KRAD - Help Framework

Purpose

The main purpose behind this requirement is to support a good user experience, with task-specific help content that is delivered electronically in the context of doing tasks within Kuali software, so the user does not have to navigate separately to find relevant help content.

In order to achieve this, we need to support a help information architecture that makes it easy to create, update, maintain, translate and link high-quality help content with the logic/codebase.

See slide deck for KAI review meeting 2-16-2012.

Detailed Description

Primary:

- #1: Create a context-sensitive help information architecture, that supports
    - Concise, brief help in a tooltip, which could include text only or also other content that can be interacted with (links, etc.). Tooltips are intended for field and object-level help, but there is no technical reason they could not be used also for sub-section, section or any higher level of help. The key is that these are short (can include a link to additional help that would open in another window - see next bullet) and are task-specific, not leading the user into content not relevant for the immediate task/choice.
    - Document, Page or View-level help in a separate browser window (opened to a specific anchor point that is relevant for that page, and sized & positioned appropriately). (Note that KRAD will support and simplify this linking, it will not determine the content management system or user interface that is presented in the separate browser window. See the secondary requirement below.)

- #2: Create an integrate-able help information architecture that makes it easy to create, update, maintain, translate and link high-quality, context-sensitive help content within the logic/codebase:
    - That is separable from the logic/code-base.
    - That makes it easy to programmatically link field, sub-section, section, page, view and document-level help "hooks" into the logic/code-base, to appropriate anchor-points in the help content/architecture. This could be to anchor points within a single online help structure, to deliver the right information from within the larger information space (see the page-level help model in the material below). Or, it could be to field and section-level tooltip content pulled from a single resource file or other centralized help construct (related to the requirement for Centralized Message Repository).

- #3: Enable applications to choose
    - which level(s) of the help content to implement. For example, one application might need field-level help only, while another needs field-level and section-level help, while another needs field-level and page-level help, and so on.
    - which construct to use for which level of help in the application. For example, one application might need tooltips for sections, while another might need the separate help icon & help window for sections.

- #4: Support the integration/use of the html output from any authoring tools that output into standard XML/HTML source -- essentially, support pulling the content from any content management repository selected by the application / university.

Ideally, the user interface to the application itself should be designed in such a way that it does not require a large amount of collateral information. We caution against making the user feel lost in too much information. This is not aimed to be full conceptual information about the application's architecture, rather, is aimed to be task-specific help.

The main requirement is to support a good user experience with online help, with task-specific content that is delivered electronically in the context of doing tasks within Kuali software, so the user does not have to navigate separately to find relevant help content.

Note that this represents a step towards, but is not the same thing as full convergence that unifies all help and electronic documentation.

Secondary:

- #5: Create an online help architecture that fully converges the aspect of online help with the aspect of electronic documentation.

  If this is achieved, it is conceivable that all user help and user documentation could be from the same source - there would be no duplication of effort required to create, update or maintain the content.

  The primary requirement even in #4 is that when accessing task-oriented help (such as field-level, group-level), the potential unity with all user help information is achieved in a way that does not lead the user to become lost within a larger online documentation structure. See the usage scenarios, and the mocks and diagrams sections above (these are coming).

  The user interface to this type of converged overall help/documentation structure must support the following:
    - table of contents - Accordion-style expandable table of contents in left sidebar
    - help index - Alphabetical index

- help search - Great search capability with highlighted hits
- help glossary
- hyper-linked cross-references within and across the help content and these constructs.
- The information in this converged overall form must be printable, with pagination cues (page numbers printed).
- Ability to save help topics to Favorites for infrequent tasks
- Feedback e-mail capability

Usage Scenarios

Usage Scenario One:

A student is researching the courses that will be offered in the next semester, and selecting those of potential interest to put into a list to then research further before deciding which to enroll in. In viewing the list of course abbreviations, she wonders about a particular abbreviation, since in her mind it could stand for several offerings, one of which she's interested in. She hovers over the course label and views a simple text tooltip that spells out the full course name, which shows that this is indeed the course she is interested in. She selects it to add to her draft list.

Usage Scenario Two:

A faculty member is adding details into the course listings for the next year, about a new class he will offer. He's about half-way through the task and gets to a sub-section of the form he's working with, and is not sure he needs to complete this sub-section (and he doesn't think he has enough information to be able to do so). He hovers on the sub-section label and views a rich tooltip that provides additional information about the sub-section, but he still has questions. He clicks on the link at the bottom of the tooltip to view additional information. When he does, the tooltip closes and another browser window opens, slightly overlayed on top of the application window. He views the information side-by-side with the form fields in this section and finds he now has enough information to complete half of them. He does so and saves the form, to work on it again tomorrow after he asks his administrator about the few remaining required fields.

Mocks and Diagrams

For context, in this section we cover

- what is already supported in KNS
- what is already supported in KRAD, new in Rice 2.0
- the additions planned for Rice 2.2 KRAD, to both support functional equivalency with KNS and to enhance the online help capabilities, by achieving the primary objectives described above.

