Rule and Search Attributes 2

One of the features of KFS Release 2.0 was a higher level of integration between Kuali Enterprise Workflow (KEW) and the Kuali Nervous System (KNS)'s data dictionary. This means that it's a whole lot easier to create workflow searches and rules. Here, we'll cover these new futures and show how to put them together to make KEW document types much more flexible.

The Data Dictionary and Search Attributes

Also see the KEW search documentation.

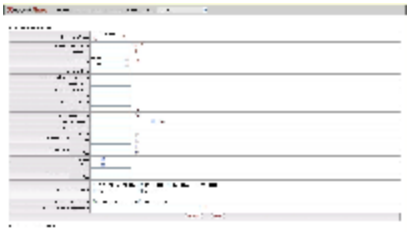
It's very common to do a normal document search in KFS.

But we've got to know the document number, or we have to look up the document type, or we get far too many documents back because we searched on initiators. It would be nice if we could make the search a bit more specific.

Let's say that we choose to search for a financial document of a specific type...an Internal Billing document.

Here, we've got more fields to help you search on the document: an account number, the amount of the document, and so on.

Now, let's say, you wanted to look up a Credit Memo document.



Look at all those extra fields!

Kuali Workflow actually adds functionality that makes it very simple to add extra search fields to a specific workflow document type, through a special class: org.kuali.workflow.attribute.KualiXmlSearchableAttributeImpl. This class allows search fields to be defined in XML and hooked into the data dictionary, so that those fields match their KNS equivalences. Let's see this is done.

How was that done?

As stated, we declare all of these extra fields in XML. Therefore, let's lookup the KualiFinancialDocument document type in workflow. When we look at the XML, we see:

```
<attributes>
  <attribute>
    <name>KualiAccountSearchAttribute</name>
  </attribute>
  <attribute>
    <name>KualiDocumentHeaderTotalAmountRange</name>
  </attribute>
</attributes>
```

The attributes tag associates rule attributes with the document type. Let's look up the KualiAccountSearchAttribute rule attribute. We find this:

```
1.  <ruleAttribute>
2.    <name>KualiAccountSearchAttribute</name>
3.
<className>org.kuali.workflow.attribute.KualiXmlSearchableAttributeImpl</className>
4.    <label>KualiAccountSearchAttribute</label>
5.    <description>KualiAccountSearchAttribute</description>
6.    <type>SearchableXmlAttribute</type>
7.    <searchingConfig>
8.      <fieldDef name="accountNumber" title="kuali_dd_label(SourceAccountingLine)">
9.        <display>
10.        <type>text</type>
11.       </display>
12.       <validation required="false">
13.         <regex>^[0-9 ]{1,10}$</regex>
14.         <message>Invalid Account Number Value.  Account numbers must be all
numeric, with at least one digit.</message>
15.       </validation>
16.       <fieldEvaluation>
17.
<xpathexpression>wf:xstreamsafe('//org.kuali.core.bo.SourceAccountingLine/accountNumbe
r') |
 wf:xstreamsafe('//org.kuali.core.bo.TargetAccountingLine/accountNumber')</xpathexpres
sion>
18.       </fieldEvaluation>
19.     </fieldDef>
20.   </searchingConfig>
21. </ruleAttribute>
```

There's a lot of stuff here, so let's go through it line by line.

Lines 3 and 6 are what give us the power to tie the search attribute into the data dictionary. In line 6, the rule attribute that we declare for extra search fields is "SearchableXmlAttribute", so we set that as the Rule Attribute's type. Likewise, as seen in line 3, the className for these search fields should be "org.kuali.workflow.attribute.KualiXmlSearchableAttributeImpl" or a child of that class; this class, again, is what allows us to define XML search attributes and tie them into the data dictionary.

Lines 7 through 20 are where we define the "searchingConfig" - show KEW will present the search. This is where the way field looks and what the field will search on are defined. In line 8, we have a fieldDef that declares a name for a field to search on and the title. The title, you'll note, uses the kuali_dd_label function to look up what the label for "accountNumber" should be, based on the data dictionary entries for SourceAccountingLine. That function looks up the label in the data dictionary and when the field is shown on the search screen, uses that label to show the search.

Lines 9 through 10 define what kind of search field we want. This example is really plain vanilla...it's just a text field.

Lines 12 through 15 show that we can add some validation to our fields; in this example, any input in the field is expected to match the given regular expression (ie, the account number should be between 1 and 10 digits or spaces) and if that's not true, the error message is shown.

