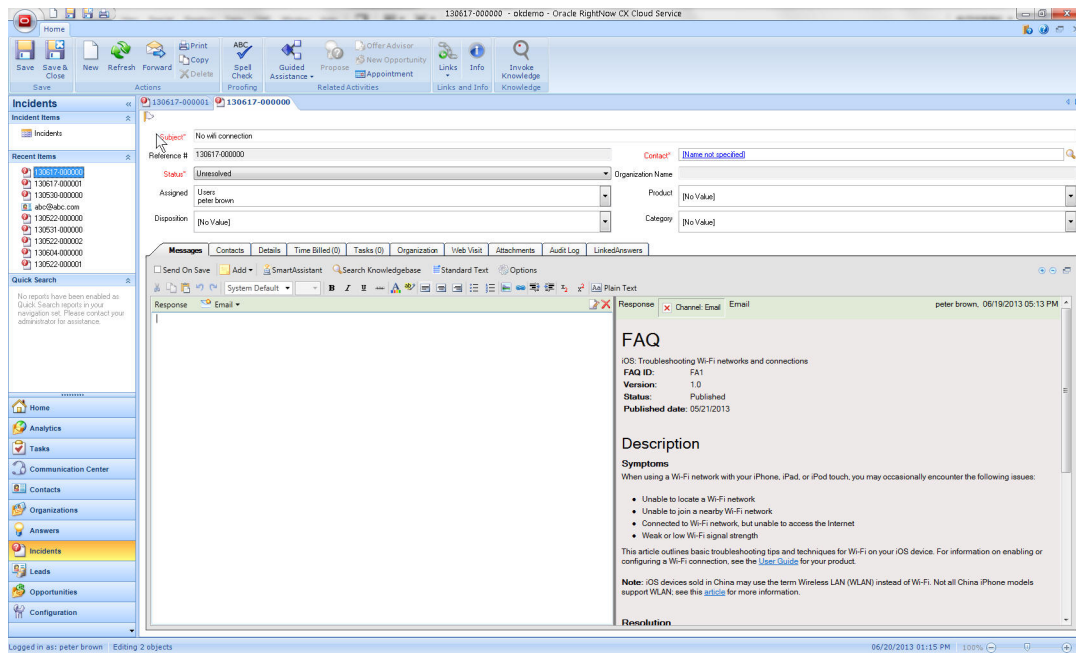


- 7 Click the **Design** tab on the toolbar and then select the **OK Knowledge** tab.
- 8 Click within the OK Knowledge tab space where the red text **Oracle Knowledge** is.  
The toolbar now includes the SearchImmediately and iConnectProperties add-in options.  
See [Configuring the Oracle Knowledge Add-In Options](#).

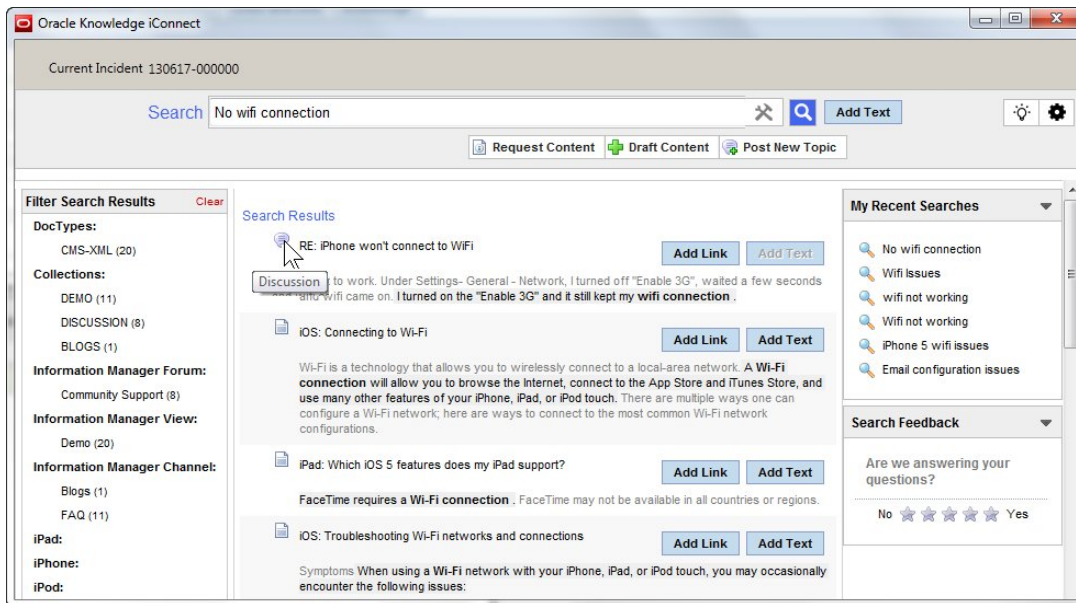
## Testing the Integration

To test the integration with Oracle Knowledge, use the following procedure:

- 1 From the Incidents navigation panel on the left, open a **Recent Item** or search for an incident that you want to add Oracle Knowledge solutions to. To open the Search dialog box, double click the **Incidents** icon in the Incident Items panel. For more information, refer to *Custom Workspaces*.  
The incident workspace opens in the Messages tab. The history of an existing incident is in the right panel.

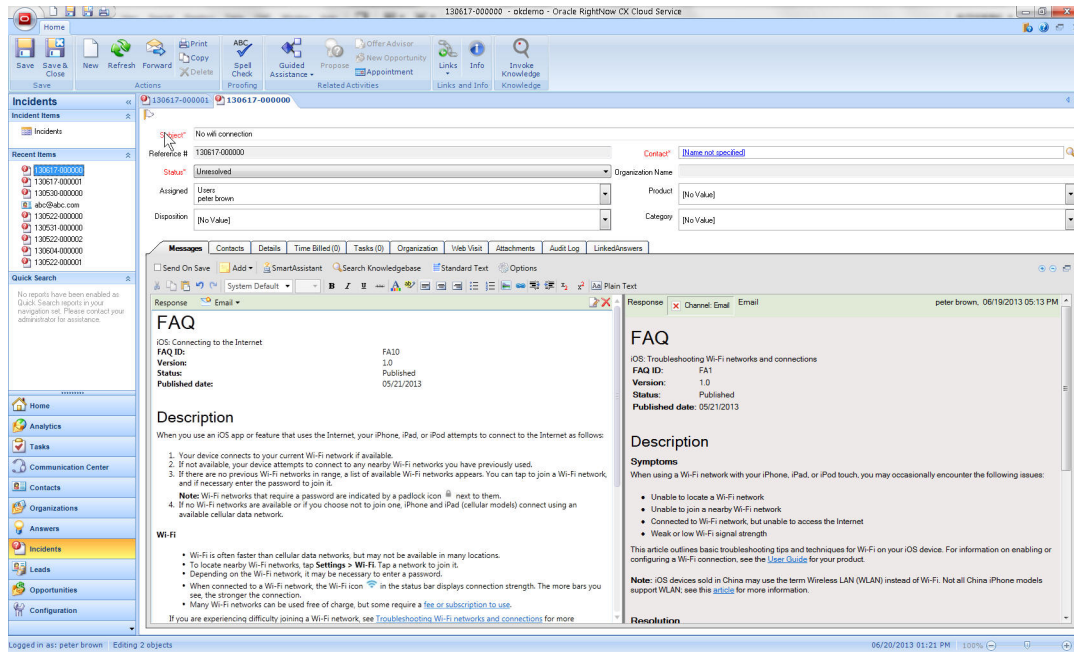


- 2 Click the **Invoke Knowledge** button on the menu.  
The Oracle Knowledge iConnect search window opens, listing possible solutions.



- 3 You can use the criteria in the Filter Search Results panel to narrow the search, or select an existing search from My Recent Searches.
- 4 Scroll through the search results to find applicable solutions.
- 5 Optionally, agents can also use the following buttons:
  - Request Content: Request additional information.

- Draft Content: Draft new content immediately.
  - Post New Topic: Choose a question and post it to forums for other agents to use.
- 6 Click **Add Link** to insert a link to the solution or **Add Text** to insert the text of the solution into the incident workspace.
- You return to the workspace. The link or text that you selected is now in the left workspace panel.



- 7 Save the incident.

