ADF Code Corner

043. How-to integrate remote task flows in your ADF applications (POJO DC Example)

twitter.com/adfcodecorner

Abstract:

ADFc remote task flows are bounded task flows that are deployed as Java EE applications to a server for clients to access. Remote task flows use JSPX documents for their views and are open for direct browser access. Like bounded task flows in ADF libraries, remote task flows are a design pattern for sharing business logic. However, using remote task flows, you don't import the bounded task flow sources to the application project but access the remote task flow instance directly using a task flow call activity. In a way remote bounded task flow are comparable to Web Services, which are also hosted on remote servers to provide business functionality to a calling application. The major difference between remote task flows and Web Services, beside of messaging and discoverablility, is that remote task flows have their own user interface that users work with. But how do you build, deploy and call such remote flows within your application development project? The answer to this question is in the focus of this blog article.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
13-JUL-2010

Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

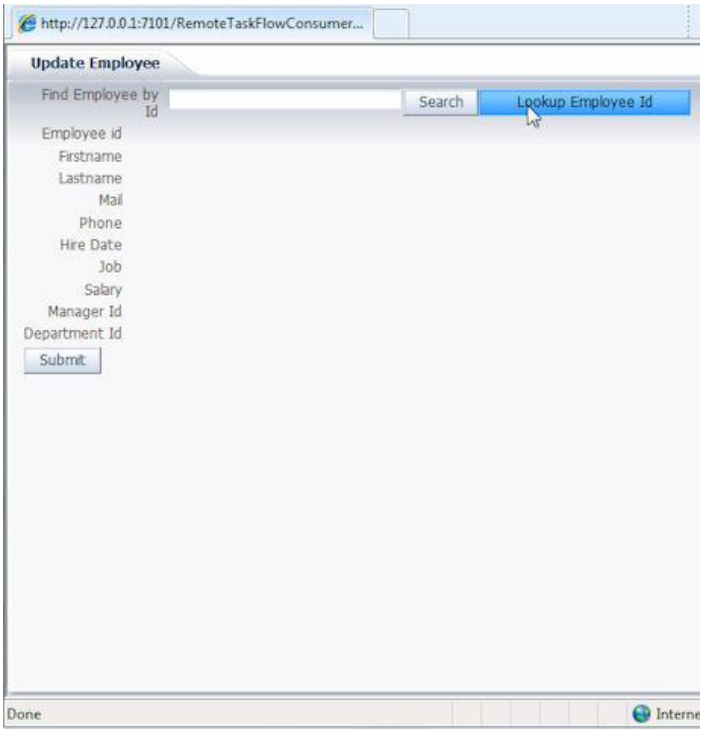
Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

Introduction

The use case in this blog article has an employee search form in the client application that takes the employeeId as an input parameter to query the employee for update.

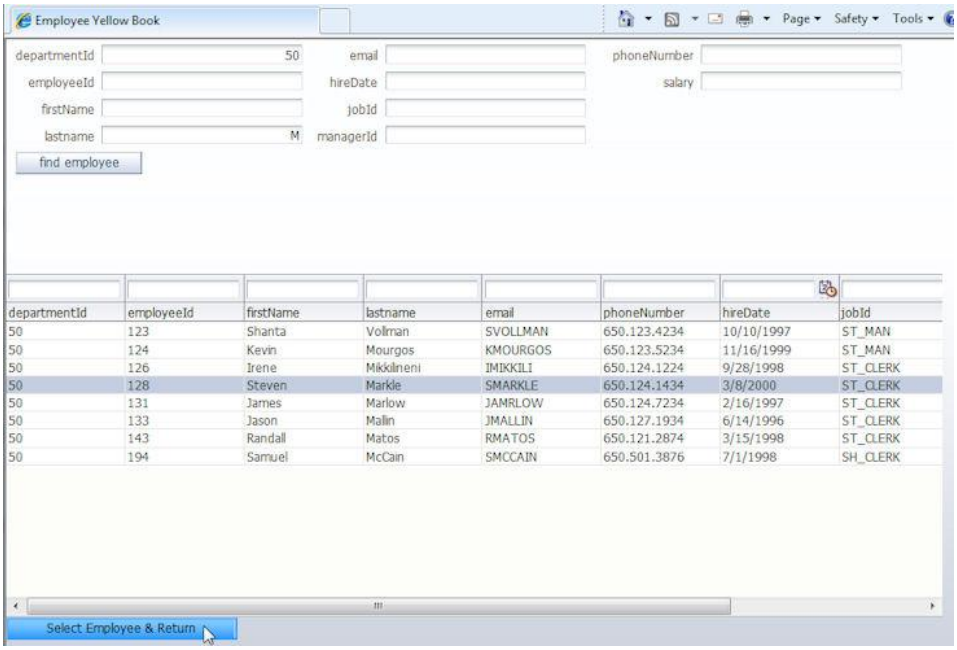
For users that don't know the employeeId to lookup, a "yellow pages"-like remote task flow can be called that provides advanced search functionality. The "yellow pages" task flow used for this article is simple in the complexity - which usually is good for demos - but may also process complex routing before returning a selected value back to the calling application.



