Oracle FLEXCUBE BPMN Process Flow Definition Guide

Oracle FLEXCUBE Investor Servicing

Release 12.0.4.7.11

[December] [2017]

ORACLE

FINANCIAL SERVICES

Table of Contents

1. Introduction

1.1 Background

This document provides a brief idea about the BPMN components and to create new BPMN process flows using the BPMN components.

1.2 Difference between BPEL and BPMN

BPEL is an XML-based language for describing a business process in which most of the tasks represent interactions between the process and external Web services. The BPEL process itself is represented as a Web service, and is realized by a BPEL engine which executes the process description. BPMN is a standard set of diagramming conventions for describing business processes. It is designed to visualize a rich set of process flow semantics within a process and the communication between independent processes. It is intended to support capture of sufficient detail to allow it to be the source of an executable process description.

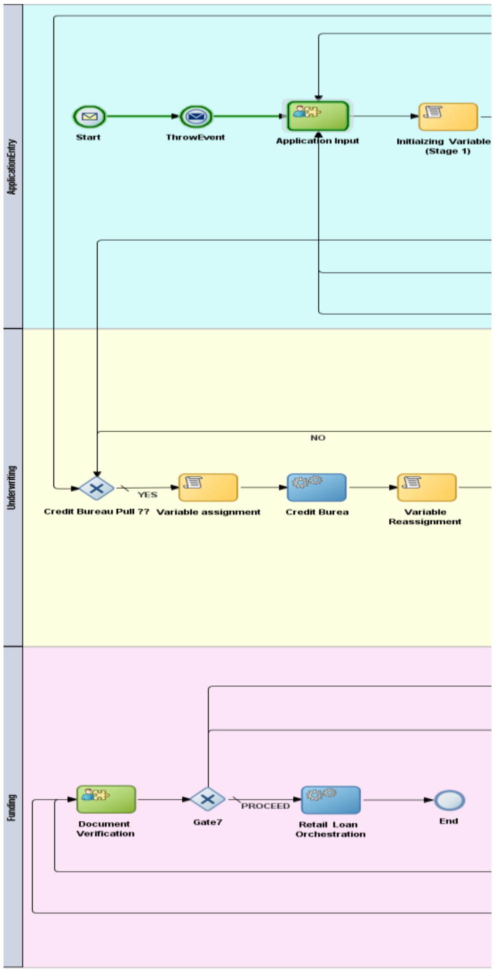
1.3 Advantages of BPMN over BPEL

- Simple and Understandable methodology.
- BPMN provides a standard notation for modelling Business Processes that can be used and understood by all types of Business Professionals, including the business analysts who create the processes, the technical developers who create the technology that will carry out those processes and the business managers who oversee those processes.
- Business Users will be able to easily read and understand a BPMN business process diagram
- Tasks can be linked in any form (similar to “goto” in programming languages) Where as BPEL imposes more restrictions (closer to a “real” programming language).
- Structures work that needs to be done, whether this be automated or manually.
- Resembles more like a Flow Diagram (i.e) What You See is What You Get.

1.4 BPMN Components

Swimlanes

Swimlanes are used for grouping flow objects based on the roles defined within your process.

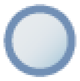


The None Start Event:

None events are always used to define the beginning of sub-processes.

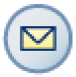
The none start event cannot have incoming sequence flows. It can only have default out-going sequence flows.

The None End Event



The none end event is always used to mark the end of a sub-process and event sub-process.

The Message Start Event

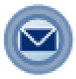


The message start event cannot have incoming sequence flows. Message start events require a default outgoing sequence flow.

To expose a process as a service, your process must begin with a message start event.

The message start event responds to a message sent to a specific process.

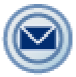
The Message End Event



The message end event is used to send a message to another process or service when the process is completed.

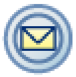
The message end event is always used with either a message start event or message catch event.

The Message Throw Event



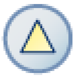
The message throw event enables you to send a message to another process or service.