## Configuring the Oracle Knowledge Add-In Options

Administrators can configure the Oracle Knowledge options for each incident workspace. The Oracle Knowledge add-in has two configuration options:

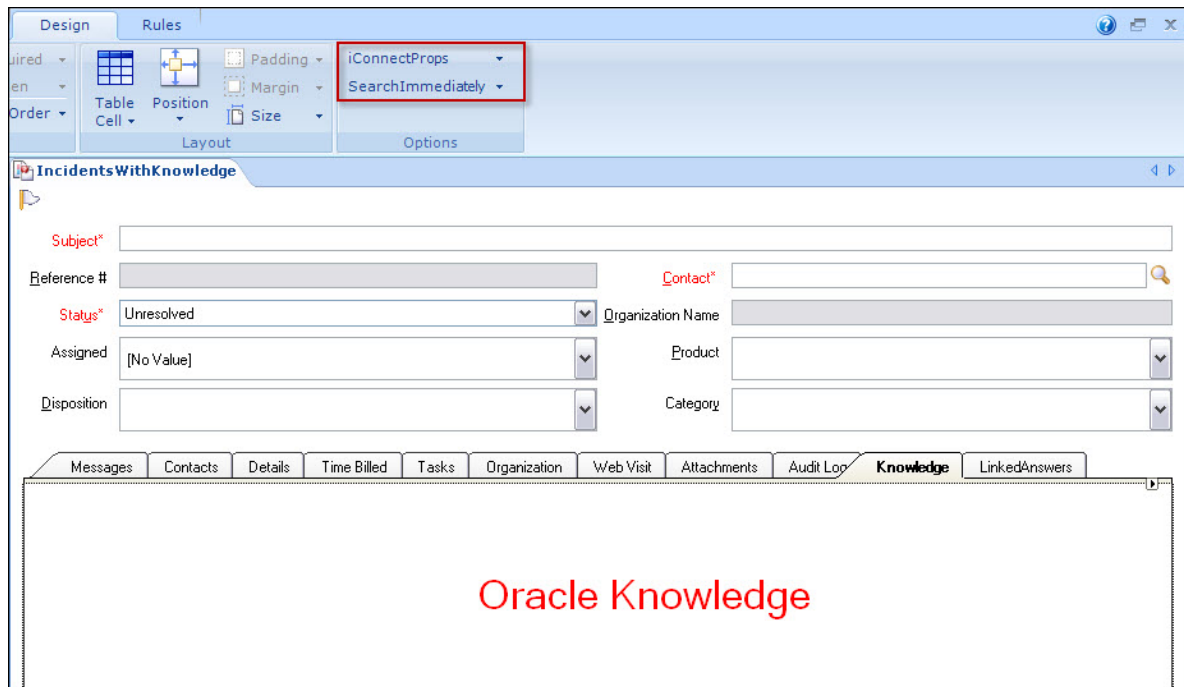
- The `SearchImmediately` option allows users to choose whether or not a search runs automatically. When a user selects the Oracle Knowledge workspace, a search runs on the text entered in the Subject field and a window pops up immediately with the results. If the search does not return results, then the window opens without search results. If the `SearchImmediately` option is not configured, the search does not run immediately and the user can add more criteria to the search.
- The `iConnectProperties` option allows users to use additional incident information, such as additional fields in the workspace, to Oracle Knowledge to filter the search results more precisely.

To configure the `SearchImmediately` and `iConnectProperties` options, use the following procedure:

- 1 In the left navigation panel, click **Configuration, Application Appearance**, and then **Workspaces / Workflows**.
- 2 Open **OK-Incident** workspace. Refer to *Custom Workspaces*.
- 3 Select the **OK Knowledge** tab and click anywhere in the tab area to select it.

- 4 Click the **Design** tab on the toolbar ribbon.

The SearchImmediately and iConnectProperties options appear on the toolbar ribbon.



- 5 Click the **SearchImmediately** option.

A text box opens.

**Note:** If you do not want a search to run automatically when you select the Oracle Knowledge workspace tab, you can disable the SearchImmediately option by clearing the text box. To enable the SearchImmediately option, type *true* in the text box.

- 6 Click the **iConnectProperties** option.

A text box opens.

- 7 The default value `cca_system=RNOW` or a blank value runs a search only on the incident subject. Optionally, you can edit the value in the text box to add incident fields that you want to include as additional search filter criteria.

To add additional filters, add parameters after `cca_system=RNOW` with the pattern `fieldname<#>=$table_name.field` where:

*Table\_name* is the name of the RightNow table (also the object in Connect Web Services for SOAP).

*Field* is the name of the field in the RightNow table (or object).

For example, if you want to include the incident threads, product, and category, you would enter the following text in the iConnectProperties option:

```
cca_system=RNOW?fieldname1=$incident.threads&fieldname2=
$incident.product&fieldname3=$incident.category
```

- 8 Optionally, you can also pass incident custom fields. For example, if you want to pass the incident custom field with an ID of 3, you would enter the following text in the iConnectProperties option:

```
cca_system=RNOW?fieldname1=$incident.c$3
```

**Tip:** To find the custom field ID, in the Custom Fields editor hover the cursor over the custom field name.

**Note:** Custom fields with Text Area or Text Field data types cannot be passed using the iConnectProperties option. Any other incident fields supported by Connect Web Services for SOAP are supported in the iConnectProperties option. For information about Connect Web Services for SOAP, click [here](#).

# Customizing Applications with Oracle Knowledge Widgets

You can use Oracle Knowledge widgets to embed knowledge within applications, using your web application development skills, such as HTML, CSS, and JavaScript.

The widget library consists of individual components that perform a particular function and expose interactions with each other to provide a high degree of flexibility and customization.

This topic explains how to use widgets to quickly build pages with functions such as Search, Results, Facets, Categories, and Channels, instantiate and configure each widget, and also set them to interact with each other.

Widgets can use other widgets. A widget may be able to share another widget with other widgets by having the other widgets access an attribute related to a widget instance. For example, the Result widget has an attribute "answerViewWidget", which is an instance of AnswerView. If a variable "X" is defined to be an instance of the Result widget, other widgets can access this widget's AnswerView instance by X.get('answerViewWidget').

For information on installing widgets, see "Adding Widgets for Oracle Knowledge".

The following table lists and describes Oracle Knowledge widgets.

**TABLE 1. Oracle Knowledge Widgets**

Uses	Widget	Description
Used by all other widgets	OKBase	Provides generic functionality of event handling and event handler overriding.
	OKUtility	Provides widgets with a way to get information from Oracle Knowledge. One instance must be shared across all widgets that require it on a page.
	LocaleUtility	Provides localization helper routines that the widgets leverage.
Usually used and implemented together to search and view Oracle Knowledge contents	Search	Provides the search form along with optional pre-search filters, such as languages and document types. Result requires a Search instance. You can configure it in a CaseDeflection, and style the same instance of Search differently.
	Result	Renders search results data after a search is performed in Search. It also reacts to filters set in Facets. It requires a Search instance. The specified AnswerList displays document details. If no AnswerList is specified, then it creates its own instance. You can configure it in a CaseDeflection and style the same instance of Result differently.
	Facets	Applies multiple filters to the set of search results. It requires a Search instance.
	AnswerView	Displays document details. Multiple widgets can share a single instance. You can configure it with the AnswerList and Result widgets.
Uses the Search, Result, and AnswerView widgets	CaseDeflection	Intercepts users' questions submitted to Support. Before submitting, this widget can run a search on the text from the submission form and provide possible answers with a Results widget and the instance of Search that the Results uses. It requires a Result instance.

TABLE 1. Oracle Knowledge Widgets

Uses	Widget	Description
Used and implemented mostly together for browsing Oracle Knowledge contents	Channel	Lists channels from Oracle Knowledge. You can set the AnswerList, Category, and CategoryExplorer widgets to react to channel changes.
	Category	Selects a single category, which can be applied as a filter to AnswerList. All categories are displayed in a tree view with the specified number of levels. When a category is selected, it becomes the top level category.
	CategoryExplorer	Expands and collapses instead of displaying all categories at all times. Otherwise, it is the same as Category.
	AnswerList	Displays a list of recent, popular, or common documents, depending on the configured type. The specified AnswerList displays document details. If no Answer is specified, it creates its own instance.

For more information on Oracle Knowledge widgets, see “Widget Details”.

## Getting Started with Widgets

The following examples demonstrate how you can apply Oracle Knowledge widgets to build an HTML page, quickly get a widget on a page, how adding one or two widgets can add complexity, make the popular answers list react to channel selection, and tie the Search/Results/Facets widgets together so there is no need to declare a listener.