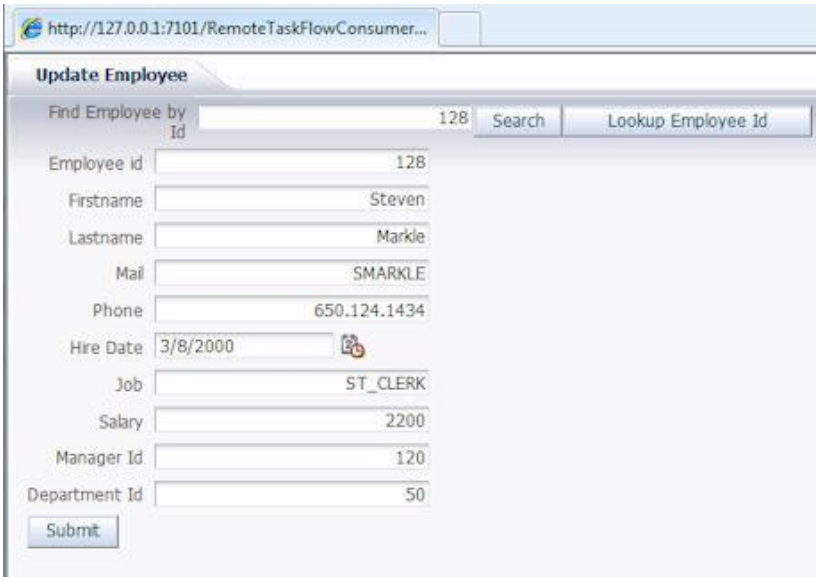
The call of a remote task flow direct navigation to a different Java EE application, which means that users possibly needs to be re-authenticated when accessing this service. In the sample for this article, the screen below is owned by the remote task flow and replaces the application screen shown to the time the user requested the remote task flow. The remote task flow request

URL contains return information about the calling application and a controller state token that is used to re-create the state of the calling application to the time the remote task flow happened.

Note: The example uses the POJO Data Control in the remote task flow search form.



Using an output parameter and a return activity on the remote task flow definition, the selected value of the "yellow page" sample task flow is returned to the calling application, where it is stored in a memory scope attribute. To use the selected value when querying the "find employee" form, the returned value is referenced in the ADF binding PageDef file, which is covered later in this article.



Note: Though the example uses the ADF POJO DataControl feature, remote task flows work the same using ADF Business Components or EJB

Calling Application Navigation Flow

The navigation flow of the calling application is that a task flow call activity is processed when the user clicks the "Lookup Employee Id" command button. The task flow call activity is used to navigate from a calling task flow, which can be a bounded or unbounded flow, to a target flow, which must be a bounded flow. Usually, task flow call activities are used to navigate between task flows defined for an application. With a little change though, the task flow call can be to an externally deployed flow. For the configuration below, the called task flow was deployed to the integrated WLS server using an EAR file. The task flow thus is not part of the application. The task flow call activity configuration for calling the remote task flow is shown next

```
<task-flow-call id="CallRemoteTaskFlow">
   <task-flow-reference>
     <document>WEB-INF/yellow-pages-flow.xml</document>
     <id>yellow-pages-flow</id>
   </task-flow-reference>
   <return-value id="__11">
     <name id="__13">employeeId</name>
     <value id="__12">#{pageFlowScope.employeeId}</value>
   </return-value>
   <remote-app-
url>#{initParam['adf.sample.remote.taskflow.url']}</remote-app-url>
 </task-flow-call>
```