The Message Catch Event



The message catch event is frequently used with the message throw event to communicate with another BPMN process.

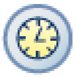
The Signal Start Event



The signal start event is a response to a signal broadcast to multiple processes.

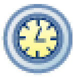
Signals can be broadcast from a BPMN process using the signal throw event. Using a combination of signal throw and signal start events, you can invoke multiple processes simultaneously.

The Timer Start Event



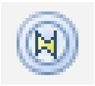
The timer start event triggers the creation of a process instance based on a specific time condition.

The Timer Catch Event



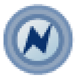
You can use timer event as boundary events on an activity. Timer events can be defined as either interrupting or non-interrupting boundary events.

The Error Catch Event



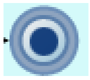
When a service or process fails with an error, the error catch event triggered. Similar to a catch in BPEL.

The Error End Event



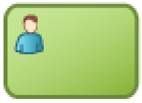
The error end event “throws” an error, which can be captured in another part of the process.

The Terminate Event



The terminate end event is used to immediately terminate a process.

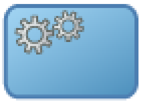
Interactive Activities

 User	Process participants interact with your business application using User Tasks.
 Complex	Uses a complex routing flow that is defined within the Human Task.
 FYI	Bases assignment on the participant, role, or group defined in the swimlane. Similar to the user interactive activity, but the FYI activity does not wait until completion before continuing.
 Group	Uses the group vote pattern. The assignee for the is automatically set to the role/group associated with the Lane. This interactive activity can only be added to swimlanes that are assigned to roles or groups.
 Initiator	The initiator pattern is used to create a process instance.
 Management	Uses the management chain pattern where the assignee is set to the management chain pattern for the process participant belonging to the group or role assigned to the swim lane.

The Manual Task

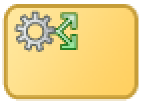
The manual task does not allow you to manipulate data objects. Data objects associated with the previous flow element are passed through as-is to the next flow element.

The Service Task



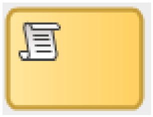
The service task enables you to communicate with other processes and services. Process analysts can add the service task when they know that a process must invoke an external service or process.

The Business Rule Task



Business rules are statements that describe business policies or describe key business decisions.

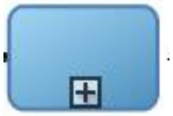
The Script Task



The script task is used to change values of data objects within your process.

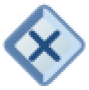
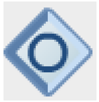
It is often used to set initial values of data objects at the beginning of a process.

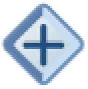
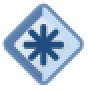
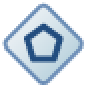
The Subprocesses



Subprocesses are contained as part of the parent subprocess. Subprocesses must begin with a start none event and must end with a none end event.

Gateways

 Exclusive gateways	The exclusive gateway enables you to split your process into two or more paths. However, the process only continues down one of these paths even if multiple outgoing sequence flows are present. Exclusive gateways can have conditional outgoing sequence flows and must have at least one default outgoing sequence flow.
	The inclusive gateway enables you to split your process into two or more paths. Unlike the exclusive gateway, however, a token may flow down one or more of these paths depending on how the outgoing conditional sequence flows are evaluated.

Inclusive gateways	
 Parallel gateways	The parallel gateway enables you to split your process into two or more paths when you want your process flow to follow all paths simultaneously. The parallel gateway is useful where your process must perform multiple tasks in parallel.
 Complex gateways	The complex gateway splits a process similar to an inclusive gateway. However, it enables you to define a condition that determines if the instance can continue even if not all of the tokens have arrived at the complex gateway merge.
 Event-based gateways	The event based gateway is different than other gateways in that decisions about process flow are based on an event rather than data-specific conditions.