- “Building an HTML Page”
- “Adding Widget Dependencies”
- “Adding Widget Events and Interaction”
- “Extending Widget Dependencies”

## Building an HTML Page

To start, build a skeleton HTML page with divs as placeholders for where each widget should be placed. You can instantiate and render widgets with YUI on the same page. Use the following bare minimum code to start.

```
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
  <script src="http://yui.yahooapis.com/3.7.2/build/yui/yui-min.js"></script>
  <link type="text/css" href="apps/infocenter/resources/css/oracleKnowledgeWidget.css"
    rel="stylesheet" media="screen">
</head>

<!--placeholder div-->
<div id="channelWidgetDiv"></div>

<script>
YUI({
  modules: {
    "oracle-knowledge/base" : {
      fullpath: "apps/infocenter/resources/js/OKWidgets/okBaseWidget.js"
    },
    "oracle-knowledge/ok-utility" : {
      fullpath: "apps/infocenter/resources/js/OKWidgets/okUtility.js"
    },
    "okLocale": {
```



```

    fullpath: "apps/infocenter/resources/js/OKWidgets/localeUtility.js"
  },
  "oracle-knowledge/channel-widget" : {
    fullpath: "apps/infocenter/resources/js/OKWidgets/channelWidget.js"
  }
},
combine: false,
filter: 'RAW'
}).use('oracle-knowledge/base', 'oracle-knowledge/ok-utility', 'okLocale', 'oracle-knowl-
edge/channel-widget', function(Y) {

  var okUtility = new Y.OracleKnowledge.OKUtility({
    baseURL : 'http://slc01qjy:8226/infocenter/', //This should point to your instance
    of InfoCenter
    widgetDir : 'apps/infocenter/resources/js/OKWidgets/',
    setLocale: 'es-ES',
    transport: 'jsonp' //alternative is 'ajax'
  });

  var channelWidget = new Y.OracleKnowledge.ChannelWidget({
    utility : okUtility,
    selectedChannel : 'FAQ' //Only specify a channel reference key here if you want
    something preselected upon rendering of the widget. Otherwise, leave as ''.
  }).render('#channelWidgetDiv');
});
</script>

```

This code example demonstrates how easy it is to get a widget on your page. The following section demonstrates how to add more widgets to build advanced functionality and interactivity.

## Adding Widget Dependencies

You can add an AnswerList and AnswerView to a page.

```

var answerViewWidget = new Y.OracleKnowledge.AnswerView({
  utility : okUtility
}).render('#answerViewDiv');
var popularAnswerList = new Y.OracleKnowledge.AnswerList({
  'answerViewWidget' : answerViewWidget, //This is the instance of AnswerView you
  just created
  'utility' : okUtility,
  type: 'popular',
  categoryReferenceKey: 'IPAD', //Only specify if you want this preselected upon
  rendering the widget
  channelReferenceKey: 'FAQ' //Only specify if you want this preselected upon
  rendering the widget
}).render('#popularAnswerListDiv');

//add the corresponding placeholder divs in your HTML skeleton
//update the 'use' section with 'oracle-knowledge/answer-view' and 'oracle-knowledge/
answer-list'
//update the 'modules' section with the corresponding file paths for both widgets

```

This renders three widgets. Click one of the items in the popular answers list, and the answerView renders the corresponding document.



## Adding Widget Events and Interaction

To make the popular answers list react to the channel selection, add a listener for 'selectChannel' on the channel, and add some code to re-render the popular answers list.

You can use regular YUI events:

```
channelWidget.on('selectChannel', function(e) {
    popularAnswerList.set('channelReferenceKey', e.key);
});
```

Or, you can override one of the widget's exposed event callback functions:

```
//add the following line to the instantiation configuration for channelWidget
//afterSelectChannel : [refreshAnswerList]
function refreshAnswerList() {
    popularAnswerList.set('channelReferenceKey',
        channelWidget.get('currentSelectedChannel'));
}
```

Now the AnswerList widget reacts to changing channels.

## Extending Widget Dependencies

You can tie the Search/Results/Facets widgets together in a similar way. Because these three functions are closely tied together, there is no need to declare a listener, such as the one you made for listening to the Channel widget's 'selectChannel' event. With these widgets, you declare them with the Facets and Results widgets taking in the Search widget, specified under the widget configurations:

```
var searchWidget = new Y.OracleKnowledge.SearchWidget({
    utility : okUtility,
    inputType : 'area'
}).render('#searchContainer');

var facetTree = new Y.OracleKnowledge.FacetTree({
    utility : okUtility,
    searchWidget : searchWidget
}).render('#facetContainer');

var resultWidget = new Y.OracleKnowledge.ResultWidget({
    utility : okUtility,
    searchWidget : searchWidget,
    answerViewWidget : answerViewWidget
}).render('#resultContainer');
```

## Widget Details

This section explains what each of the Oracle Knowledge widgets does and gives details of their attributes, properties, event flags, elements, callbacks, and object structures.

## OKBase

OKBase is the basic widget in the OK Widget Library. All other widgets extend to this one. It provides generic functionality of event handling and event handler overriding.

**TABLE 2. Configurable Attributes**

Name	Type	Values	Description
showActivityIndicator (available in Release 8.5.1.1)	Boolean	True or False	An activity indicator (spinner) is displayed while data is loading.

**TABLE 3. Event Flags**

Name	Description
loadData	_onLoadData has finished executing

**TABLE 4. Event Callbacks**

Name	Description
onRenderWidget	This is the function in which the widget gets rendered. If no value is specified, default widget rendering behavior executes. If a function is specified, the default behavior is overridden.
afterRenderWidget	If no value is specified, nothing additional executes after the widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.
onLoadData	If no value is specified, nothing additional executes after onCSSLoad. If a function is specified, the function is executed after either one of the default or overriding functions for onCSSLoad is finished.
beforeLoadData	
afterLoadData	

## LocaleUtility

The LocaleUtility widget provides localization helper routines that the widgets leverage. This has functions that return the localized text from a language resource bundle. All language resource bundles for the OK Widget Library are available at <apps/infocenter/resources/js/OKWidgets/okLocale/lang/>.

## OKUtility

OKUtility is always required. It is responsible for making calls to InfoCenter to get data for all the widgets. You can share one instance across all widgets. When a widget, such as Search, needs to get information from InfoCenter, such as run a search or get search results, it calls the appropriate function inside OKUtility.

**TABLE 5. Configurable Attributes**

Name	Type	Values	Description
baseURL (required)	String	"http:// infocenterhost:infocenterport/ <infocenterappname>/"	The URL of an Infocenter instance.

TABLE 5. Configurable Attributes

Name	Type	Values	Description
widgetDir (required)	String	"http:// infocenterhost:infocenterport/ <infocenterappname>/apps/ infocenter/resources/js/ OKWidgets/)"	The path to OK Widget Library.
css	String	"http:// infocenterhost:infocenterport/ <infocenterappname>/ directory/CSSFilename"	If the default CSS will not be used, set the path and name (without the suffixes '_ltr' and 'rtl' and file extension) of the CSS file to be used by the OK Widget Library.
setLocale	String	<Locale code>	If the default Infocenter locale must be overridden, set the language code.

TABLE 6. Properties

Name	Values	Type	Description
availLang	List<Language>	JSON Array	Lists available languages configured for InfoCenter.
defaultLocale	<Locale code>	String	Determines which language resource bundle should be used.
languageDirection	"ltr" or "rtl"	"String"	Determines which CSS version (LTR or RTL) should be used.
previousLanguageDirection	<Locale code>	String	Holds the previous value of 'languageDirection' and helps determine whether or not the CSS file switch is needed.
admin (available in Release 8.5.1.1)	True or False	Boolean	Indicates whether or not the signed-in user has administration roles in Infocenter.

TABLE 7. Event Flags

Name	Description
cssLoaded	_onCSSLoad has finished executing

TABLE 8. Event Callbacks

Name	Description
onCSSLoad	This function is triggered when a CSS file is loaded successfully. The default behavior first fires the 'cssLoaded' event. When finished, it sets the 'cssLoaded' event flag to true. If a function is specified, the default behavior is overridden.
beforeCSSLoad	If no value is specified, nothing additional executes before onCSSLoad. If a function is specified, the function is executed before either one of the default or overriding functions for onCSSLoad is finished.
afterCSSLoad	If no value is specified, nothing additional executes after onCSSLoad. If a function is specified, the function is executed after either one of the default or overriding functions for onCSSLoad is finished.

## AnswerView

AnswerView displays the details of a document. Multiple widgets can share a single instance. You can configure AnswerView with AnswerList and Result widgets.

**TABLE 9. Configurable Attributes**

Name	Type	Values	Description
utility (required)	OKUtility	<OKUtility instance>	Provides this widget with the ability to grab data from InfoCenter.
deflection	Boolean	True or False	Indicates whether or not this widget is used for case deflection.
deflectionWidget	CaseDeflection	<CaseDeflection instance>	Provides the case deflection widget instance associated with this widget, if attribute 'deflection' is set to True,.
closeButtonTextKey	String	<Resource key>	Resource key that holds the text to display on the Close button of this widget.  If not set, the default value is 'SearchTips_close'.

**TABLE 10. Exposed DOM Elements**

Name	Tag	CSS Suffix	Description
iFrameContainer	DIV	iFrameContainer	The div that contains 'iFrameParent'.
iFrameParent	DIV	iFrameParent	The div that contains 'answerFrame' and 'closeButton'.
answerFrame	IFRAME	answerFrame	An iFrame that references the document's URL.
closeButton	BUTTON	closeButton	The button that hides the widget.