For context, what "help" is already supported

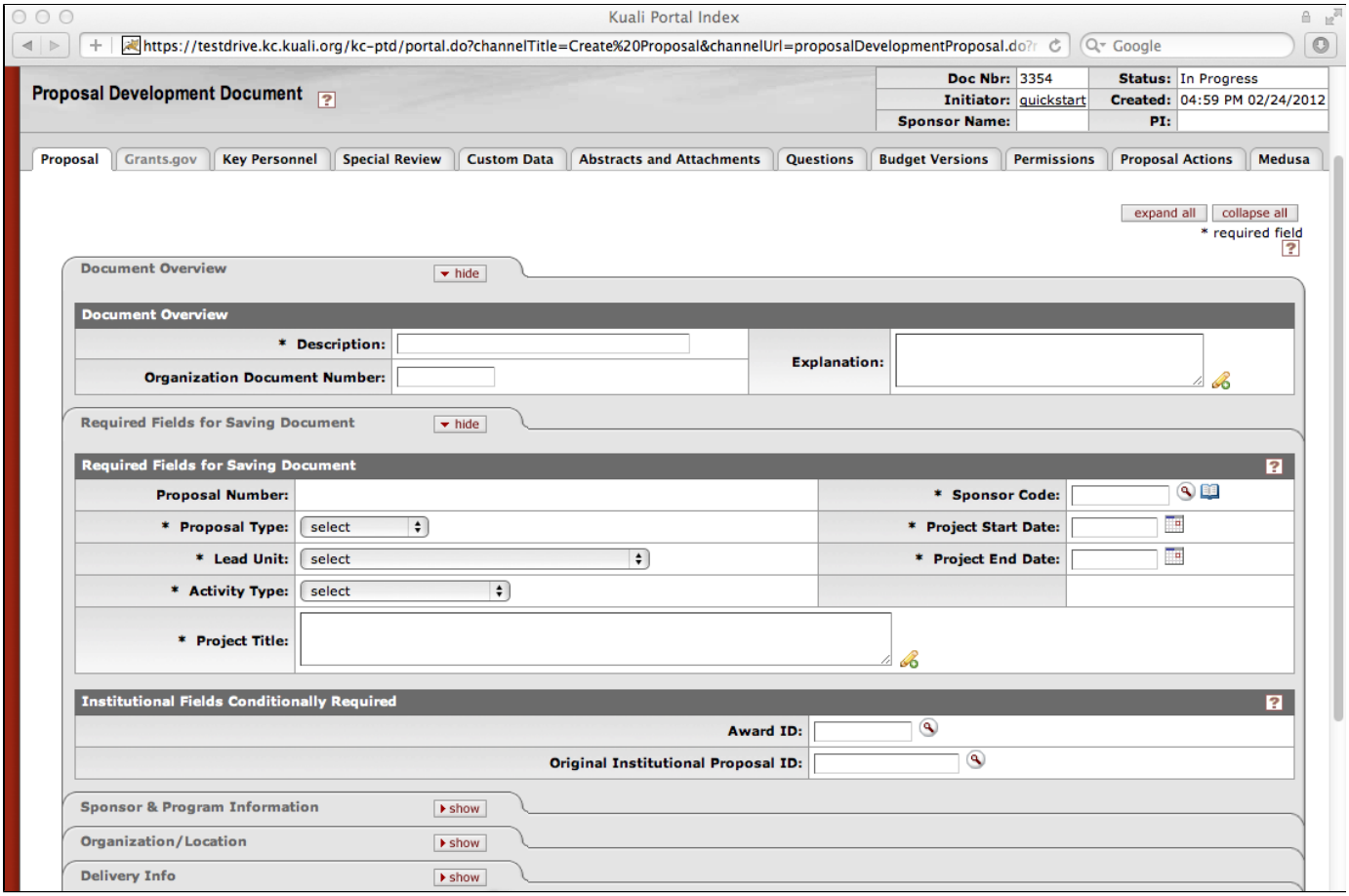
Already Supported in KNS:

Kuali applications today, such as Kuali Financial System and Kuali Coeus, provide context-sensitive help for some items, for example, at the document type or document level.

To explore the KFS Test Drive (login as admin). To explore the KC Test Drive (login at quickstart).

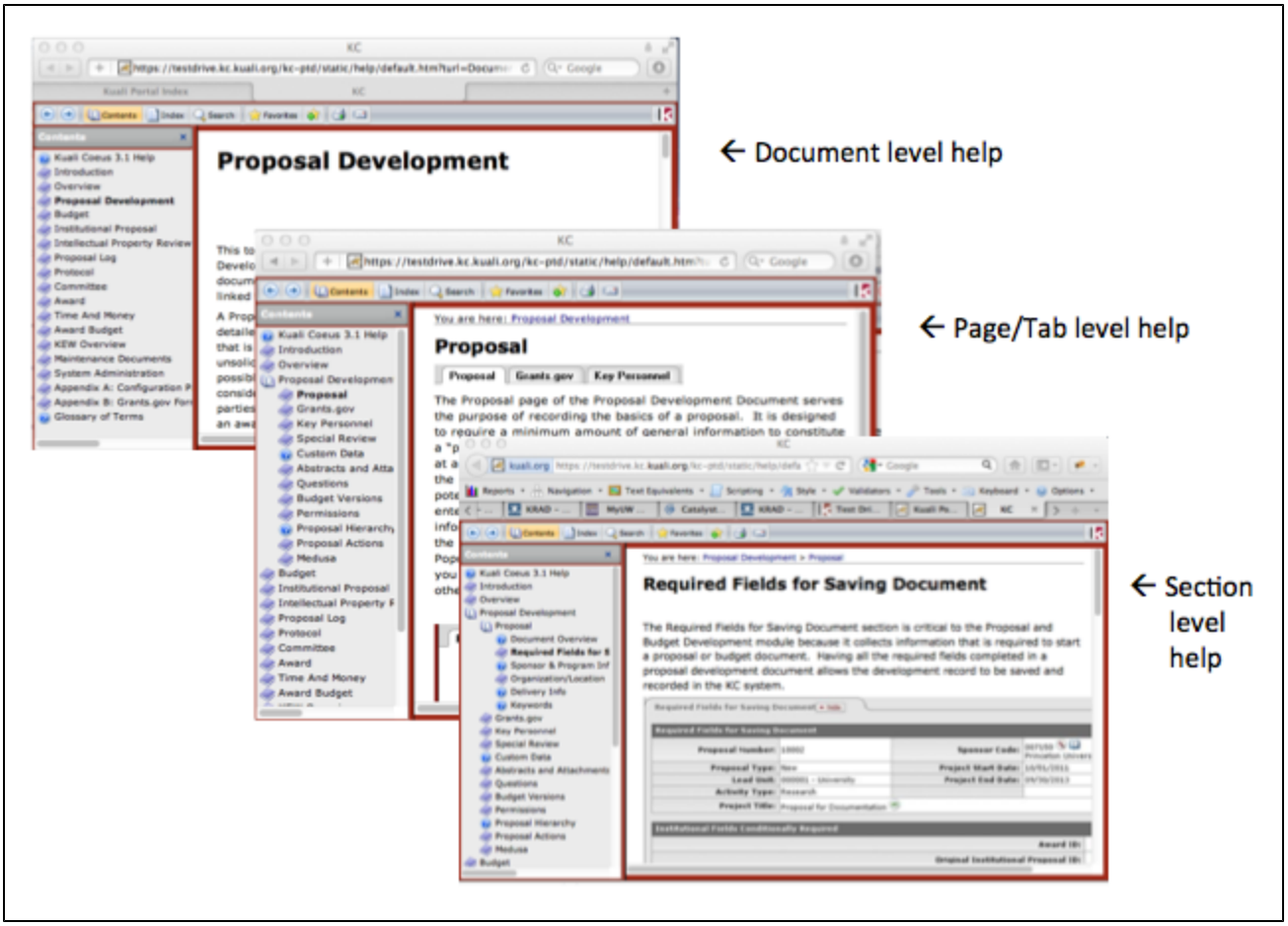
For example, see the following screenshot from the KC test drive. There is a help icon at the following levels:

- The document level (associated with the "Proposal Development Document" heading's title tag)
- The page/tab level (associated with the "Proposal" tab's page div - *look to the right, under the "*required field" text*)
- The section level (associated with the spans defined for the "Required Fields for Saving Documents" and "Institutional Fields Conditionally Required" headings)

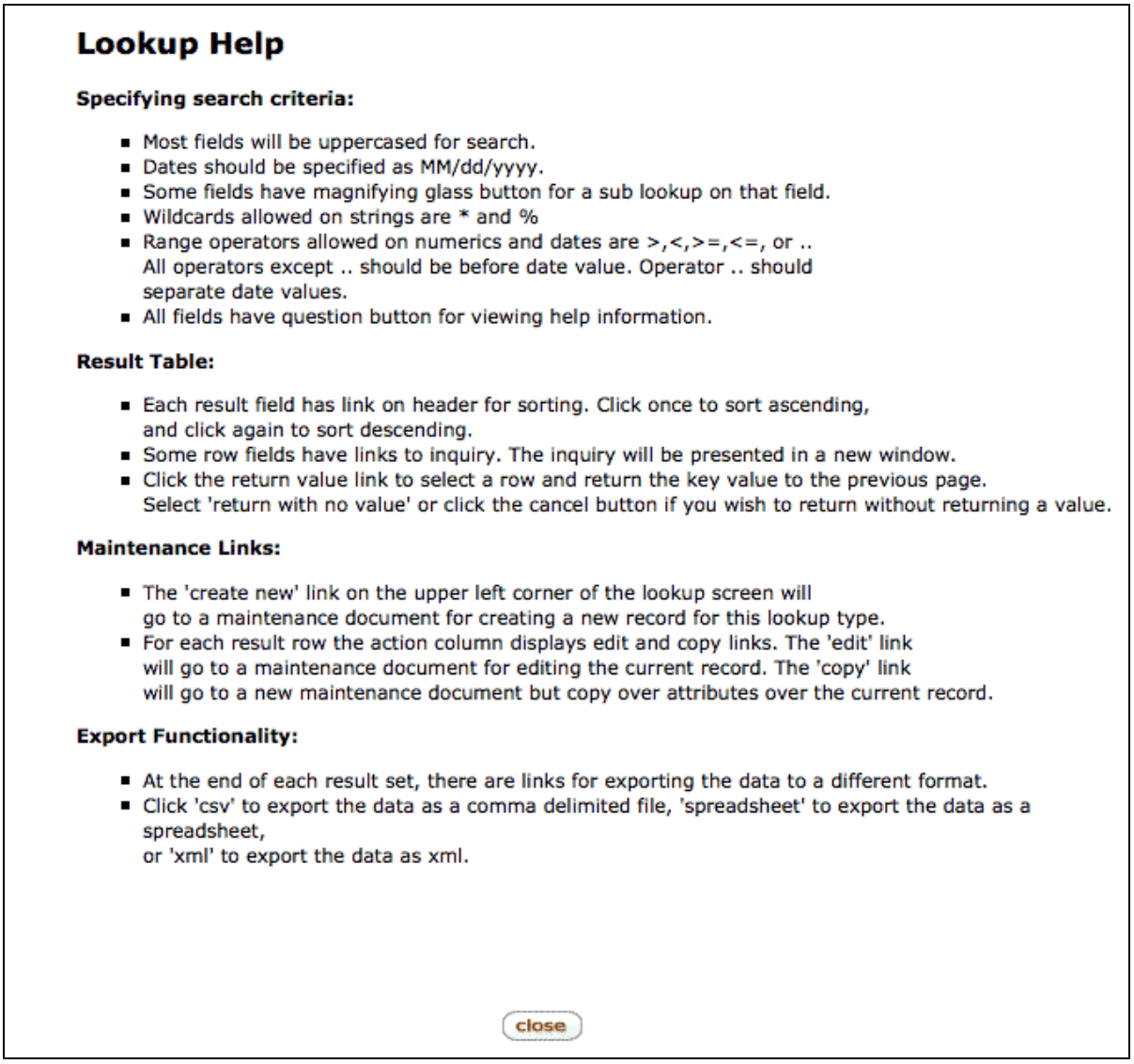


When the user activates the document-level help icon, another browser tab opens with the "stand-alone" electronic documentation opened to the appropriate place within the larger information content. Therefore, context-sensitivity is achieved at the document, page/tab, and section levels, a significant step forward.