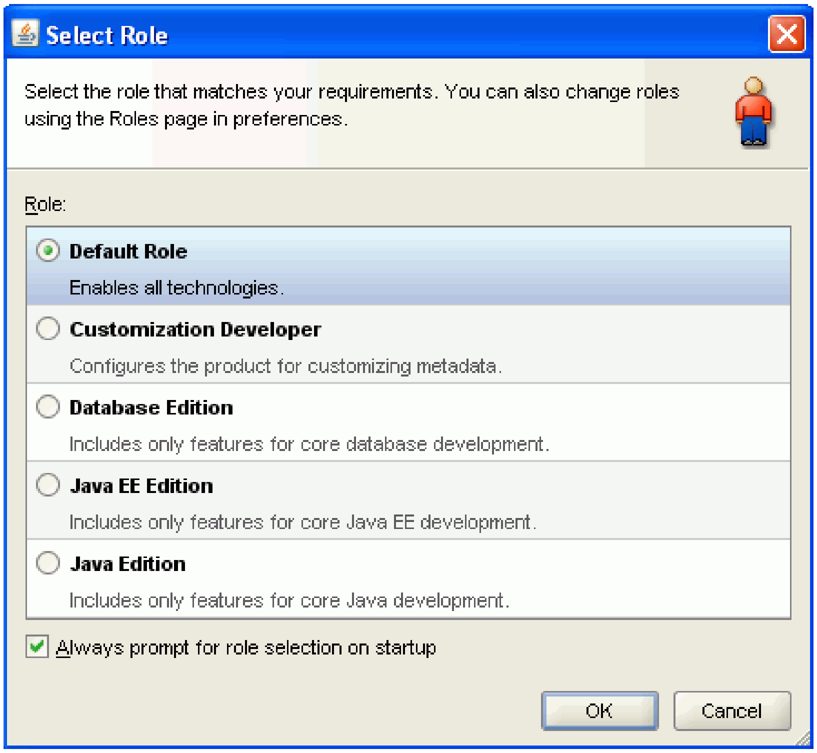
For more Information regarding BPMN Components Refer the Link given in the ***References****.*

1.5 Creating new BPMN Process

Follow the below steps to create a new process using the BPMN

1.5.1 Creating a BPMN Process Model

Step 1: Open JDeveloper Studio from the Windows Start menu. When prompted to select a role, choose the **Default Role**. Click OK.



Select Role

Select the role that matches your requirements. You can also change roles using the Roles page in preferences.

Role:

● **Default Role**
Enables all technologies.

○ **Customization Developer**
Configures the product for customizing metadata.

○ **Database Edition**
Includes only features for core database development.

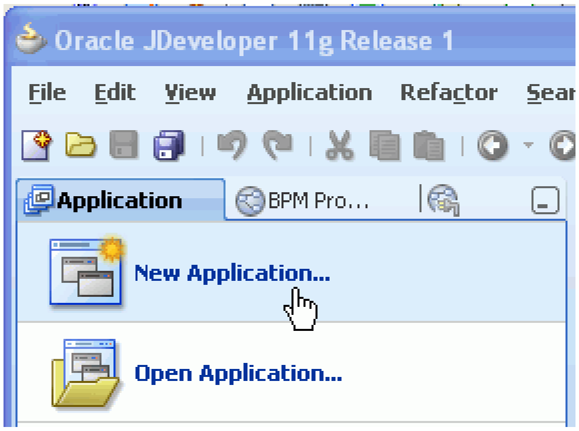
○ **Java EE Edition**
Includes only features for core Java EE development.

○ **Java Edition**
Includes only features for core Java development.