**TABLE 11. Event Flags**

Name	Description
_eventPrevented	An event has been prevented from being triggered. Respective _before- _on- _after-events not executed.
show	_onShowAnswerView has finished executing
dismiss	_onDismissAnswerView has finished executing

**TABLE 12. Event Callbacks**

Name	Description
onRenderWidget	The function in which the widget gets rendered. If no value is specified, default widget rendering behavior executes. Any specified function overrides the default behavior.
afterRenderWidget	If no value is specified, nothing additional executes after the widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.
onShowAnswerView	This function is triggered when the 'show' event is fired with 'uri' passed in the event firing. The default behavior unhides the widget and updates the uri of the iframe. When finished, it sets the 'show' event flag to true. Any specified function overrides the default behavior.
beforeShowAnswerView	If no value is specified, nothing additional executes before onShowAnswerView. If a function is specified, the function is executed before either one of the default or overriding functions for onShowAnswerView is finished.

TABLE 12. Event Callbacks

Name	Description
afterShowAnswerView	If no value is specified, nothing additional executes after onShowAnswerView. If a function is specified, the function is executed after either one of the default or overriding functions for onShowAnswerView is finished.
onDismissAnswerView	Clicking the Close button triggers this function. The default behavior first fires the 'dismiss' event, which hides the widget. When finished, it sets the 'dismiss' event flag to true. Any specified function overrides the default behavior.
beforeDismissAnswerView	If no value is specified, nothing additional executes before onDismissAnswerView. If a function is specified, the function is executed before either one of the default or overriding functions for onDismissAnswerView is finished.
afterDismissAnswerView	If no value is specified, nothing additional executes after onDismissAnswerView. If a function is specified, the function is executed after either one of the default or overriding functions for onDismissAnswerView is finished.
deflectionShowAnswerView	If set, this is the additional function that is executed with 'onShowAnswerView' if the Answer View widget instance is used for case deflection.
deflectionHideAnswerView	If set, this is the additional function that is executed with 'onDismissAnswerView' if the Answer View widget instance is used for case deflection.

TABLE 13. Properties

Name	Values	Type	Description
blankUrl	about:blank	String	Default source value for iFrame
strings	<Localized texts>	Object	Localized texts in the current language

## AnswerList

Depending on the configured type, this widget displays a list of 'recent', 'popular', or 'common' documents. Document details are displayed in the specified AnswerView. If no AnswerView widget is specified, it creates its own instance.

TABLE 14. Configurable Attributes

Name	Type	Values	Description
utility (required)	OKUtility	<OKUtility instance>	Provides this widget with the ability to grab data from InfoCenter.
answerViewWidget	AnswerView	<AnswerView widget instance>	Displays the contents of an answer with this instance of AnswerView widget when an answer is clicked.
type	String	recent	Displays most recent articles.
		popular	Displays most popular articles.
		common	Displays most common articles.
maxSize	int	<int>	Fetches answers up to the specified number.
pageSize	int	<int>	Answers up to the number specified will be displayed per page.
channelReferenceKey	String	<empty string>	Fetches answers without filtering by channel.
		<ChannelReferenceKey>	Fetches answers only from specified channel.

TABLE 14. Configurable Attributes

Name	Type	Values	Description
categoryReferenceKey	String	<empty string>	Fetches answer without filtering by category.
		<CategoryReferenceKey>	Fetches answers only from specified category.
sortDirection	String	ascending	Lists answers from lowest value first.
		descending	Lists answers from highest value first (default).
sortBy	String	documentid	Sorts list by the DocID value.
		indexmasteridentifiers	Sorts list by the master identifier value.
		views	Sorts list by number of views.
		displaystartdate	Sorts list by value of the first posted date (default).
pageNumber	int	<int>	Opens list to the specified page number.
listPageNumbers	int	<int>	Displays maximum number of page numbers in the pagination sections before an ellipses is needed (default = 7).
descendingImageName	String	<Image Filename>	Filename of the image to use as descending sort icon. Default value is "navigate_close2.png".  This file must be available in the directory <i>apps/infocenter/resources/images/</i> .
ascendingImageName	String	<Image Filename>	Filename of the image to use as ascending sort icon. Default value is "navigate_open2.png".  This file must be available in the directory <i>apps/infocenter/resources/images/</i> .

TABLE 15. Properties

Name	Values	Type	Description
answerListData	Data necessary to render the answer list	JSON	Populates answerListData when the widget is rendered. Makes a call to a function inside OKUtility to retrieve data from InfoCenter.
pageText1		String	Holds the 'Next' and 'Previous' text for the top pagination bar.
pageText2		String	Holds the 'Next' and 'Previous' text for the bottom pagination bar.
draft	'Y'	String	Value for the option of the drop-down filter to show all draft and published articles.
	'N'	String	Value for the option of the drop-down filter to show only the published articles.
strings	<Localized texts>	Object	Localized texts in the current language.

TABLE 16. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
answerListBlock	TBODY	'type'+AnswerListBlock	Body of the outermost table.
answerListHeader	TR	<type>+'AnswerListHeaderRow	Holds 'answerListHeaderTitle' for the title displayed in the top bar.

TABLE 16. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
answerListPaginationTop	TD	paginationtop	Top pagination bar.
answerListPaginationBottom	TD	paginationbottom	Bottom pagination bar.
answerListTitle	TR	<type>+'AnswerListTitle'	Inside the table within 'answerListBody'.
answerListBody	TD	<type>+'AnswerListBody'	Holds the inner table and excludes the pagination bars.
answerListHeaderTitle	TD	<type>+'AnswerListHeader'	Inside 'answerListHeader' to display the title in the top bar.
filterContainer	TD	filter	Holds the drop-down list for the Filter options.
filterTitle	Label		Label for the drop-down list.
filterSelect	Select		Drop-down list for the Filter options.
allOption	Option		Option to show all draft and published articles.
publishedOnlyOption	Option		Option to show only the published articles.

TABLE 17. Event Flags

Name	Description
_eventPrevented	An event has been prevented from being triggered. Respective _before- _on- _after-events not executed.
sort	_onSort has finished executing.
changePage	_onChangePage has finished executing.
clickThrough	_onClickThrough has finished executing.
filter	_onFilter has finished executing.

TABLE 18. Event Callbacks

Name	Description
onRenderWidget	The function in which the widget gets rendered. If no value is specified, default widget rendering behavior executes. Any specified function overrides the default behavior.
afterRenderWidget	If no value is specified, nothing additional executes after the widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.
onSort	Clicking the sort icon or one of the header titles triggers this function. The default behavior first fires the 'sort' event and broadcasts the sort key in the event data. The list is re-rendered. When finished, it sets the 'sort' event flag to true. Any specified function overrides the default behavior.
beforeSort	If no value is specified, nothing additional executes before onSort. If a function is specified, the function is executed before either one of the default or overriding functions for onSort is finished.
afterSort	If no value is specified, nothing additional executes after onSort. If a function is specified, the function is executed after either one of the default or overriding functions for onSort is finished.
onChangePage	Clicking a 'previous', 'next', or other page number link triggers this function. The default behavior first fires the 'changePage' event and broadcasts the new page number in the event data. Then, the widget is re-rendered. When finished, it sets the 'changePage' event flag to true. Any specified function overrides the default behavior.
beforeChangePage	If no value is specified, nothing additional executes before onChangePage. If a function is specified, the function is executed before either one of the default or overriding functions for onChangePage is finished.

TABLE 18. Event Callbacks

Name	Description
afterChangePage	If no value is specified, nothing additional executes after onChangePage. If a function is specified, the function is executed after either one of the default or overriding functions for onChangePage is finished.
onClickThrough	Clicking a document id link or an answer title link triggers this function. The default behavior first fires the 'clickThrough' event and broadcasts the 'uri', or both 'uri' and 'docid', respectively, in the event data. Then, the 'show' event of the AnswerView widget fires with the 'uri' data. When finished, it sets the 'clickThrough' event flag to true. Any specified function overrides the default behavior.
beforeClickThrough	If no value is specified, nothing additional executes before onClickThrough. If a function is specified, the function is executed before either one of the default or overriding functions for onClickThrough is finished.
afterClickThrough	If no value is specified, nothing additional executes after onClickThrough. If a function is specified, the function is executed after either one of the default or overriding functions for onClickThrough is finished.
onFilter	This function is triggered when one of the options of the drop-down list filter is selected. The default behavior first fires the 'filter' event and broadcasts the filter key in the event data. The list is re-rendered. When finished, it sets the 'filter' event flag to true. If a function is specified, the default behavior is overridden.
beforeFilter	If no value is specified, nothing additional executes before onFilter. If a function is specified, the function is executed before either one of the default or overriding functions for onFilter is finished.
afterFilter	If no value is specified, nothing additional executes after onFilter. If a function is specified, the function is executed after either one of the default or overriding functions for onFilter is finished.