Document-level, page/tab level, and section level help window examples:



Some documents have help at the document type level rather than context-specific help. For example, the help for documents that are lookup types, are information about lookups in general, instead of information for the specific task or field in the context of the user's lookup task. This example can be found in both the KC and KFS test drives:



There are pros and cons to each of these approaches.

How the help icon is linked with the appropriate content today:

Today, the KFS online help authors complete a spreadsheet that identifies the URL for each help topic (the name of the top-level html file for each document), and delivers the spreadsheet along with a single zip file that contains the html files, to the application's development team. The developers then define in a workflow document which html file url is associated with which document type ID. The shorter document type IDs are used to associate the url with its associated data dictionary xml file.

Below is example content from this type of workflow document (note the "AGCY" name - the document type ID - is being associated with the helpdefinition url/file "agency.htm".

```
 <data xmlns="ns:workflow" xsi:schemaLocation="ns:workflow resource:WorkflowData"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <documentTypes xmlns="ns:workflow/DocumentType"
xsi:schemaLocation="ns:workflow/DocumentType resource:DocumentType">

  <documentType>
      <name>
        AGCY
      </name>
      <parent>
        CGSM
      </parent>
      <label>
        Agency
      </label>
      <helpDefinitionURL>
        default.htm?turl=WordDocuments%2Fagency.htm
      </helpDefinitionURL>
      <active>
        true
      </active>
      <routingVersion>
        2
      </routingVersion>
    </documentType>

  </documentTypes>
  </data>
```

After the help workflow document and its contents are defined (example above), developers insert the appropriate document type ID into the place in the code-base where the target UI element is defined, so that this help content is displayed in the new browser tab for the help when the user activates the help icon associated with that UI element.

See the following code snippet from a KFS example for context-sensitive help. The URL tells the application where the help content is located and which topic is the anchor point to open to (the document type ID of "INVW" in this example, will be resolved on the server, to the appropriate helpdefinition url/htm-file).

```
<div id="headerarea" class="headerarea">
    <h1>
     Customer Invoice Writeoff 
      <a title="[Help]document help" target="helpWindow" tabindex="-1"
       href="http://testdrive.kfs.kuali.org/kfs-ptd/kr/
       help.do?methodToCall=getDocumentHelpText&documentTypeName=INVW">
           <img hspace="5" border="0" align="middle"
            alt="[Help]document help" src="http://testdrive.kfs.kuali.org/
            kfs-ptd/kr/static/images/my_cp_inf.gif">
      </a>
    </h1>
<div class="headerbox">
</div>
```

And the following code snippet brings up generic help information (for lookups), rather than context-specific help:

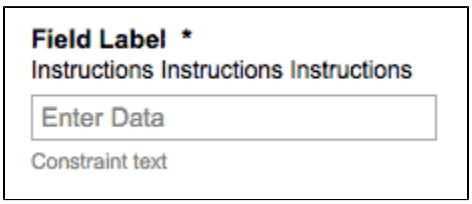
```
<div id="headerarea-small" class="headerarea-small">
<h1>
Pessimistic Lock Lookup
<a title="[Help]lookup help" target="helpWindow"
tabindex="-1" href="https://testdrive.kc.kuali.org:443/kc-ptd/kr/help.do?methodToCall
=getLookupHelpText&lookupBusinessObjectClassName=
org.kuali.rice.kns.document.authorization.PessimisticLock">
<img hspace="5" border="0" align="middle"
alt="[Help]lookup help"
src="/kc-ptd/kr/static/images/my_cp_inf.gif">
</a>
</h1>
</div>
```

Though a context-sensitive help structure is possible, prior to Rice 2.2 KRAD, linking appropriate help content at the field or sub-section-level via a help icon, for context-sensitive help, is not done in most Kuali applications today. It may be perceived as significant additional work to accomplish through the method described above, and/or may be perceived as not needed for the target audience.

What is already supported in Rice 2.0 KRAD:

KRAD in Rice 2.0 supports including visible text anywhere on a page. This enables an application to include short explanatory text at the top of a form or at the beginning of a section -- virtually anywhere in the view, where visible explanatory text is needed and appropriate.

In Rice 2.0, KRAD also makes it easy to associate "visible" instructional text and constraint text with input fields. The input field itself can include a watermark "suggestion" or it can be pre-filled with a valid default selection alternative, that the user can accept or change. The field can include auto-complete behavior (recognition/type-ahead).



The input field is grouped with a label, which can include a "required-ness" indicator (the *). As with the KNS capability, the input field can have other controls associated with it, such as a lookup or inquiry control, and could have multiples of these. It can also include a drop-down control that enables the user to select from a valid set of alternatives (not shown).