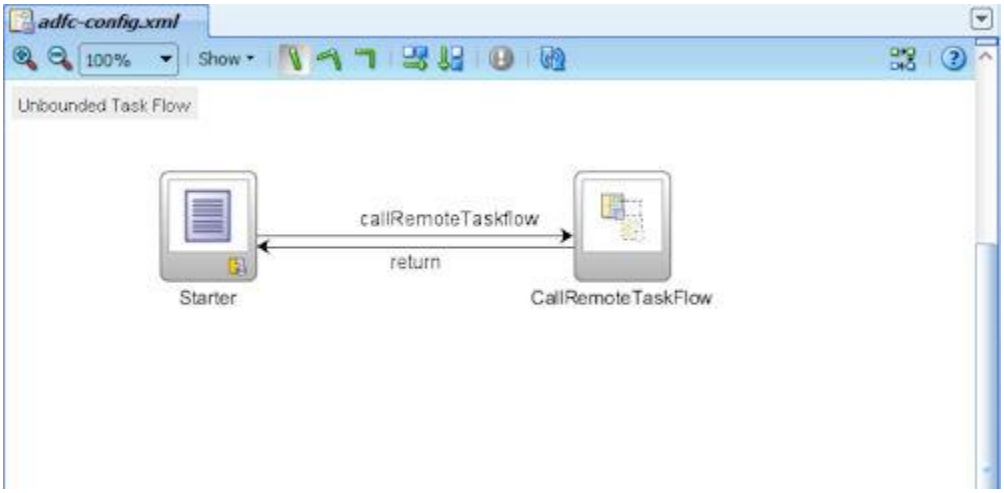
The task flow document to call is referenced from the WEB-INF directory on. In the example remote task flow, the document name is WEB-INF/yellow-pages-flow.xml. As there does not exist facilities to dynamically discover remote task flow documents, this information need to be known by the developer of the consuming application. The task flow Id to call is "yellow-pages-flow".

Until here, there is no difference between the configuration of a local task flow call and a remote call. The difference however is in the use of the remote-app-url element in the configuration. Here, expression language must be used to reference a managed bean, memory attribute or - like in the example above - a resource definition. Assuming you don't like the remote task flow access to be hard coded in the application, like in a managed bean property, this example shows how to define the remote task flow URL in a context parameter of the web.xml file. The calling application's web.xml file has the following entry, for the EL reference initParam['adf.sample.remote.taskflow.url'] to work.

```
 </context-param>
   <context-param>
   <param-name>adf.sample.remote.taskflow.url</param-name>
   <param-value>
     http://127.0.0.1:7101/RemoteTaskFlow-ViewLayerProject-context-
     root/faces/adf.task-flow
   </param-value>
 </context-param>
```

The parameter name of this context parameter is what developers can freely choose. The parameter value though must be the absolute URL to access the remote task flow, including the **adf.task-flow** servlet reference. Also note that the **/faces/** path must be within the request URL to ensure the request is a JSF request.

Also shown in the first configuration code is the return value that is defined on the task flow call activity. The name of the return value matches the output parameter name of the called remote task flow, which also is an information application developers need to have by hand as there is no remote task flow discovery mechanism available. The returned value is stored in the calling task flow's pageFlowScope, a HashMap in memory.



Hint: In Oracle JDeveloper 11.1.1.3, the remote task flow document Id cannot be edited in the Property Inspector, which is a known issue in that release. So when configuring the remote task flow definition, add the <id> value in the source code view of the task flow definition.