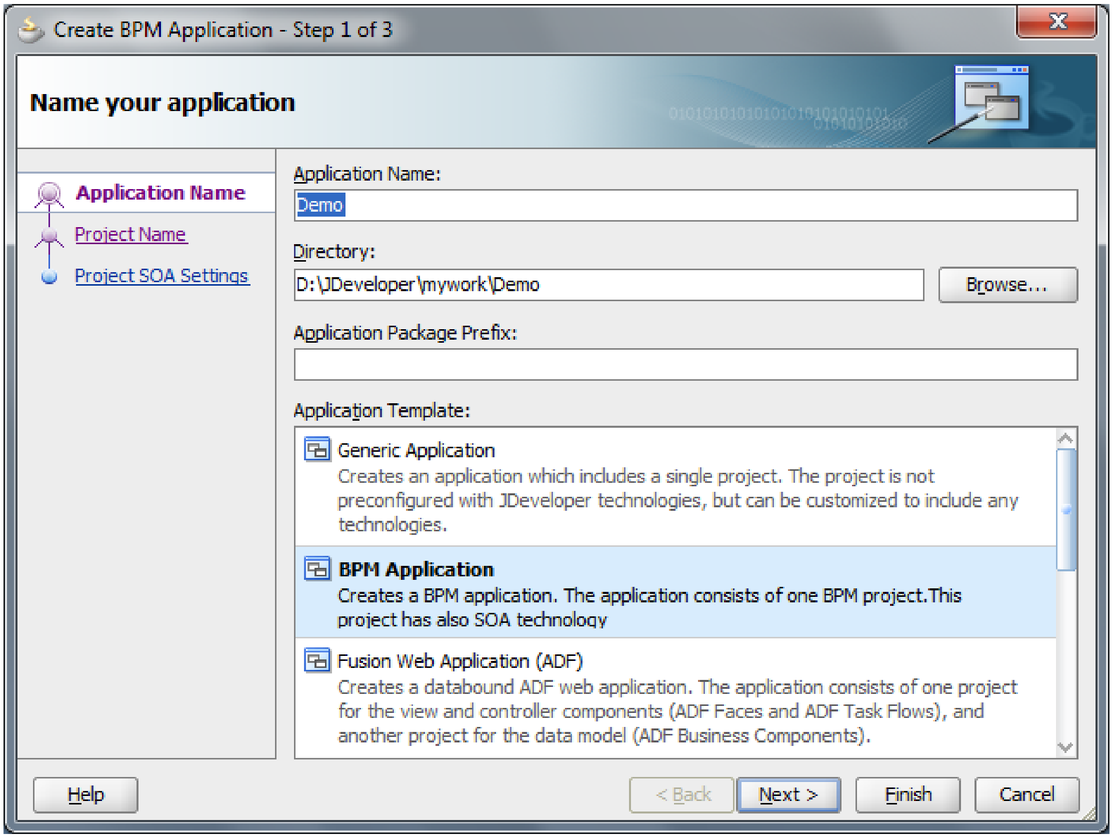
☑ Always prompt for role selection on startup

Close the Daily Tips window.

Step 2: Create a new application. Click the **New Application** bar in the left panel.


Oracle JDeveloper 11g Release 1
File Edit View Application Refactor Sear
Application
BPM Pro...
New Application...
Open Application...

The BPM Application wizard opens. Name the application ex: "**Demo**" and accept the default directory for storing application files (C:\JDeveloper\mywork). Select BPM Application in the **Application Template panel**.



Application Name: Demo

Directory: D:\JDeveloper\mywork\Demo

Application Package Prefix:

Application Template:

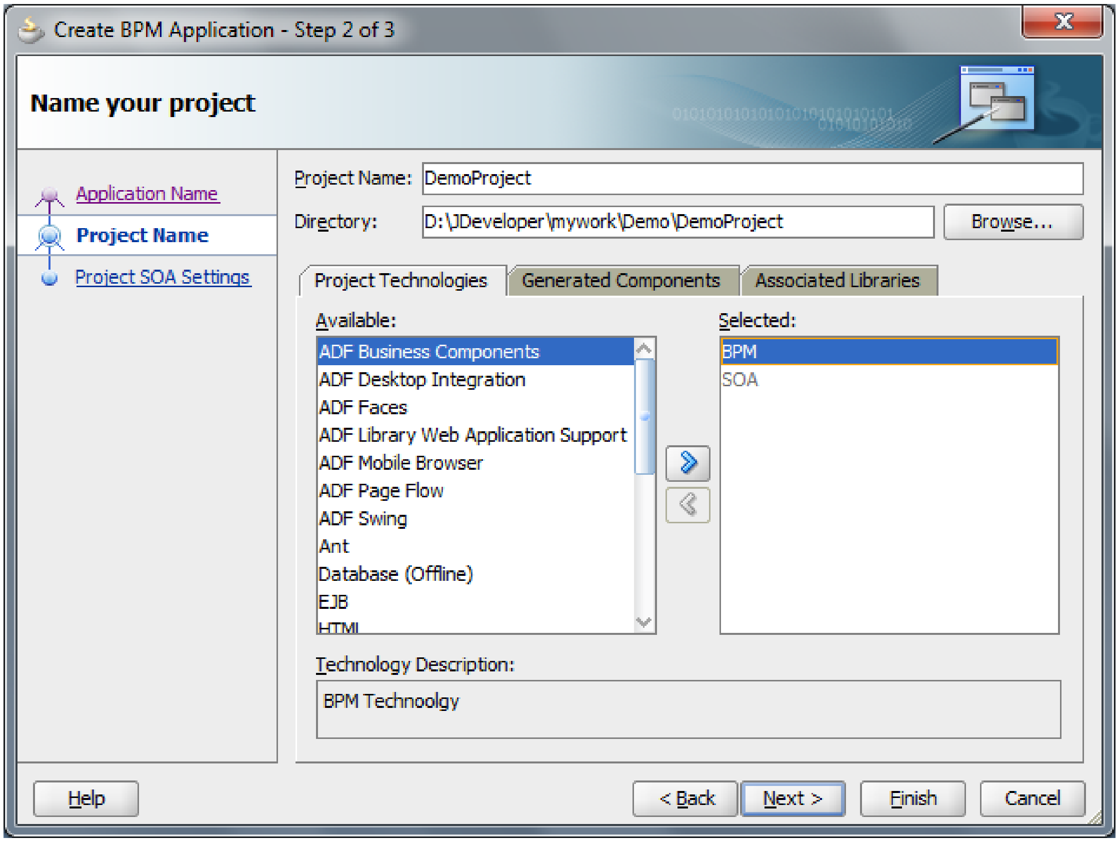
Generic Application
Creates an application which includes a single project. The project is not preconfigured with JDeveloper technologies, but can be customized to include any technologies.

BPM Application
Creates a BPM application. The application consists of one BPM project.This project has also SOA technology

Fusion Web Application (ADF)
Creates a databound ADF web application. The application consists of one project for the view and controller components (ADF Faces and ADF Task Flows), and another project for the data model (ADF Business Components).

Click **Next**.

Step 3: Enter the Project Name ex: “**DemoProject**”. Notice that **BPM** and **SOA** are selected as project technologies by default. Click Finish.



Create BPM Application - Step 2 of 3

Name your project

Project Name: DemoProject

Directory: D:\JDeveloper\mywork\Demo\DemoProject

Available:

- ADF Business Components
- ADF Desktop Integration
- ADF Faces
- ADF Library Web Application Support
- ADF Mobile Browser
- ADF Page Flow
- ADF Swing
- Ant
- Database (Offline)
- EJB
- HTML