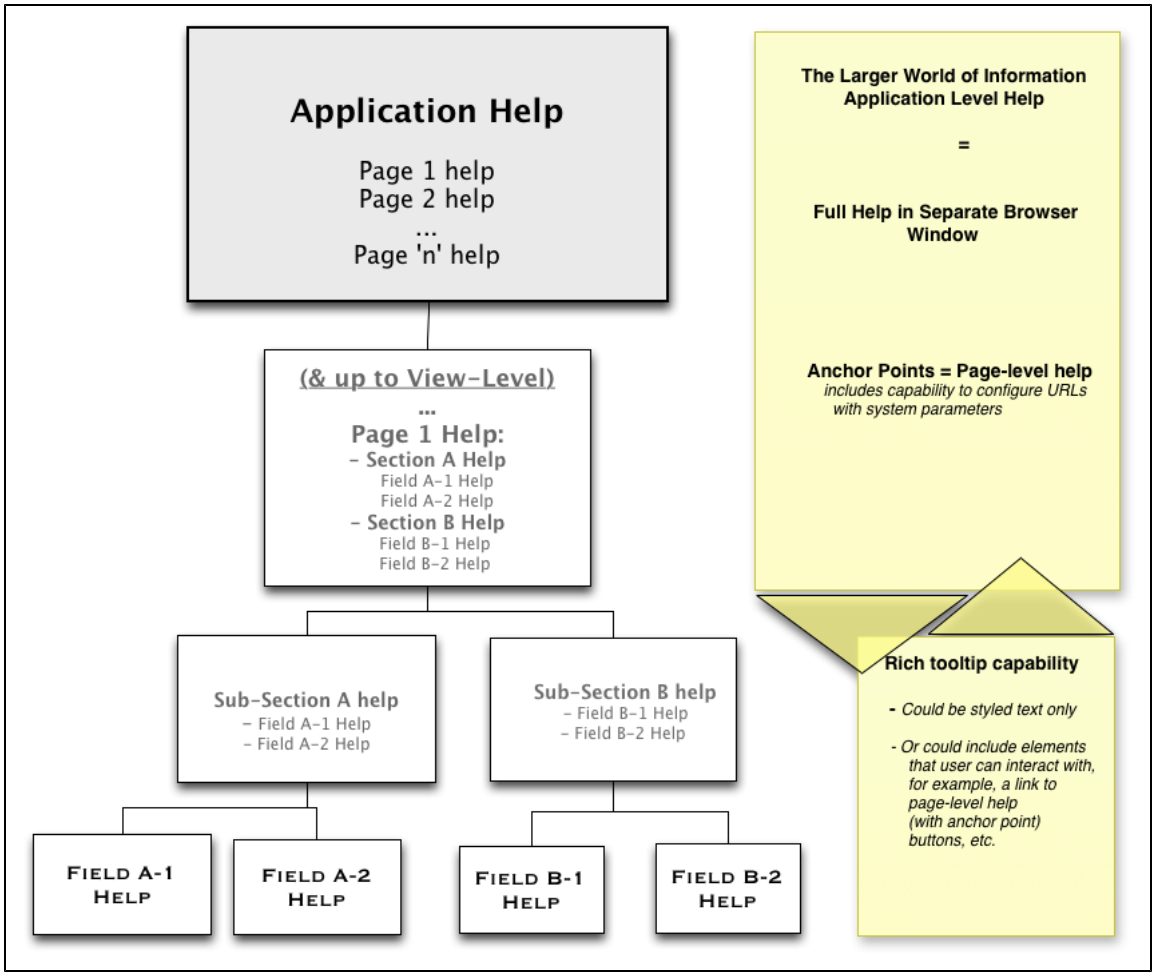
All these are optional, but available to the application designer.

In addition, as with KNS, a help icon can be placed in the view, for example to provide document or view-level, or page-level, section or sub-section level, or field-level help. However, if there are many field-level help icons throughout a densely constructed page, this could lead to visual clutter, making it more difficult for the user. And it could be especially problematic if a help icon is added to a field that already has a lookup or inquiry icon (or both!).

What "help" constructs will be new in Rice 2.2, KRAD:

There is a new UI construct related to help architecture in Rice 2.2 KRAD, and a data structure addition to make it easier to hook the appropriate help content into the appropriate place in the UI when developing an application.

Diagram of the relationships among help constructs:



The diagram above shows how the new help framework can support the KNS model, in addition to extending it to support the rich UI needs for student-facing applications and for administrative functions that can't incur a steep learning curve for infrequently performed tasks. In this model:

- **Help Tooltip:** Lower-level help, that is concise and specific to one particular decision, and which should not interrupt the users' task by bringing up a separate window that will have to be managed, can be provided in a tooltip. This could be particularly helpful for field-level and sub-section level help. Tooltips can be simple text or can contain links or other controls that the user can interact with. In either case, the tooltip appears on hover (or focus for keyboard users) and disappears when the user moves outside of the contiguous area formed by the trigger element and the tooltip content.
    - *Note that tooltips can include a link to an anchor point in the larger world of information, to the full application-level documentation.*
- **Help Window:** Higher-level help, that could be extensive, or could be helpful for the user to see in the context of the other help information, can be provided in a separate browser window. This could be particularly helpful for View-level (e.g. document types), Page-level, and Section-level help.