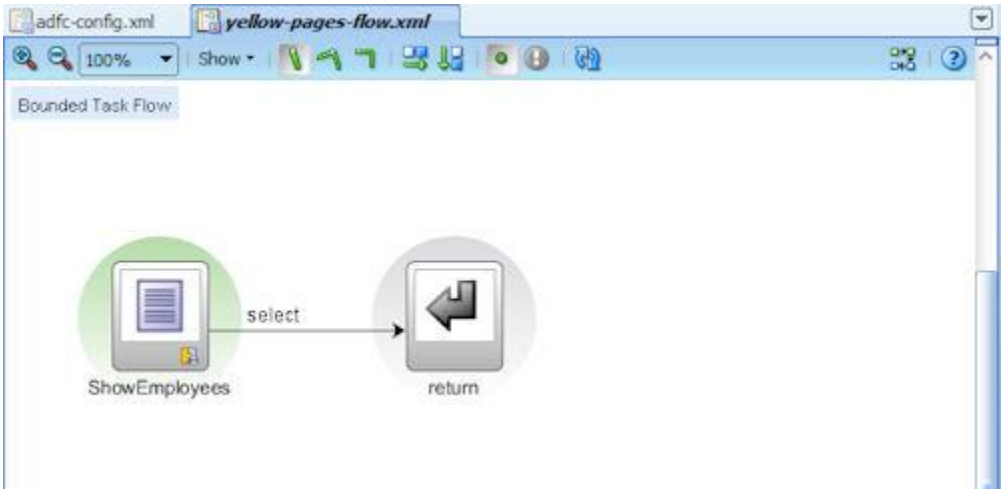
Building the Remote Task Flow

There is not much to consider when designing the remote task flow to access

- The "Visibility | URL Invoke" property displayed in the Property Inspector must be set to "url-invoke-allowed" as otherwise a GET request to the task flow is disallowed
- Define return value definitions for the values that should be passed back to the calling application.

```
<return-value-definition id="__11">
   <name id="__12">employeeId</name>
   <value>#{pageFlowScope.outcome}</value>
   <class>java.lang.Long</class>
</return-value-definition>
```

The "employeeId" name of the return value is also configured on the task flow call activity that references this task flow. Similar, if the remote task flow requires input parameters, you create them in the bounded task flow and reference them from the task flow call activity. (Please see chapter 15 of the Oracle ADF developer guide to learn more about task flow call activities).



To close the remote task flow and return the selected value, the "Select Employee & Return" button uses a setPropertyListener to store the selected employee value in the memory scope referenced from the employeeId task flow return value definition:

```
<af:commandButton text="Select Employee &amp; Return" id="cb2"
                  action="select">
  <af:setPropertyListener
from="#{bindings.currentSeletedEmployeeId.inputValue}"
                          to="#{pageFlowScope.outcome}"
                          type="action"/>
</af:commandButton>
```

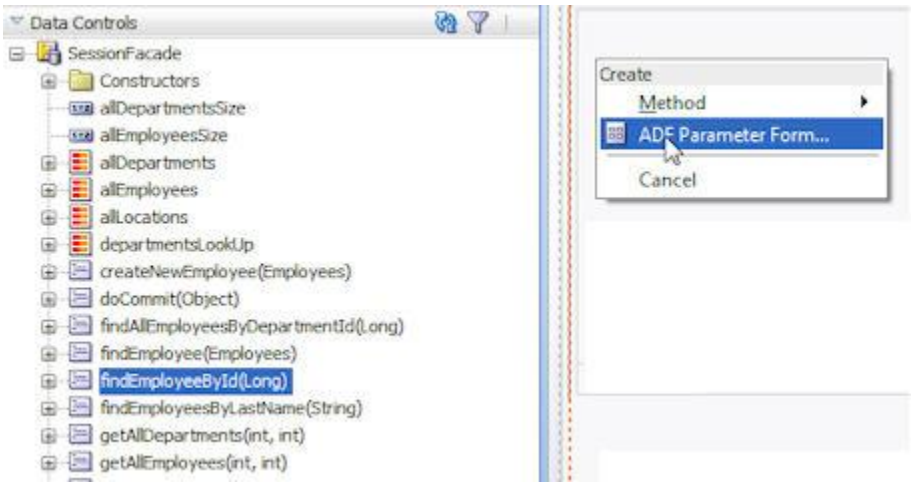
Calling a Remote Task Flow

As explained earlier, the remote task flow is called from task flow call activity that is configured for remote task flow access. The return value of the bounded task flow is configured on the task flow call activity so that the returned value, which can be a literal value or object, is stored in a memory scope or managed bean accessible to the calling application.