## Category

Category gives the ability to select a single category, which can be applied as a filter to AnswerList. All categories are displayed in a tree view with the specified number of levels. When you select a category, it becomes the top-level category.

TABLE 19. Configurable Attributes

Name	Type	Values	Description
okUtility (required)	OKUtility	<OKUtility instance>	Provides this widget with the ability to grab data from InfoCenter.
topLevelCategoryReferenceKey	String	undefined	Displays the categories tree from the root.
		<CategoryReferenceKey>	Displays the categories tree from this category.
renderTopLevelCategory	Boolean	true	The category with reference key 'topLevelCategoryReferenceKey' is rendered in the tree.
		false	The category with reference key 'topLevelCategoryReferenceKey' is not rendered in the tree.
depth	int	undefined	Renders as many levels of the category tree as possible.
		<int>	Renders the tree with the specified number of levels.



TABLE 20. Properties

Name	Values	Type	Description
categoryData	Data necessary to render the category tree	JSON	categoryData is populated when the widget is rendered. A call is made to a function inside OKUtility to retrieve data from InfoCenter.
parentReferenceKey	<CategoryReferenceKey>	String	Reference key of the parent category of current tree.
strings	<Localized texts>	Object	Localized texts in the current language.

TABLE 21. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
contentDiv	DIV	contentDiv	The div just within the contentBox.
topUL	UL	topUL	A channel that is not selected.
titleDiv	DIV	titleDiv	The title of the widget.
backButton	INPUT type=button	upOneCategory	Button to change topLevelCategoryReferenceKey to the parent of current tree.

TABLE 22. Event Flags

Name	Description
_eventPrevented	An event has been prevented from being triggered. Respective _before- _on- _after-events not executed.
selectCategory	_onSelectCategory has finished executing.
upOneCategory	_onUpOneCategory has finished executing.

TABLE 23. Event Callbacks

Name	Description
onRenderWidget	This is the function in which the widget gets rendered. If no value is specified, default widget rendering behavior executes. Any specified function overrides the default behavior.
afterRenderWidget	If no value is specified, nothing additional executes after the widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.
onSelectCategory	Clicking a category link triggers this function. The default behavior first fires the 'selectCategory' event and broadcasts the category's reference key in the event data. The topLevelCategoryReferenceKey is changed to the clicked category, and the parentReferenceKey is updated accordingly. The entire tree is rendered according to the configurations specified. Upon finishing, it sets the 'selectCategory' event flag to true. Any specified function overrides the default behavior.
beforeSelectCategory	If no value is specified, nothing additional executes before onSelectCategory. If a function is specified, the function is executed before either one of the default or overriding functions for onSelectCategory is finished.
afterSelectCategory	If no value is specified, nothing additional executes after onSelectCategory. If a function is specified, the function is executed after either one of the default or overriding functions for onSelectCategory is finished.
onUpOneCategory	Clicking the Up On Category button triggers this function. The default behavior first fires the 'upOneCategory' event and broadcasts the parent category's reference key in the event data. The topLevelCategoryReferenceKey is changed to the clicked category, and the parentReferenceKey is updated accordingly. The entire tree is rendered according to the configurations specified. When finished, it sets the 'upOneCategory' event flag to true. Any specified function overrides the default behavior.

TABLE 23. Event Callbacks

Name	Description
beforeUpOneCategory	If no value is specified, nothing additional executes before onUpOneCategory. If a function is specified, the function is executed before either one of the default or overriding functions for onUpOneCategory is finished.
afterUpOneCategory	If no value is specified, nothing additional executes after onUpOneCategory. If a function is specified, the function is executed after either one of the default or overriding functions for onUpOneCategory is finished. The specified AnswerListWidget displays document details.

## CategoryExplorer

CategoryExplorer is the same as Category, but the rendering is different. Instead of displaying all categories at all times, it has expand and collapse capabilities.

TABLE 24. Configurable Attributes

Name	Type	Values	Description
okUtility (required)	OKUtility	<OKUtility instance>	Provides this widget with the ability to grab data from InfoCenter.
topLevelCategoryReferenceKey	String	undefined	Starts displaying the categories tree from the root.
		<CategoryReferenceKey>	Starts displaying the categories tree from this category.
renderTopLevelCategory	Boolean	true	The category with reference key 'topLevelCategoryReferenceKey' is rendered in the tree.
		false	The category with reference key 'topLevelCategoryReferenceKey' is not rendered in the tree.
depth	int	undefined	Renders as many levels of the category tree as possible.
		<int>	Renders the tree with the specified number of levels.

TABLE 25. Properties

Name	Values	Type	Description
categoryData	Data necessary to render the category tree	JSON	categoryData is populated when the widget is rendered. A call is made to a function inside OKUtility to retrieve data from InfoCenter.
strings	<Localized texts>	Object	Localized texts in the current language.

TABLE 26. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
contentDiv	DIV	contentDiv	The div just within the contentBox.
topUL	UL	topUL	A channel that is not selected.
titleDiv	DIV	titleDiv	The title of the widget
currentSelectedCategory	A	linkSelected	The selected channel
previousSelectedCategory	A	link	The channel that was last previously selected

TABLE 27. Event Flags

Name	Description
_eventPrevented	An event has been prevented from being triggered. Respective _before- _on- _after-events not executed.
selectCategory	_onSelectCategory has finished executing.
expandCategory	_onExpandCategory has finished executing.
collapseCategory	_onCollapseCategory has finished executing.

TABLE 28. Event Callbacks

Name	Description
onRenderWidget	This is the function in which the widget gets rendered. If no value is specified, default widget rendering behavior executes. Any specified function overrides the default behavior.
afterRenderWidget	If no value is specified, nothing additional executes after the widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.
onSelectCategory	Clicking a category link triggers this function. The default behavior first fires the 'selectCategory' event and broadcasts the category's reference key in the event data. Then, it deselects the previously selected category and selects the category that was clicked, and the previousSelectedCategory and currentSelectedCategory are updated, respectively. Upon finishing, it sets the 'selectCategory' event flag to true. Any specified function overrides the default behavior.
beforeSelectCategory	If no value is specified, nothing additional executes before onSelectCategory. If a function is specified, the function is executed before either one of the default or overriding functions for onSelectCategory is finished.
afterSelectCategory	If no value is specified, nothing additional executes after onSelectCategory. If a function is specified, the function is executed after either one of the default or overriding functions for onSelectCategory is finished.
onExpandCategory	Clicking an Expand button triggers this function. The default behavior first fires the 'expandCategory' event and broadcasts the category's reference key in the event data. Then, it unhides its children and hides its siblings and their ancestors. Upon finishing, it sets the 'expandCategory' event flag to true. Any specified function overrides the default behavior.
beforeExpandCategory	If no value is specified, nothing additional executes before onExpandCategory. If a function is specified, the function is executed before either one of the default or overriding functions for onExpandCategory is finished.
afterExpandCategory	If no value is specified, nothing additional executes after onExpandCategory. If a function is specified, the function is executed after either one of the default or overriding functions for onExpandCategory is finished.
onCollapseCategory	Clicking a Collapse button triggers this function. The default behavior first fires the 'collapseCategory' event and broadcasts the category's reference key in the event data. Then, it hides its children. Upon finishing, it sets the 'collapseCategory' event flag to true. Any specified function overrides the default behavior.

TABLE 28. Event Callbacks

Name	Description
beforeCollapseCategory	If no value is specified, nothing additional executes before onCollapseCategory. If a function is specified, the function is executed before either one of the default or overriding functions for onCollapseCategory is finished.
afterCollapseCategory	If no value is specified, nothing additional executes after onCollapseCategory. If a function is specified, the function is executed after either one of the default or overriding functions for onCollapseCategory is finished.

## Channel

You can set up a list of channels from InfoCenter so that AnswerList, Category, and CategoryExplorer widgets react to channel changes.

TABLE 29. Configurable Attributes

Name	Values	Type	Description
utility (required)	<OKUtility instance>	OKUtility	Provides this widget with the ability to grab data from InfoCenter.
currentSelectedChannel	<empty string>	String	No channel is selected when the widget is first rendered.
	<ChannelReferenceKey>		Specified channel is selected when the widget is first rendered.

TABLE 30. Properties

Name	Values	Type	Description
channelData	Data necessary to render the channel list	JSON	channelData is populated when widget is rendered. A call is made to a function inside OKUtility to retrieve data from InfoCenter.
previousSelectedChannel	<ChannelReferenceKey>	String	Holds the reference key of the previously selected channel.
strings	<Localized texts>	Object	Localized texts in the current language.