Notes:

- *There is no technical obstacle to providing the tooltip for any level of the help, though there will be a size constraint (you do not want to have a very large tooltip, too much content).*
- *There is also no technical obstacle to providing the icon and separate browser window for any level of help, though having another window open for very brief field-level help isn't recommended (you do not want to have many help icons littering the page, and there may be no room to place a help icon on some elements).*
- *The recommendation, however, is that the tooltip is used for lower-level help and that the help icon/window is used for higher-level help.*
- ***It is also recommended that this decision be consistent within an application.*** *For example, if a tooltip is used for section-level help on one page, it is used for section-level help on all pages (and vice versa, if a help icon/window is used for section-level help on one page, it is used for section-level help on all pages).*

These are application-level decisions. The page-level help content structure itself is also up to the application. Rice 2.2 KRAD, as a middleware platform, aims to be agnostic with respect to the Content Management System chosen by the application or university system, rather than to require a specific CMS. KRAD will include default templates for configuring URLs with system parameters, so that applications can link into their application help content from the appropriate points in the application, to the appropriate anchor points in the help content. *(We will work with the Kuali applications to provide a good reference platform / model for other applications and teams coming new to Rice.)*

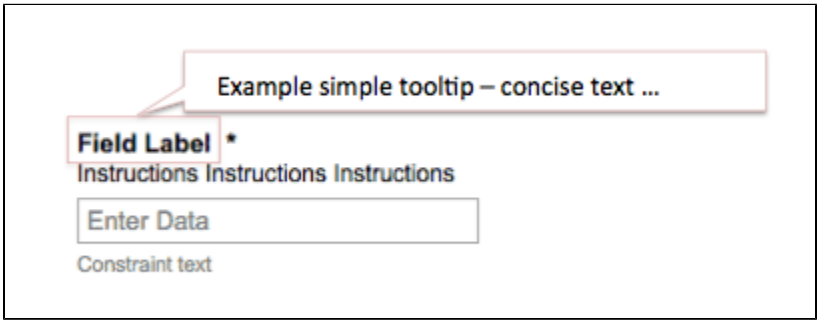
Tooltips:

Research has shown that users often don't notice helpful text on a densely laid-out page, so the tooltip strategy for field-level and section-level help offers the application designer a "just-in-time" help capability.

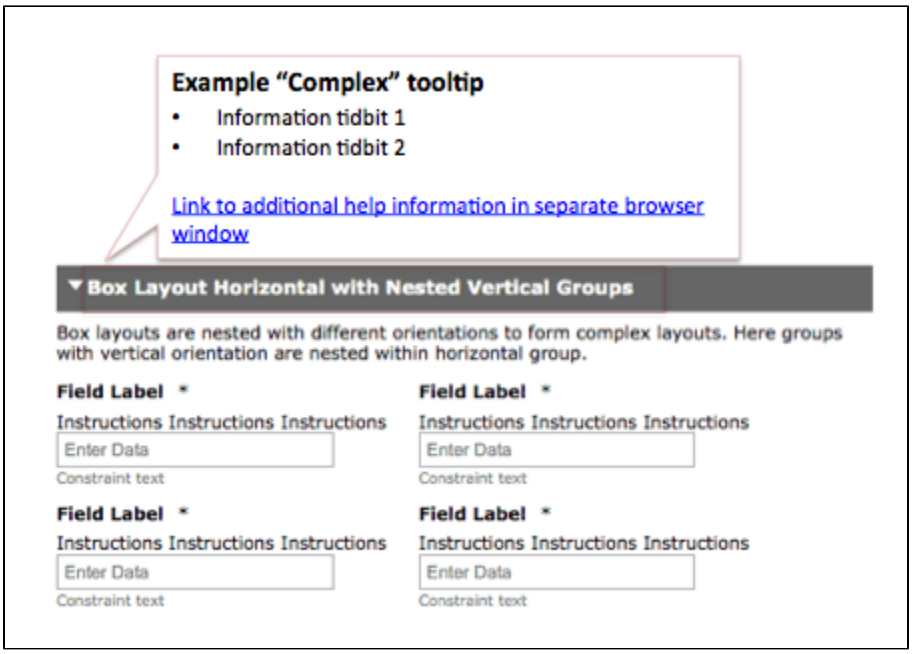
There will be no help icon in the UI to signal that there is a tooltip associated with an element. But tooltips have become standard fare in rich applications, and users now actively explore user interfaces looking for these relevant 'nuggets' of information (tooltips and other) when they use a new application or infrequently use it.

The tooltip appears on hover (and on focus for keyboard users) and disappears when the user moves outside of the contiguous area formed by the trigger element and the tooltip content area. The content can be concise, styled text, or richer content enabling user interaction.

Below is a low-fidelity example of a text-only tooltip (specific visual treatment yet to be determined):



And below is a low-fidelity example of a richer tooltip, containing a link that the user can select (specific visual treatment yet to be determined):