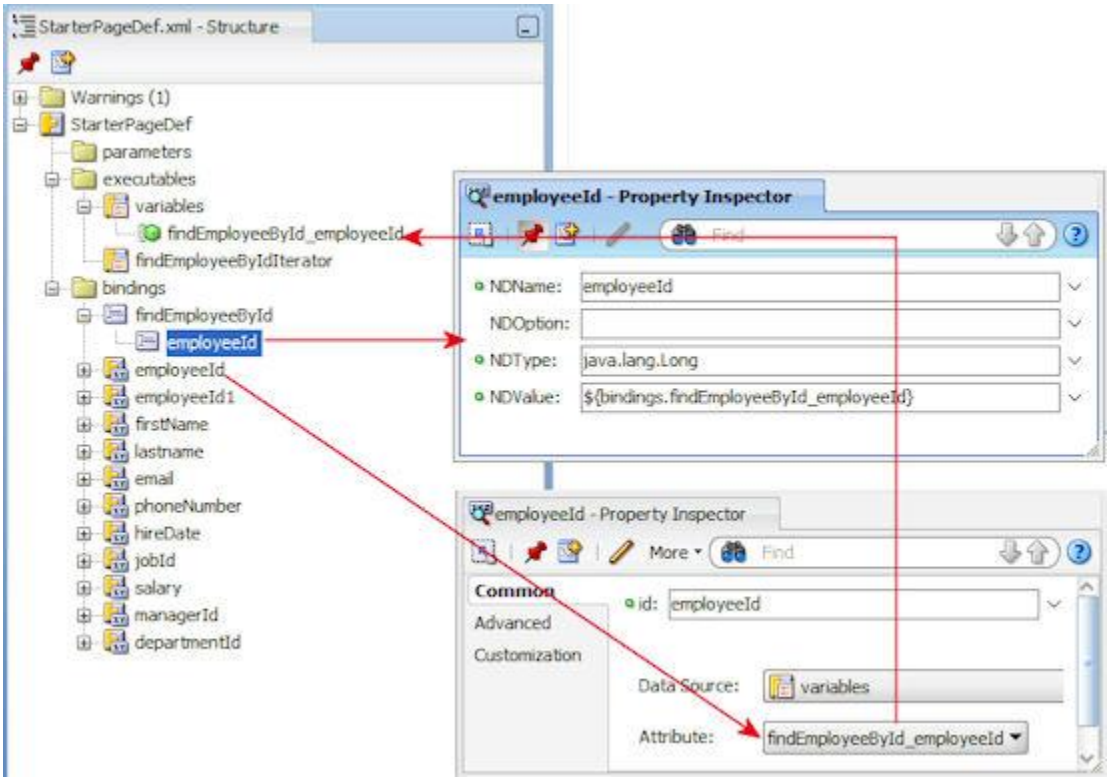
```
<return-value id="__11">
  <name id="__13">employeeId</name>
  <value id="__12">#{pageFlowScope.employeeId}</value>
</return-value>
```

Note: bounded task flows may return more than a single value, in which case the call activity must be configured to handle multiple

The find employee form of the calling application is built by dragging and dropping a search method as a parameter form to the page



In the example, the parameter form has a single input text field for the user to input the employee Id of the employee to search for. As shown in the image below, the input field is bound to an attribute binding "employeeId" in the PageDef file. The attribute binding is created by ADF when the finder method is dragged onto the page and dropped as a parameter form. It reads and writes its value from a variable - a value holder - in the PageDef file. In addition to the input file, the parameter form has a command button added that executes the query. This command button is bound to the "findEmployeeById" method. The method argument "employeeId" references the variable as shown in the image below.