TABLE 31. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
channelList	UL	channelList	Reference key of currently selected channel.

TABLE 32. Event Flags

Name	Description
_eventPrevented	An event has been prevented from being triggered. Respective _before- _on- _after- events not executed.
selectChannel	_onSelectChannel has finished executing.

TABLE 33. Event Callbacks

Name	Description
onRenderWidget	The function in which the widget gets rendered. If no value is specified, default widget rendering behavior executes. Any specified function overrides the default behavior.
afterRenderWidget	If no value is specified, nothing additional executes after widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.

TABLE 33. Event Callbacks

Name	Description
onSelectChannel	Clicking a channel triggers this function. The default behavior first fires the 'selectChannel' event and broadcasts the channel's reference key in the event data. Then, it deselects the previously selected channel and selects the channel that was clicked, and the previousSelectedChannel and currentSelectedChannel are updated, respectively. When finished, it sets the 'selectChannel' event flag to true. Any specified function overrides the default behavior.
beforeSelectChannel	If no value is specified, nothing additional executes before onSelectChannel. If a function is specified, the function is executed before either one of the default or overriding functions for onSelectChannel is finished.
afterSelectChannel	If no value is specified, nothing additional executes after onSelectChannel. If a function is specified, the function is executed after either one of the default or overriding functions for onSelectChannel is finished.

## Facets

The Facets widget has the ability to apply multiple filters to the set of search results. It requires a Search widget instance.

TABLE 34. Properties

Name	Values	Type	Description
strings	<Localized texts>	Object	Localized texts in the current language.

TABLE 35. Configurable Attributes

Name	Values	Type	Description
searchWidget (required)	<Search widget instance>	Search	Provides this widget with the ability to receive the latest facets set.

TABLE 36. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
facetsList	DIV	facetsList	Div that contains the UL of facets.
clearFacetsLink	A	clearFacets	Clears selected facets.

TABLE 37. Event Flags

Name	Description
_eventPrevented	An event has been prevented from being triggered. Respective _before- _on- _after-events not executed.
facetClick	_onFacetClick has finished executing
clearFacets	_onClearFacets has finished executing

TABLE 38. Event Callbacks

Name	Description
onRenderWidget	This is the function in which the widget gets rendered. If no value is specified, default widget rendering behavior executes. Any specified function overrides the default behavior.
afterRenderWidget	If no value is specified, nothing additional executes after the widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.
onFacetClick	Clicking a facet link triggers this function. The default behavior first fires the 'facetClick' and broadcasts the facet's reference key in the event data. If the facet was not selected before, it is now selected. If the facet was selected before, it is now deselected. When finished, it sets the 'facetClick' event flag to true. Any specified function overrides the default behavior.
beforeFacetClick	If no value is specified, nothing additional executes before onFacetClick. If a function is specified, the function is executed before either one of the default or overriding functions for onFacetClick is finished.
afterFacetClick	If no value is specified, nothing additional executes after onFacetClick. If a function is specified, the function is executed after either one of the default or overriding functions for onFacetClick is finished.
onClearFacets	Clicking the Clear link triggers this function. The default behavior first fires the 'clearFacets' event. All facets are deselected. When finished, it sets the 'clearFacets' event flag to true. Any specified function overrides the default behavior.
beforeClearFacets	If no value is specified, nothing additional executes before onFacetClick. If a function is specified, the function is executed before either one of the default or overriding functions for onFacetClick is finished.
afterClearFacets	If no value is specified, nothing additional executes after onFacetClick. If a function is specified, the function is executed after either one of the default or overriding functions for onFacetClick is finished.

## Search

Search provides the search form along with optional pre-search filters, such as languages and document types. The Result widget requires a Search widget instance. It can be configured in a CaseDeflection, and the same instance of the Search widget can be used to style differently.

TABLE 39. Configurable Attributes

Name	Values	Type	Description
utility (required)	<OKUtility instance>	OKUtility	Provides this widget with the ability to receive the latest results set.
displayLanguages	true	Boolean	Default value. Language selection is rendered.
	false	Boolean	Language selection is not rendered.
displaySearchTips	true	Boolean	Default value. Search tips option is rendered.
	false	Boolean	Default value. Search tips option is not rendered.

TABLE 39. Configurable Attributes

Name	Values	Type	Description
displayContentTypes	true	Boolean	Default value. Dropdown box for content types pre-search filter is rendered.
	false		Dropdown box for content types pre-search filter is not rendered.
displayStartOverLink	true	Boolean	Default value. Link for starting over a search is rendered.
	false		Link for starting over a search is not rendered.
deflection	true	Boolean	Appends 'deflection' to all class names.
	false		Does not append 'deflection' to any class name.
inputType	field	String	Default value. Input box is a text field element.
	area		Input box is a text area element.
shortPlaceholderText	true	Boolean	Reference key 'AskBox_askCaption' is used for the localized placeholder text of the input field. This reference key has a value of 'Search'.
	false		Default value. Reference key 'AskBox_example' is used for the localized placeholder text of the input field. This reference key has a value of 'Enter questions, specific terms, or Document IDs.'
imageButtonURL	<directory/ imageFilename>	String	The search button becomes of image type when this attribute is set.
contentTypeVisible	true	Boolean	Default value. The drop down list for filtering search by content type is rendered.
	False	Boolean	The drop down list for filtering search by content type is not rendered.

TABLE 40. Properties

Name	Type	Values	Description
searchText	String	<input text>	Value of search input field.
searchID	String	<search id>	A random ID generated by search when a new search is performed.
type	String	'search'	Default value. Default type of search.
		'empty'	Used for new search
subfacetID	String	<Facet reference key>	Reference key of the selected facet.

TABLE 40. Properties

Name	Type	Values	Description
selLocales	String	<local code, locale code>	List of locale codes (separated with comma) that represent the selected languages.
searchResult	JSON	<results>	searchResult is populated when a search is made. A call is made to a function inside OKUtility to retrieve data from InfoCenter.
facetData	JSON	<facets>	facetData is populated when a search is made. A call is made to a function inside OKUtility to retrieve data from InfoCenter.
validate	String	'y'	Default value. Validates the search results against the database.
restrict	String	'IM' or 'IM_DISCUSSION' or 'IM_CHANNEL'	InQuira search looks only for InfoManager documents, Discussions, or Non-Discussions. The three valid values are IM, IM_DISCUSSION, and IM_CHANNEL, respectively. If this parameter is not specified, no restrictions are applied.
answerid	String	<answer id>	A unique value for an answer. Used when calling the request to "open" an external document. Returned for each answer.
relatedids	String	<related id>	A unique value for an answers related ids. Used when calling the request to find "similar" answers.
baseurl	String	"http:// infocenterhost:infocenterport /infocenterappname/"	Value of the baseUrl attribute of OKUtility instance related to this widget.
searchTypesSelectName	String	'restrict'	
searchTipsCloseImg	String	<directory/imageFilename>	Image used for closing the Search Tips overlay
strings	Object	<Localized texts>	Localized texts in the current language.

TABLE 41. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
searchForm	FORM	searchForm	Form containing the input, button, and other elements (search tips, content type, languages, start over link) related to a search.
searchField	FIELD	searchField	Contains text to be submitted in search query.
searchButton	INPUT type= submit	searchButton	Submits search query.
searchTipsShowButton	INPUT type= button	searchTips	Button that triggers the display of Search Tips overlay.



TABLE 41. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
searchTipsOverlayContainer	YUI Overlay	Not Applicable	YUI Overlay that contains the Search Tips content.
searchTipsDragElement	HTML element		Default value is 'searchTipHeader'. Element that makes the Search Tips overlay draggable.
searchTipHeader	H4		This is the 'searchTipHeader'.
languageContainer	DIV	languageContainer	Contains list of checkbox selections for filtering search by languages.
startOverLink	A	startOverLink	Triggers a new searchID to be generated.
languagesLink	A	languagesLink	Inside 'searchBoxDivLower'.
searchBoxDivOuter	DIV	searchBoxDivOuter	Contains 'searchBoxDivLower' and 'upperDiv'.
allOption	OPTION	Not Applicable	Value is ". Searches for everything.
discussionOption	OPTION		Value is 'IM_DISCUSSION'. Searches only in discussion.
channelOption	OPTION		Value is 'IM_CHANNEL'. Searches only in selected channel.
searchBoxDivLower	DIV	lowerDiv	Contains 'contentTypeSelectionDiv', 'startOverLink', and 'languagesLink'.
searchFieldset	FIELDSET		Fieldset that groups elements of within the search form.

TABLE 42. Event Flags

Name	Description
_eventPrevented	An event has been prevented from being triggered. Respective _before- _on- _after-events not executed.
search	_onSearch has finished executing.
startOverSearch	_onStartOverSearch has finished executing.
newsearch	_onNewsearch has finished executing.
startover	_onStartover has finished executing.
languageVisible	showLanguagesFn sets this to True when it has finished executing . _hideLanguagesFn sets this to False once it has finished executing.
searchTipsVisible	_onSearchTipsVisible has finished executing.