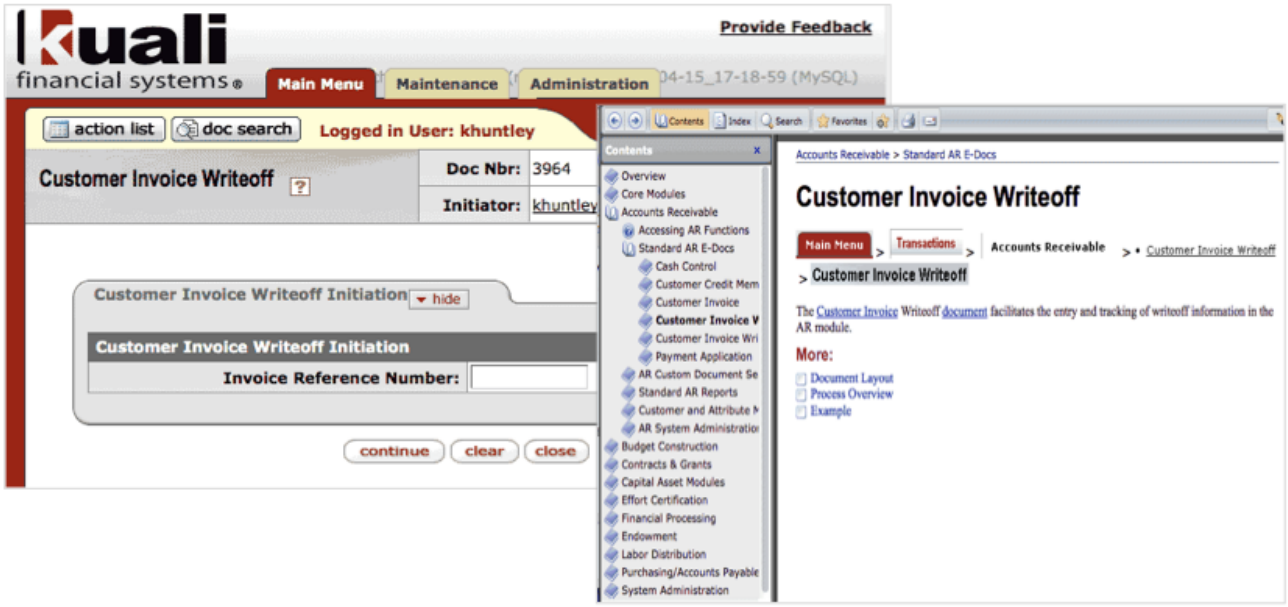


See the tooltip requirement for additional information and details on the tooltip construct.

The separate Help window for view and page-level help:

This is the same behavior as available today in KNS, except we want to launch a new browser window (not a tab), and open it with a default window size and placement, relative to the application window. This is tbd.

See how this would appear in the KFS content/example (note that there will be a new help icon & visual treatment in Rice 2.2 KRAD:



Performance

If applicable, list expectations for performance (optimal and worst cases would be fine, give time in seconds):

References

- See the tooltip requirement for additional reference information and details.

See also the following background reference information on how Kuali applications can link their online help content with a help icon today:

1. "Customizing" -- for technical writers/help developers. Describes how implementing institutions can customize the out-of-box foundation-delivered content based on their unique customizations and implementation.
2. "Configuring" -- for software developers/database administrators. Describes how the published help content is configured in the Kuali application so that the help icons take the user to the appropriate topic. Covers configuring the "hooks" to integrate the help with the application user interface -- involves modification of data dictionary files and help parameters in the master data source.

Requirements Listing

Primary:

The Help Framework consists of the following:

1. Tooltip:

The below is a high-level overview only. For more details, see the separate tooltip requirement.

1. The tooltip is the UI component that delivers context-sensitive help for fields and other user-manipulable controls.
2. When the user hovers over a tooltip trigger element (e.g., a field label or a control in a form), brief help summary text for the specific field/control should be displayed as a tooltip.
3. The tooltip should remain open while the control or field label is selected.
4. The tooltip will close when the user moves out of the contiguous area formed with the tooltip trigger element and the content itself.

2. Help Window associated with the Help icon/button:

1. KRAD should enable displaying a help icon associated with any element, for example, with a view header (in KNS, this would be a document) or a page header. The default placement of this help icon will be to the right of its associated UI element. (Precise default visual treatment and positioning to be determined.)
    a. The KRAD code structure will group this icon appropriately with its element (i.e., fieldset/legend semantics in the code structure

will enable assistive technologies to convey that this help icon is for its associated element).
2. KRAD should ship with a default help icon (precise default visual treatment to be determined), with default alt-text for it. (Research whether the alt-text can include a variable that enables it to be pre-filled with the associated element's label.)
    a. Secondary: KRAD should also allow the ability, at a global level (per application, not per view or page), to replace the default help icon with a different image or with text. Consistency across an application is important.
3. When the user hovers over the help icon, brief summary text should be displayed as a tooltip for the help icon. This doesn't include the help content itself, but a brief description of the type of help that selecting the help icon will produce. Examples: "Display help for *this page"*. (Research whether the help icon's tooltip content can include a variable that enables it to be pre-filled with the associated element's label or its alt-text.)
4. The Help icon should be selectable and, when selected, should open the help information in a new browser window (not new browser tab):
    a. The new browser window should be sized and placed by default, and appear above the application window in the z-order (not be occluded by the application window upon opening). After the initial opening of this new browser window, the user controls its size and positioning (standard operating system and browser controls). The user can move and resize the new browser window, as appropriate to the needs. The opened browser window appears in the operating system's open applications bar (standard behavior), helping the user to locate it if, in the user doing other things, it gets hidden by the application or other open windows.
    b. Applications should be able to configure a URL with system parameters, for example, to link the help icon with where the help content is stored and to name an anchor point within the help content.

Secondary

Today it is possible to turn off help (hide the help icon), but it must be done individually for each component. Rice 2.2 KRAD should differ from this KNS behavior in the following way:

1. It should be possible to assign the tooltip or the help icon/window usage to any element level within an application (for example, tooltip or help icon/window to section level headings).
    a. But the assignment should apply application-wide, not per heading or per page.
    b. And the help icon/window should not be applied for elements lower in the class hierarchy than elements that have already been assigned the tooltip. In building the application, this should be flagged as a violation, though this type of violation should be able to be accepted/overridden by the application team. (For example, sections assigned a tooltip should not include sub-sections assigned a help icon/window).
2. KRAD should not display a help icon with any element unless the application has explicitly associated a help ID or URL (this depends on the technical architecture, which will be specified soon) with the particular element. The help icon associated with the UI elements will be hidden unless there is an associated ID/URL. There should be no need to 'turn off' help icons in order to ensure there are no "dead" help icons. Therefore,
    a. There should be no need for KRAD to include a setting to enable an application (development/design intention, not per user during usage) to turn off (hide) the help icon, at a global level (per application).
    b. There should be no need for KRAD to enable doing this same thing at a view (document) and page level.
3. KRAD should have a mechanism for handling cases where a developer has put both a tooltip and a help icon on a target element (this should not happen). If there is a help icon with url content/id associated with an element, that should over-ride the tooltip - it should not be displayed.