Finally, in lines 16-18, we have our fieldEvaluation element, which define which fields of workflow documents should be searched. As we can see, an xpathexpression element is defined to find those elements, and then two XPath expressions are concatenated to define the search. If a user enters an account number, and it's valid, workflow search will only search for documents that have, anywhere in the document, an instance of a SourceAccountingLine business object with an account number field that matches or an instance of a TargetAccountingLine business object with an account number field that matches. Also note that a workflow function, "xstreamsafe", is called on both of the XPath queries; this function replaces proxy classes with their actual class names.

The fieldDef, display, and fieldEvaluation elements are all required, and the validation element is optional. Put all the elements together, and we've got a basic searchable XML rule attribute.

The Data Dictionary and Rule Attributes

Also see the KEW rule documentation and the delivered Java rule attributes, e.g. `KualiOrgReviewAttribute`.

KEW also lets us define rules with much the same ease, using org.kuali.workflow.attribute.KualiXmlRuleAttributeImpl. Let's take a look at an example of a rule attribute which uses KualiXmlRuleAttributeImpl, a rule attribute named "KualiAccountSubFundGroupAttribute".

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <data xmlns="ns:workflow" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="ns:workflow resource:WorkflowData">
3.    <ruleAttributes xmlns="ns:workflow/RuleAttribute"
xsi:schemaLocation="ns:workflow/RuleAttribute resource:RuleAttribute">
4.      <ruleAttribute>
5.        <name>KualiAccountSubFundGroupAttribute</name>
6.
<className>org.kuali.workflow.attribute.KualiXmlRuleAttributeImpl</className>
7.        <label>Account SubFund Group Routing</label>
8.        <description>Account SubFund Group Routing</description>
9.        <type>RuleXmlAttribute</type>
10.       <routingConfig>
11.         <fieldDef name="subFundGroupCode" title="kuali_dd_label(SubFundGroup)"
workflowType="ALL">
12.           <display>
13.             <type>text</type>
14.           </display>
15.           <validation required="true" />
16.           <quickfinder appliesTo="subFundGroupCode" draw="true"

service="workflow-org.kuali.module.chart.bo.SubFundGroup-Lookupable(workflow-subFundGr
oupCode:subFundGroupCode)" />
17.           <fieldEvaluation>
18.
<xpathexpression>wf:xstreamsafe('//newMaintainableObject/businessObject/subFundGroupCo
de') = wf:ruledata('subFundGroupCode')</xpathexpression>
19.           </fieldEvaluation>
20.         </fieldDef>
21.       </routingConfig>
22.     </ruleAttribute>
23.   </ruleAttributes>
24. </data>
```

This looks very much like how defining the search attributes looks. On the screen, it looks very similar as well.

Here, we've done a search for a rule. By choosing the KualiAccountSubFundTemplate rule template, which uses the KualiAccountSubFundGroupAttribute rule, we get an extra field: Sub-Fund Group Code. Entering a value into that allows us to find which document types use the rule template for which rules (in this case, the rule attribute is for the routing of AccountMaintenanceDocument documents, so that Accounts for certain sub fund groups are routed to the right people). Instead of a searchingConfig element, as we had in our searching attribute, we've got a routingConfig element in lines 10-21, however, our fieldDef in lines 11-20 look pretty much the same as they did in the search config.

We should note, though, that in the xpathexpression element on line 18, we use the XPath "=" to make sure that something in our document is matching the rule data. KEW will look through its rules and if it finds a rule for the AccountMaintenanceDocument doc type that matches the sub fund group in the document, it will route the document to the correct workgroup or person.

There is also an extension of KualiXmlRuleAttributeImpl, org.kuali.workflow.attribute.AccountLineReplacingXmlRuleAttribute, used for accounting documents. This class looks at the XPath for a rule extension, specifically for "kfs_sourceAccountingLines" and "kfs_targetAccountingLines". It then replaces those with whatever the class name is of the source or target accounting line class. An excerpt from KualiSubFundGroupAttribute shows its use:

```
<fieldEvaluation>
  <xpathexpression>
    wf:xstreamsafe('//kfs_sourceAccountingLineClass/account/subFundGroupCode') =
wf:ruledata('subFundGroupCode') or
    wf:xstreamsafe('//kfs_targetAccountingLineClass/account/subFundGroupCode') =
wf:ruledata('subFundGroupCode')
  </xpathexpression>
</fieldEvaluation>
```

If we're writing a document with accounting lines, chances are we want to use AccountingLineReplacingXmlRuleAttribute.

Adding Controls to the Attribute

There are two parts to this functionality: creating the XPath expression that forms the rule and creating the controls that KEW uses on the document search, rule search, or rule creation screens. On this document, we'll concentrate on the ways we can add a fields to the search or rule attribute. Of course, we've already seen how to do a plain old boring old text field...but where's the fun in that? Surely, the text field is the oatmeal or baked potato of the HTML control set; but can we use controls with a bit more punch? Let's take a look.

Adding a field with a lookup

One of the things our users are going to want in documents are the ability to click the good old magnifying glass icon and open a search screen. Is there a way to add quickfinders to our attribute fields? Thankfully, there is. When we look at the Credit Memo document search screen, we see that there's quickfinder text fields for Chart and Organization. It turns out that both of those fields are defined in a search attribute named "KualiPurApDocumentOrganization", so let's take a look.

```
1.  <ruleAttribute>
2.    <name>KualiPurApDocumentOrganization</name>
3.
<className>org.kuali.workflow.attribute.KualiXmlSearchableAttributeImpl</className>
4.    <label>Kuali Purchasing &amp;amp; Accounts Payable Document Organization</label>
5.    <description>Attribute to identify the Organization fields of a Purchasing &amp;amp;
Accounts Payable Document</description>
6.    <type>SearchableXmlAttribute</type>
7.    <searchingConfig>
8.      <fieldDef name="purapDocumentChartOfAccountsCode" title="kuali_dd_label(Org)">
9.        <display>
```

```
10.         <type>text</type>
11.       </display>
12.       <visibility>
13.         <column visible="false" />
14.       </visibility>
15.       <searchDefinition caseSensitive="false" />
16.       <quickfinder appliesTo="chartOfAccountsCode" draw="true"
service="workflow-org.kuali.module.chart.bo.Chart-Lookupable(null)" />
17.       <fieldEvaluation>
18.
<xpathexpression>wf:xstreamsafe('//document/chartOfAccountsCode')</xpathexpression>
19.       </fieldEvaluation>
20.     </fieldDef>
21.     <fieldDef name="purapDocumentOrganizationCode" title="kuali_dd_label(Org)">
22.       <display>
23.         <type>text</type>
24.       </display>
25.       <visibility>
26.         <column visible="false" />
27.       </visibility>
28.       <searchDefinition caseSensitive="false" />
29.       <quickfinder appliesTo="organizationCode" draw="true"
service="workflow-org.kuali.module.chart.bo.Org-Lookupable(workflow-organizationCode:o
rganizationCode,workflow-chartOfAccountsCode:chartOfAccountsCode)" />
30.       <fieldEvaluation>
31.
<xpathexpression>wf:xstreamsafe('//document/organizationCode')</xpathexpression>
32.       </fieldEvaluation>
33.     </fieldDef>
34.     <fieldDef name="purapDocumentOrgDisplayValue"
title="kuali_dd_label(PurApAttributeReferenceDummy)">
35.       <display>
36.         <type>text</type>
37.       </display>
38.       <visibility>
39.         <field visible="false" />
40.       </visibility>
41.       <fieldEvaluation>
42.
<xpathexpression>substring(concat(wf:xstreamsafe('//document/chartOfAccountsCode'),
concat('-', wf:xstreamsafe('//document/organizationCode'))),
number(not(boolean(concat(wf:xstreamsafe('//document/chartOfAccountsCode'),
wf:xstreamsafe('//document/organizationCode'))))*string-length(concat(wf:xstreamsafe(
'//document/chartOfAccountsCode'),
concat('-', wf:xstreamsafe('//document/organizationCode'))))+1)</xpathexpression>
43.       </fieldEvaluation>
44.     </fieldDef>
```

```
45.   </searchingConfig>
46. </ruleAttribute>
```

First of all, we notice that we can define as many fieldDef elements as we want within a searchingConfig. In this case, where the organization code lookup needs the related chart code lookup, we need to have multiple fieldDefs defined! Even though it's longer, this rule attribute bears a lot of resemblance to the KualiAccountSearchAttribute rule attribute; only the fieldDef elements have changed.

Let's take a look at lines 8 through 20, which is the fieldDef element for the chart code lookup. We still have a display element in lines 9-11; that's required. We're also using the "text" type still...even though we're adding a lookup, the type is still "text." We also have the fieldEvaluation element in lines 17-19. This fieldEvaluation defines an XPath expression to find the document's own chartOfAccountsCode element.

However, we've got some new elements, too. In lines 12-14, we've got a visibility element. In this example, we have column visibility set to false. This means that while we can use this column to search on, the column will not show up in the search result set - users would have to open the document to see the chart code in this field. The valid child elements of the visibility element include column, field - which determines if the field will show up in the search field while making sure that it shows up in the search results, and fieldAndColumn - which sets the visibility on both the search field and the result column at once. We can also do something like this:

```
  <visibility>
    <isMemberOfWorkgroup>SOME_WORKGROUP_NAME</isMemberOfWorkgroup>
  </visibility>
```

That would make the field visible only to those users who are members of the specified workgroup.

Line 15 defines the searchDefinition tag, which tells the search to not be case sensitive on this field; if we enter "bl" or "BL" or "bL" for that matter, we should still have the same chart looked up. We could also set if we want to allow wildcards in searches using the searchDefinition element, using the allowWildcards attribute. More features of the searchDefition tag will be reviewed in the section about date ranges.

Finally, on line 16, we've got the quickfinder element, which was the whole reason we looked at this rule attribute. Here it is again:

```
16. <quickfinder appliesTo="chartOfAccountsCode" draw="true"
service="workflow-org.kuali.module.chart.bo.Chart-Lookupable(null)" />
```

We see that the quickfinder element has three attributes:

- appliesTo - the name that will be set as the lookup's name; this allows us to refer to it later, such as in field conversions of other lookupables
- draw - true if the quick finder should be drawn, false otherwise
- service - the name of the lookup service

Workflow gets an instance of the proper lookup through a Spring bean, and the name of that bean in this case is "workflow-org.kuali.module.chart.bo.Chart-Lookupable" (we can find it's definition in KualiSpringBeansChart.xml). The "(null)" defines field conversions that need to be performed...we'll take a closer look at those in a moment.

Every quickfinder element needs a workflow lookupable bean defintion. This points to a shim implementation of the nervous system Lookupable interface that uses the nervous system to determine the appropriate lookup to hand off to, so that the user can use the same account lookup on the workflow screens as is available within KFS documents. You must also define a workflow lookupable bean for any lookups that are nested within the main lookup that you are trying to put on the rule or document search screen. Here is an example of a lookupable bean definition...

```
    <bean id="workflow-org.kuali.module.chart.bo.ObjectType-Lookupable"
parent="workflowLookupable" singleton="false">
        <property name="businessObjectClassName">
            <value>org.kuali.module.chart.bo.ObjectType</value>
        </property>
    </bean>
```

Thankfully, most of the common chart lookups are already implemented, including:

workflow-org.kuali.module.chart.bo.Campus-Lookupable	workflow-org.kuali.module.chart.bo.Chart-Lookupable	workflow-org.kuali.module.chart.bo.Account-Lookupable
workflow-org.kuali.module.chart.bo.SubAccount-Lookupable	workflow-org.kuali.module.chart.bo.ProjectCode-Lookupable	workflow-org.kuali.module.chart.bo.ObjectCode-Lookupable
workflow-org.kuali.module.chart.bo.ObjectCodeCurrent-Lookupable	workflow-org.kuali.module.chart.bo.ObjLevel-Lookupable	workflow-org.kuali.module.chart.bo.ObjectType-Lookupable
workflow-org.kuali.module.chart.bo.ObjSubTyp-Lookupable	workflow-org.kuali.module.chart.bo.FundGroup-Lookupable	workflow-org.kuali.module.chart.bo.SubFundGroup-Lookupable
workflow-org.kuali.module.chart.bo.SubFundGroupType-Lookupable	workflow-org.kuali.module.chart.bo.Org-Lookupable	workflow-org.kuali.module.chart.bo.ResponsibilityCenter-Lookupable
workflow-org.kuali.module.chart.bo.OrgType-Lookupable	workflow-org.kuali.module.chart.bo.AcctType-Lookupable	workflow-org.kuali.module.chart.bo.HigherEdFunction-Lookupable

Here is an example

That just leaves us conversion codes, which the quickfinder for the Organization Code lookup gives us (line 29 but here it is again):