TABLE 43. Event Callbacks

Name	Description
onRenderWidget	The function in which the widget gets rendered. If no value is specified, default widget rendering behavior executes. Any specified function overrides the default behavior.
afterRenderWidget	If no value is specified, nothing additional executes after the widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.

TABLE 43. Event Callbacks

Name	Description
onSearch	This function is triggered when 'searchButton' is clicked. The default behavior first fires the 'search' event. Search parameters (type, subfacetID, searched, searchText, selLocales, startover, newsearch) are passed to the OKUtility' inQuiraSearch function and 'searchResult' is populated. When finished, it sets the 'search' event flag to true. If a function is specified, the default behavior is overridden.
beforeSearch	If no value is specified, nothing additional executes before onSearch. If a function is specified, the function is executed before either one of the default or overriding functions for onSearch is finished.
afterSearchSuccess	If no value is specified, nothing additional executes after a successful onSearch. If a function is specified, the function is executed after either one of the default or overriding functions for successful onSearch is finished.
afterSearchFailure	If no value is specified, nothing additional executes after a failed onSearch. If a function is specified, the function is executed after either one of the default or overriding functions for failed onSearch is finished.
onStartOver	This function is triggered when the 'startOverLink' is clicked. The default behavior first fires the 'startOverSearch' event. Search parameters are reset and passed to OKUtility' inQuiraSearch function. This also resets the value of 'searchResult' and provides a new value for 'searchID'. When finished, it sets the 'startOverSearch' event flag to true. If a function is specified, the default behavior is overridden.
beforeStartOver	If no value is specified, nothing additional executes before onStartOver. If a function is specified, the function is executed before either one of the default or overriding functions for onStartOver is finished.
afterStartOverSuccess	If no value is specified, nothing additional executes after a successful onStartOver. If a function is specified, the function is executed after either one of the default or overriding functions for successful onStartOver is finished.
afterStartOverFailure	If no value is specified, nothing additional executes after a failed onStartOver. If a function is specified, the function is executed after either one of the default or overriding functions for failed onStartOver is finished.
onShowLanguages	This function is triggered when the 'languagesLink' link is clicked if 'languageVisible' is false. The default behavior first fires the 'showLanguages' event. 'languageContainer' is displayed. When finished, it sets the 'languageVisible' event flag to true. If a function is specified, the default behavior is overridden.
beforeShowLanguages	If no value is specified, nothing additional executes before onShowLanguages. If a function is specified, the function is executed before either one of the default or overriding functions for onShowLanguages is finished.
afterShowLanguages	If no value is specified, nothing additional executes after onShowLanguages. If a function is specified, the function is executed after either one of the default or overriding functions for onShowLanguages is finished.
onHideLanguages	This function is triggered when the 'languagesLink' is clicked if 'languageVisible' is true. The default behavior first fires the 'hideLanguages' event. 'languageContainer' is hidden. When finishing, it sets the 'languageVisible' event flag to false. If a function is specified, the default behavior is overridden.
beforeHideLanguages	If no value is specified, nothing additional executes before onHideLanguages. If a function is specified, the function is executed before either one of the default or overriding functions for onHideLanguages is finished.

TABLE 43. Event Callbacks

Name	Description
afterHideLanguages	If no value is specified, nothing additional executes after onHideLanguages. If a function is specified, the function is executed after either one of the default or overriding functions for onHideLanguages is finished.
onShowSearchTips	This function is triggered when the 'searchTipsShowButton' is clicked if 'searchTipsVisible' is false. The default behavior first fires the 'showSearchTips' event. 'searchTipsOverlayContainer' is displayed. When finished, it sets the 'searchTipsVisible' event flag to true. If a function is specified, the default behavior is overridden.
beforeShowSearchTips	If no value is specified, nothing additional executes before onShowSearchTips. If a function is specified, the function is executed before either one of the default or overriding functions for onShowSearchTips is finished.
afterShowSearchTips	If no value is specified, nothing additional executes after onShowSearchTips. If a function is specified, the function is executed after either one of the default or overriding functions for onShowSearchTips is finished.
onHideSearchTips	This function is triggered when the 'searchTipsShowButton' is clicked if 'searchTipsVisible' is true. The default behavior first fires the 'hideSearchTips' event. 'searchTipsOverlayContainer' is hidden. When finished, it sets the 'searchTipsVisible' event flag to false. If a function is specified, the default behavior is overridden.
beforeHideSearchTips	If no value is specified, nothing additional executes before onHideSearchTips. If a function is specified, the function is executed before either one of the default or overriding functions for onHideSearchTips is finished.
afterHideSearchTips	If no value is specified, nothing additional executes after onHideSearchTips. If a function is specified, the function is executed after either one of the default or overriding functions for onHideSearchTips is finished.

## Result

Result renders search results data after a search is performed in the Search widget. It also reacts to filters set in Facets and requires a Search instance. The specified AnswerList displays document details. If no AnswerList is specified, it creates its own instance. It can be configured in a CaseDeflection, and the same instance of Result can be styled differently.

TABLE 44. Configurable Attributes

Name	Values	Type	Description
searchWidget (required)	<Search widget instance>	Search	Provides this widget with the ability to receive the latest results set.
answerViewWidget	<AnswerView widget instance>	AnswerView	Provides this widget with a way to display contents of a result.
showExcerpt	true	Boolean	Shows an excerpt of the result's contents.
	false		Does not show an excerpt of the result's contents.
deflection	true	Boolean	Appends all CSS class names with 'Deflection' to provide the ability to style the same instance in two ways: regular and CaseDeflection.

TABLE 44. Configurable Attributes

Name	Values	Type	Description
isNarrowStyle	false	Boolean	Does not append all CSS class names with 'Deflection'.
	true		This widget is rendered in full width.
	false		Default value. This widget is rendered in narrow width. Width may be modified in the CSS.

TABLE 45. Properties

Name	Type	Description
searchText	String	Value from the user input of the <Search widget instance>.
searchID	String	A random ID generated by search when a new search is performed.
selectedResultRow	Object	Holds the selected result and applies the CSS rule with suffix 'selectedResultClass' to this element for visual styling.
strings	Object	Localized texts in the current language.

TABLE 46. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
paginationTop	DIV	im-paginationtop	Renders pagination information and links at the top of the widget.
paginationBottom	DIV	im-paginationbottom	Renders pagination information and links at the bottom of the widget.
wizardsBlock	DIV	wizardsBlock	Contains the set of wizard results.
resultList	TR	results	Contains the table of results.
titleDiv	DIV	titleAnswer	Result list title.

TABLE 47. Event Flags

Name	Description
_eventPrevented	An event has been prevented from being triggered. Respective _before- _on- _after-events not executed.
showResult	_onShowResult has finished executing.
clickThrough	_onClickThrough has finished executing.
selectPage	_onSelectPage has finished executing.
toggleExcerpt	_showExcerptFn has finished executing.

TABLE 48. Event Callbacks

Name	Description
onRenderWidget	The function in which the widget is rendered. If no value is specified, default widget rendering behavior executes. Any specified function overrides the default behavior.
afterRenderWidget	If no value is specified, nothing additional executes after the widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.

TABLE 48. Event Callbacks

Name	Description
onShowResult	Changing the 'searchResult' data triggers this function. The default behavior first fires the 'showResult' and the widget re-renders. Upon finishing, it sets the 'showResult' event flag to true. Any specified function overrides the default behavior.
beforeShowResult	If no value is specified, nothing additional executes before onShowResult. If a function is specified, the function is executed before either one of the default or overriding functions for onShowResult is finished.
afterShowResult	If no value is specified, nothing additional executes after onShowResult. If a function is specified, the function is executed after either one of the default or overriding functions for onShowResult is finished.
onClickThrough	Clicking the link of an answer triggers this function. The default behavior first fires the 'showResult' event. Then, it fires the 'show' event for <AnswerView widget instance>. Any specified function overrides the default behavior.
beforeClickThrough	If no value is specified, nothing additional executes before onClickThrough. If a function is specified, the function is executed before either one of the default or overriding functions for onClickThrough is finished.
afterClickThrough	If no value is specified, nothing additional executes after onClickThrough. If a function is specified, the function is executed after either one of the default or overriding functions for onClickThrough is finished.
onSelectPage	Selecting a different page from pagination triggers this function. The default behavior first fires the 'selectPage' event. Then, it delegates to the <SearchWidget instance>, which ultimately updates and re-renders the 'resultsList'. Any specified function overrides the default behavior.
beforeSelectPage	If no value is specified, nothing additional executes before onSelectPage. If a function is specified, the function is executed before either one of the default or overriding functions for onSelectPage is finished.
afterSelectPage	If no value is specified, nothing additional executes after onSelectPage. If a function is specified, the function is executed after either one of the default or overriding functions for onSelectPage is finished.
onShowExcerptFn	This function is triggered when the excerpt icon is clicked. The default behavior first fires the 'toggleExcerpt' event. This shows or hides the excerpt of the result. When finished, it sets the 'toggleExcerpt' event flag to true. If a function is specified, the default behavior is overridden.
beforeShowExcerpt	If no value is specified, nothing additional executes before onShowExcerptFn. If a function is specified, the function is executed before either one of the default or overriding functions for onShowExcerptFn is finished.
_afterShowExcerpt	If no value is specified, nothing additional executes after onShowExcerptFn. If a function is specified, the function is executed after either one of the default or overriding functions for onShowExcerptFn is finished.