Dependencies

List any functional or technical work that must be completed before work on this item can begin:

1. (This is listed in the tooltip requirement, which is a dependency in this overall help framework: Assess & select which jQuery or other tooltip plug-in to use.)
2. Research whether there is a way to automatically pre-check for the amount of content being put into a rich tooltip, to shift it instead into the separate browser window if the content exceeds the recommended limit. (The user experience will not be great if a tooltip is very large.)
3. Research whether the two choices described above -- the rich tooltip and the separate browser window, opened by default and sized/positioned appropriately -- satisfies the user experience objectives or if we need to put back in a 2nd help construct, which is a non-modal help dialog that is tightly aligned with the application window and is above it in the z-order (will not be occluded/behind the application window), that can be moved aside and outside of the application window by the user in order to view side-by-side and maximize the view, and that can be closed by the user when no longer needed.
4. Research whether the help icon's html tooltip content can include a variable that enables it to be pre-filled with the associated element's label or its alt-text.

Issues

List any issues that need to be resolved before work on this item can begin:

Functional:

1.

Technical:

1.

QA or Regression Testing Plan

Plan to test on:

- Firefox 10 and 11
- Chrome 18.0.1025.162 and 18.0.1025.163
- IE 8 and 9 (MS Windows only)
- Safari 5.0.6 and 5.1.5 (Apple MAC only)

Required basic test: Go through steps 1 - 5 below, using the typical, preferred navigation style (typical GUI users = selecting items and scrolling with mouse, entering data with keyboard).

1. Select a form to complete and submit: *(Note 1: Should we specify which form, and where in the build? We need to make sure we test with forms that have help icons associated with help content that will open in the separate help window, sized/positioned over the form, and have other trigger elements associated with tooltip display of help content - see list below. And at least one of the tooltips must contain a live link that brings up one of these help windows.)*
2. Hover the mouse over a help icon.
    a. Check to ensure that a simple tooltip displays with a short statement "Display help for <variable name>", where the variable name is appropriately filled in with the label of the UI element that the help icon is associated with. For example, if associated with the page, it is the page title, if associated with a section, it is the section header text.
    b. Activate the help icon. Ensure that a new browser window opens on top of the application, sized smaller than the application window and positioned within the user's field of view.
    c. Interact with the help window (scroll to view all help content, move it, resize it, close it).
3. Hover the mouse over the type of elements listed below to explore whether they have help tooltips associated with them:
    - Section and Sub-section headers
    - Tab labels (the tab list items)
    - Input field labels
    - Checkbox control
    - Radiogroup control
    - Buttons (including help icons)
    - Links
4. For any tooltip that opens upon hover (see list in step 3 above), ensure that it also closes when you move the mouse focus off the contiguous area formed by the trigger element and the tooltip content.
5. If there is active content within the tooltip (a link, button, any selectable control), ensure that you can move your mouse directly from the element over which it is positioned, into the tooltip area to interact with the content. Interact with it. Close it.

Additional tests:

Second: (This should be done by the devs before checking their code in for the milestone. It can also be included in QA testing.) Run a code-standards checker. See list of code-checker alternatives*. (Note - if this second test is run and checks well, the following checks may be redundant.)*

Third: Select large fonts through the browser's settings (200%) and the browser setting to ignore the web page's font settings. Visually inspect the UI while repeating steps 1 and 2, to ensure that all UI elements inherit well the larger font size. If there are any problems, return the settings to their prior, normal state, and then set the screen to lower resolution, e.g., 800x600, or change the DPI to 200% via the operating system settings. Visually inspect teh UI while repeating steps 1 and 2, to ensure that all UI elements inherit well these larger size settings.

Fourth: Select high contrast through the browser's settings as well as the browser setting to ignore the web page's color settings. Visually inspect the UI while repeating steps 1 and 2, to ensure that all UI elements inherit well the high contrast settings. If there are any problems, return the browser settings to their prior, normal state, and then select a high contrast theme in the operating system settings. Visually inspect the UI while repeating steps 1 and 2, to ensure that all UI elements inherit well the high contrast settings.

Fifth (Not testable yet in M1): Keyboard-only test. After completing these above, repeat steps 1 and 2, but without using the mouse, using only the keyboard. Unplug your mouse if you tend to forget and find yourself switching back and forth.

Sixth: Assistive technologies test: Repeat steps 1 and 2, using a screen-reader (NVDA, Jaws, and/or Voice-Over). Turn off your monitor if you tend to forget and find yourself using your vision to supplement the cues you are hearing. Note: it is not expected that every dev or every QA resource would do this. More efficient for 1 or 2 people to invest the time to become more skilled with screen readers' shortcut keys, etc., than for everyone to do so. But everyone is welcome to do so, using NVDA (free download for Windows), JAWS (free timed download for Windows) and Voiceover (comes free on the MAC). JAWS still has market share among screen-reading users.

Checkoff

Functional Analysis Complete? **No (completed by SME)**

Needs Review by KAI? **Yes**

Technical Analysis Complete? **No (completed by DM)**

Needs Review by KTI? **No (completed by DM)**

Estimate: **30 hours (completed by DM)**

Technical Design: Link Here **(completed by DM)**

Jira: KULRICE-6674

Final Documentation: Link Here **(completed by DM**