```
29. <quickfinder appliesTo="organizationCode" draw="true"
service="workflow-org.kuali.module.chart.bo.Org-Lookupable(workflow-organizationCode:o
rganizationCode,workflow-chartOfAccountsCode:chartOfAccountsCode)" />
```

Here, the quickfinder element uses the workflow-org.kuali.module.chart.bo.Org-Lookupable service, and then passes it the field conversions it needs. Field conversions are in the form of: workflow-<lookup_field_name>:<applies_to_value> separated by commas. In this example, then, we've got two field conversions: the lookup's organizationCode applies to the organizationCode field that will be the value the fieldEvaluation XPath expression matches against. Likewise, the chartOfAccountsCode, if filled in by the previous field, populates the chartOfAccountsCode field that will appear once the user hits the quickfinder icon.

Do note that even if there aren't going to be field conversions for the service, we still need to say (null) as the field conversion list.

Select boxes, radio buttons, and check boxes

Sometimes, we know that the number of values that a certain field could possible have are limited, and therefore, it would be nicer to display a select/drop down box for that field. The Credit Memo document search has a such a drop down box for the Process Campus field. That field is defined by a rule attribute named "KualiCreditMemoDocumentProcessCampus". Let's take a look:

```
1.  <ruleAttribute>
2.    <name>KualiCreditMemoDocumentProcessCampus</name>
3.
<className>org.kuali.workflow.attribute.KualiXmlSearchableAttributeImpl</className>
4.    <label>Kuali Credit Memo Document Processing Campus</label>
5.    <description>Attribute to identify the Processing Campus of the Credit Memo
Document</description>
6.    <type>SearchableXmlAttribute</type>
7.    <searchingConfig>
8.      <fieldDef name="creditMemoProcessCampus"
title="kuali_dd_label(CreditMemoDocument)">
9.        <display>
10.         <type>select</type>
11.
<values>kuali_values_finder_class(org.kuali.core.lookup.keyvalues.CampusValuesFinder)<
/values>
12.       </display>
13.       <visibility>
14.         <column visible="false" />
15.       </visibility>
16.       <quickfinder appliesTo="campusCode" draw="true"
service="workflow-org.kuali.core.bo.Campus-Lookupable(null)" />
17.       <fieldEvaluation>
18.
<xpathexpression>wf:xstreamsafe('//document/processingCampusCode')</xpathexpression>
19.       </fieldEvaluation>
20.     </fieldDef>
21.   </searchingConfig>
22. </ruleAttribute>
```

We're getting pretty used to the template of these rule attributes by now. Practically everything we've seen before: the visibility element, the fieldEvaluation XPath expression, the quickfinder which uses the workflow-org.kuali.core.bo.Campus-Lookupable and has no field conversions. The only thing different is the type of the single fieldDef, defined on lines 9-12. Here we've got a "select" field type. That's all well and good; we expect that this will use a select drop down control in the HTML that's rendered instead of a text field.

The twist is, of course, what values to populate that select box with. Thankfully, that's very easy. We simply define a <values> element, and in that use the kuali_values_finder_class() to send in the implementation of org.kuali.core.lookup.keyvalues.KeyValuesFinder implementation we want to use to populate the drop down field.

Let's say, though, that we've only got a couple of values, and therefore, instead of showing a select box, we'd prefer a radio button. And let's say that furthermore we don't already have a predefined KeyValuesFinder implementation...

Credit Memo has a radio button example, too, in the "Search Result Type" field. That's defined in a rule attribute named "KualiPurApSearchResultsDisplayType".

```
1.  <ruleAttribute>
2.    <name>KualiPurApSearchResultsDisplayType</name>
3.
<className>org.kuali.workflow.attribute.KualiXmlSearchableAttributeImpl</className>
4.    <label>Kuali Purchasing &amp;amp; Accounts Payable Search Result Display
Type</label>
5.    <description>Attribute to identify which type of search results should be
returned for a Purchasing or Accounts Payable document</description>
6.    <type>SearchableXmlAttribute</type>
7.    <searchingConfig>
8.      <fieldDef name="displayType"
title="kuali_dd_label(KFSAttributeReferenceDummy)">
9.        <display>
10.         <type>radio</type>
11.         <values selected="true" title="Document Specific
Data">document_info</values>
12.         <values title="Workflow Data">workflow_standard</values>
13.       </display>
14.       <visibility>
15.         <column visible="false" />
16.       </visibility>
17.     </fieldDef>
18.   </searchingConfig>
19. </ruleAttribute>
```

Let's take a look at lines 9-13, which define the display type of the fieldDef defined in this rule attribute. On line 13, we see the type of this fieldDef is "radio"...that seems pretty simple. On lines 11 and 12, though, we see that the values element is defined differently: we have two of them, one for each radio button. These value definitions work with select and multibox as well. Note that these value boxes work the opposite of HTML option elements: here the label that is displayed to the user is inside the "title" attribute of the values tag, while the child of the tag is the actual value that will eventually be searched against. Also we see that we can select one of the radio button values by default, using the "selected" attribute.

Select boxes or radio buttons let us choose one value to search against, but what if we want to search against several values in a set? Credit Memo just happens to have one of those fields, as well: Credit Memo Status, defined by a rule attribute named "KualiCreditMemoDocumentStatusCode". Let's take a look:

```
1.  <ruleAttribute>
2.    <name>KualiCreditMemoDocumentStatusCode</name>
3.
<className>org.kuali.workflow.attribute.KualiXmlSearchableAttributeImpl</className>
4.    <label>Kuali Credit Memo Document Status Code</label>
5.    <description>Attribute to identify the Credit Memo Status of the Credit Memo
Document</description>
6.    <type>SearchableXmlAttribute</type>
7.    <searchingConfig>
8.      <fieldDef name="creditMemoStatusCode"
title="kuali_dd_label(CreditMemoDocument)">
9.        <display>
10.         <type>multibox</type>
11.
<values>kuali_values_finder_class(org.kuali.module.purap.lookup.keyvalues.CreditMemoSt
atusValuesFinder)</values>
12.       </display>
13.       <visibility>
14.         <column visible="false" />
15.       </visibility>
16.       <fieldEvaluation>
17.         <xpathexpression>wf:xstreamsafe('//document/statusCode')</xpathexpression>
18.       </fieldEvaluation>
19.     </fieldDef>
20.   </searchingConfig>
21. </ruleAttribute>
```

Here, we see the yet another field type, "multibox", on line 10. But, we note that this is just a variation of select drop downs and radio buttons, in that we simply send in a set of possible values with the "values" element; in this case, we're using the kuali_values_finder_class() function to send in a KeyValuesFinder implementation. If one of the Credit Memo Status Code checkboxes are chosen, then all documents selected by the search must have that status code. If multiple checkboxes are chosen, then the status code of returned documents have to be within that set.

Adding date ranges

One final field type deserves our consideration because they're so common, and if you've read the header for this section, you know already that we're talking about date range field definitions. Incredibly, Credit Memo uses a date range for the Credit Memo date field. That's defined in a rule attribute named "KualiCreditMemoDocumentVendorCreditMemoDate":

```
1.  <ruleAttribute>
2.    <name>KualiCreditMemoDocumentVendorCreditMemoDate</name>
3.
<className>org.kuali.workflow.attribute.KualiXmlSearchableAttributeImpl</className>
4.    <label>Kuali Credit Memo Document Vendor Credit Memo Date Range</label>
5.    <description>Attribute to identify the Vendor Credit Memo Date Range of the
Credit Memo Document</description>
6.    <type>SearchableXmlAttribute</type>
7.    <searchingConfig>
8.      <fieldDef name="creditMemoVendorCreditMemoDate"
title="kuali_dd_label(CreditMemoDocument)">
9.        <display>
10.         <type>text</type>
11.       </display>
12.       <visibility>
13.         <column visible="false" />
14.       </visibility>
15.       <searchDefinition dataType="datetime" rangeSearch="true" />
16.       <fieldEvaluation>
17.
<xpathexpression>wf:xstreamsafe('//document/creditMemoDate')</xpathexpression>
18.       </fieldEvaluation>
19.     </fieldDef>
20.   </searchingConfig>
21. </ruleAttribute>
```

Old hat by now, right? Therefore, let's just take a look at the display definition, on lines 9-11. Here we see, just as we expect, that the display type for this fieldDef is date and...wait...no, it isn't. It's text. Well...in KFS, we do enter dates through text fields. But we have the date lookup icons too...

Once again, the searchDefinition tag comes to our rescue. Here, on line 15, we see that this tag has two attributes, one that changes the dataType of the text field to "datetime". When workflow parses this, it adds the date lookup icon to the text field. Then, we see an attribute named rangeSearch. Obviously, this means that we're searching within a range of dates. We're not limited to dates for range searches, by the way. The rule attribute "KualiDocumentHeaderTotalAmountRange" defines a float range to search through, just by setting dataType="float".

Unable to render {include} The included page could not be found.