## CaseDeflection

The CaseDeflection widget is used to intercept users' questions submitted to Support. Before submitting, this widget can run a search on the text from the submission form and provide possible answers with a Results widget and the instance of Search that the Results widget uses. It requires a Result instance.

**TABLE 49. Configurable Attributes**

Name	Values	Type	Description
utility (required)	OKUtility	<OKUtility instance>	Provides this widget with ability to grab data from InfoCenter.
searchWidget	<Search widget instance>	Search	Provides this widget with ability to receive the latest results set.
resultWidget (required)	<Result widget instance>	Result	Set of results to display.
answerViewWidget	<AnswerView widget instance>	AnswerView	Provides this widget with a way to display the contents of a result.
submitOnNoAnswer	true	Boolean	An excerpt of the result's contents is not shown.
	false		Case Deflection will still be displayed even if no answer was found in Oracle Knowledge.

**TABLE 50. Exposed DOM Elements**

Name	Tag	CSS Suffix	Description
caseDeflectionCover	DIV	caseDeflectionCover	Overlay for fade out effect when case deflection is displayed.
caseDeflectionDiv	DIV	caseDeflectionDiv	Outermost Div of the widget.
topDiv	DIV	topDiv	Contains 'closeLink'.
bottomDiv	DIV	bottomDiv	Inside 'caseDeflectionDiv'. Holds 'answeredQuestionButton', 'submitQuestionButton', and 'editQuestionButton'.
closeLink	A	closeLink	Hides the widget.
caseMessageDiv	DIV	caseMessageDiv	Inside 'caseDeflectionDiv'. Holds message 'Your question hasn't been submitted yet.'
questionDiv	DIV	questionDiv	Inside 'caseDeflectionDiv'. Container for the SearchWidget instance.
centerTitleDiv	DIV	centerTitleDiv	Inside 'caseDeflectionDiv'. Holds message 'The following answers might help you immediately.'
answerListDiv	DIV	answerListDiv	Inside 'caseDeflectionDiv'. Container for the ResultWidget instance.
answeredQuestionButton	BUTTON	answeredQuestionButton	Inside 'bottomDiv'. User should click this if one of the results has answered the question that was about to be submitted. This cancels case submission.

TABLE 50. Exposed DOM Elements

Name	Tag	CSS Suffix	Description
submitButton	BUTTON	submitButton	Inside 'bottomDiv'. User should click this if none of the results has answered the question to be submitted. This completes case submission.
editQuestionButton	BUTTON	editQuestionButton	Inside 'bottomDiv'. The user has not found an answer yet and should click this to help find another answer to the question to be submitted.

TABLE 51. Event Flags

Name	Description
_eventPrevented	An event has been prevented from being triggered. Respective _before- _on- _after-events not executed.
show	_onShow has finished executing.
dismiss	_onDismiss has finished executing.
changeQuestion	_onChangeQuestion has finished executing.
clicksubmitButton	_onClicksubmitButton has finished executing.
clickAnsweredButton	_onClickAnsweredButton has finished executing.
clickEditQuestButton	_onClickEditQuestButton has finished executing.

TABLE 52. Event Callbacks

Name	Description
onRenderWidget	The function in which the widget gets rendered. If no value is specified, default widget rendering behavior executes. Any specified function overrides the default behavior.
afterRenderWidget	If no value is specified, nothing additional executes after the widget is rendered. If a function is specified, the function is executed after either one of the default or overriding functions for rendering is finished.
onShow	Clicking the case Submit button (not a part of the widget) triggers this function. The default behavior first fires the 'show' event. Then, 'questionDiv' is hidden, and 'contentBox' is shown. Any specified function overrides the default behavior.
beforeShow	If no value is specified, nothing additional executes before onShow. If a function is specified, the function is executed before either one of the default or overriding functions for onShow is finished.
afterShow	If no value is specified, nothing additional executes after onShow. If a function is specified, the function is executed after either one of the default or overriding functions for onShow is finished.
onDismiss	Clicking the Close button triggers this function. The default behavior first fires the 'dismiss' event. Then, 'contentBox' and 'questionDiv' are hidden. Any specified function overrides the default behavior.
beforeDismiss	If no value is specified, nothing additional executes before onDismiss. If a function is specified, the function is executed before either one of the default or overriding functions for onDismiss is finished.

TABLE 52. Event Callbacks

Name	Description
afterDismiss	If no value is specified, nothing additional executes after onDismiss. If a function is specified, the function is executed after either one of the default or overriding functions for onDismiss is finished.
onClicksubmitButton	Clicking submitQuestionButton triggers this function. The default behavior first fires the 'clicksubmitButton' event. Then, the 'dismiss' event fires. Any specified function overrides the default behavior.
beforeClicksubmitButton	If no value is specified, nothing additional executes before onClicksubmitButton. If a function is specified, the function is executed before either one of the default or overriding function for onClicksubmitButton is finished.
afterClicksubmitButton	If no value is specified, nothing additional executes after onClicksubmitButton. If a function is specified, the function is executed after either one of the default or overriding function for onClicksubmitButton is finished.
onClickAnsweredButton	Clicking answeredQuestionButton triggers this function. The default behavior first fires the 'clickAnsweredButton' event. Then, the 'dismiss' event is fired. Any specified function overrides the default behavior.
beforeClickAnsweredButton	If no value is specified, nothing additional executes before onClickAnsweredButton. If a function is specified, the function is executed before either one of the default or overriding function for onClickAnsweredButton is finished.
afterClickAnsweredButton	If no value is specified, nothing additional executes after onClickAnsweredButton. If a function is specified, the function is executed after either one of the default or overriding function for onClickAnsweredButton is finished.
onChangeQuestion	Clicking the search button of the SearchWidget triggers this function. The default behavior first fires the 'changeQuestion' event. Then, the results refresh. Any specified function overrides the default behavior.
beforeChangeQuestion	If no value is specified, nothing additional executes before onChangeQuestion. If a function is specified, the function is executed before either one of the default or overriding functions for onChangeQuestion is finished.
afterChangeQuestion	If no value is specified, nothing additional executes after onChangeQuestion. If a function is specified, the function is executed after either one of the default or overriding functions for onChangeQuestion is finished.
onClickEditQuestButton	Clicking editQuestionButton triggers this function. The default behavior first fires the 'clickEditQuestButton' event. Then, 'questionDiv' is shown. Any specified function overrides the default behavior.
beforeClickEditQuestButton	If no value is specified, nothing additional executes before onClickEditQuestButton. If a function is specified, the function is executed before either one of the default or overriding function for onClickEditQuestButton is finished.
afterClickEditQuestButton	If no value is specified, nothing additional executes after onClickEditQuestButton. If a function is specified, the function is executed after either one of the default or overriding functions for onClickEditQuestButton is finished.

TABLE 53. Properties

Name	Values	Type	Description
Question	<question text>	String	Value of input text that is used by this widget's search instance as 'searchText'.
strings	<Localized texts>	Object	Localized texts in the current language.



## Exposed DOM Elements

Exposed DOM Elements are particular HTML elements that are accessible from outside the widget. For example, if you want to get the UL tag of the Channel widget, you can write `channelWidget.get('channelList');` and store it in a variable.

## CSS

All of the CSS classes that widgets have are documented in their respective table for Exposed DOM Elements. You can find these classes in the *oracleKnowledgeWidgets.css* file, where you can edit them. You can also have multiple instances of the same widget and style them all differently from each other. For example, if you wanted two AnswerList widgets, you could render each of them in separate HTML elements.

```
<div id="answerList1"><!--answerList1 will be rendered here--></div>
<div id="answerList2"><!--answerList2 will be rendered here--></div>
```

To differentiate how each 'answerListResult' class is defined inside the CSS file:

```
/* styles for answerList1 */
#answerList1 .answerListResult {
    background-color: red;
}
/* styles for answerList2 */
#answerList2 .answerListResult {
    background-color: blue;
}
```