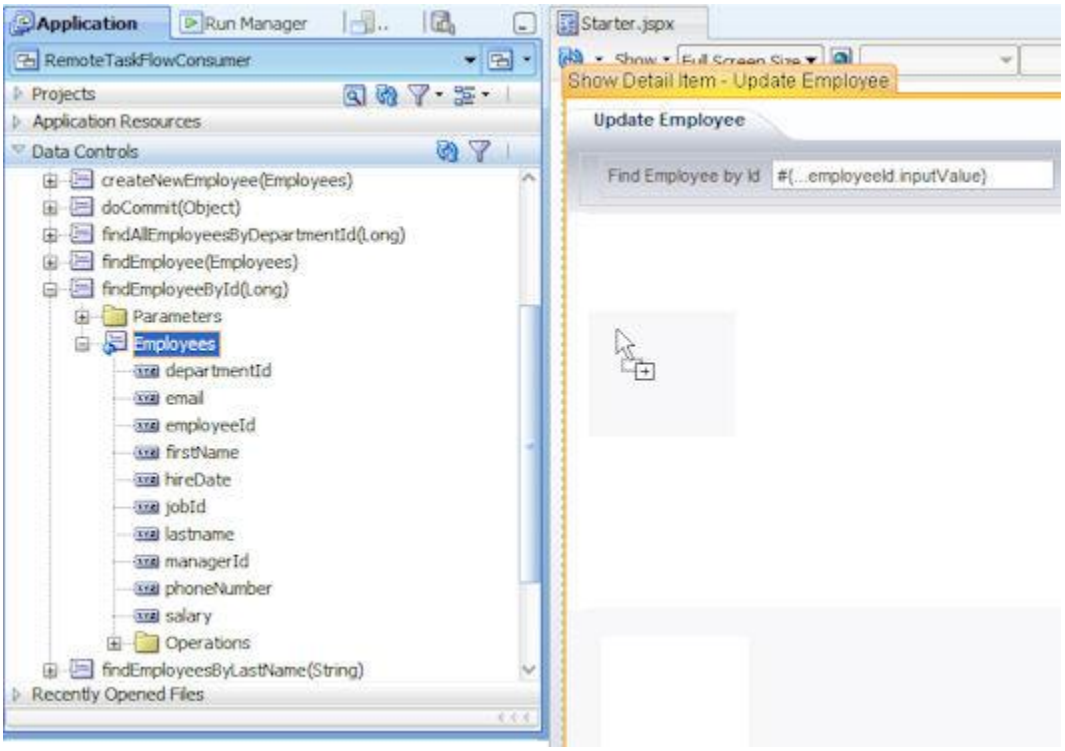
A second button, "Lookup Employee Id", was added manually with a navigation reference to the task flow call activity

```
<af:commandButton text="Lookup Employee Id" id="cb1"
                  action="callRemoteTaskflow"/>
```

Pressing the button navigates to the remote task flow. The remote task flow, upon selecting an employee, writes the employeeId to the return value before navigating to the return activity, which issues an outcome of "return". The "return" outcome must have a control flow case equivalent in the calling application, pointing from the task flow call activity to the search page to display the results. But, hold on! How does the return value get added to the search field so the query now displays the employee data? The secret is in the variable that the search form field is bound to. The variable allows developers to specify a default value, which can be read using EL.

```
<executables>
 <variableIterator id="variables">
 <variable Type="java.lang.Long" Name="findEmployeeById_employeeId"
           IsQueriable="false"
DefaultValue="#{pageFlowScope.employeeId}"/>
</variableIterator>
...
```

The **DefaultValue** attribute references the memory attribute that holds the employee Id returned from the remote task flow. To display the employee data, you drag and drop the return collection "Employees" of the "findEmployeeById" as a form



Note: If you study the request URL referencing the remote task flow at runtime, you notice a return URL to the calling page and a task flow state token added to the request. The state token ensures that the state of the calling page is recovered upon task flow return, which also includes memory scopes.

Download Sample

The sample application consists of two applications: a remote task flow and the calling application. Both applications use a POJO model to query data. The POJO does not require a database connection for you to set up. To run the sample from JDeveloper, open both Applications, the Remote Task flow and the Task Flow Consumer application in Oracle JDeveloper. Start the integrated WLS server using the Oracle JDeveloper Run | Start Server Instance menu option . Then choose Application | Deploy yellow-pages and deploy the EAR file to the integrated server. Using the integrated WLS server also means that the web URL of the deployed remote task flow is on the local host so that there is no need to change the web.xml context parameter of the calling application.

To run the sample, open the RemoteTaskFlowConsumer application and select the "Starter.jspx" file. Choose "run" from the context menu and, when the application shows up, press the "Lookup Employee Id" button. **Download Sample from ADF Code Corner.**

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

RELATED DOCOMENTATION

☒	Fusion Developer Guide Product Documentation - http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/taskflows_activities.htm#CHDJDJEF
☒	
☒	