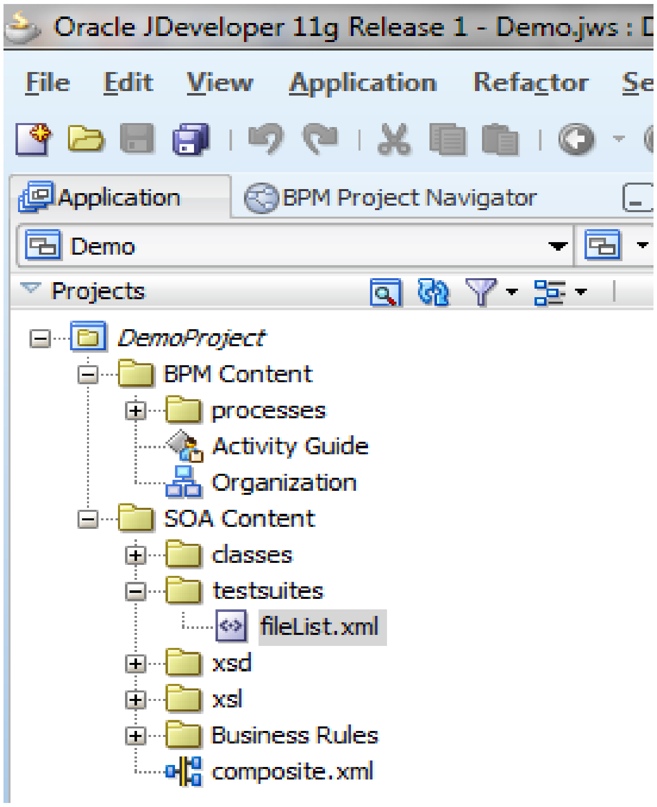
Selected:

- BPM
- SOA

Technology Description: BPM Technoolgy

Click Next.

Step 4: In the upper left corner of the JDeveloper Studio window, you see the **Navigator panel**. This contains two tabs that will be important to you as you perform this tutorial: The **Application Navigator** tab and the **BPM Project Navigator** tab. Currently the Application Navigator tab is selected by default. You can see the **Demo** application appearing in the drop-down list just above the panel and the **DemoProject** appearing as the parent node within the panel. The fact that it appears in italics indicates that there are unsaved changes.

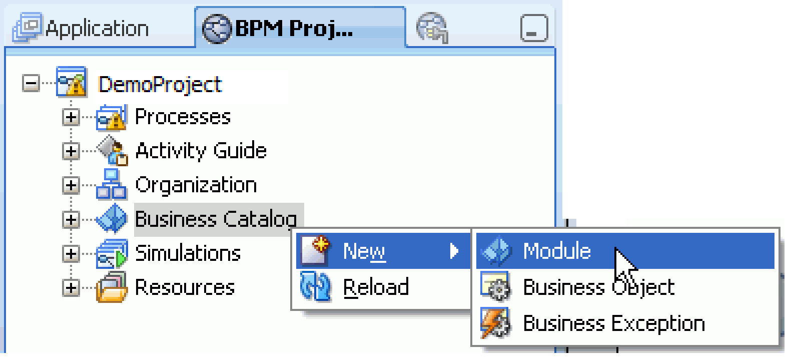


Click the Save All icon on the main toolbar.

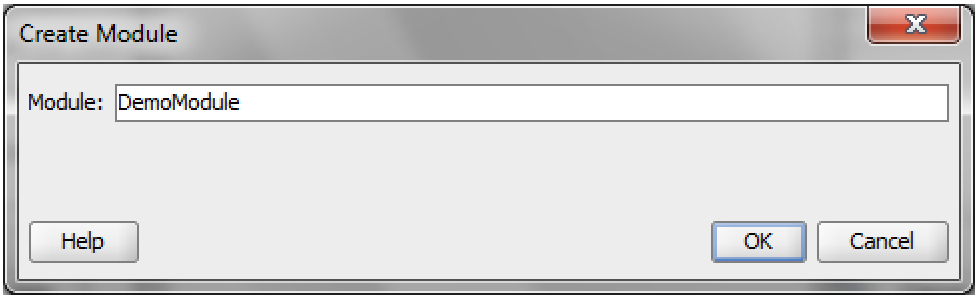
1.5.2 Creating the business object

Step 5: Now you will create a business object capable of storing multiple pieces of data.

Business objects are stored in modules within the Business Catalog. In the BPM Project Navigator, expand the **DemoProject** node. Right click on Business Catalog and select **New** > **Module**.

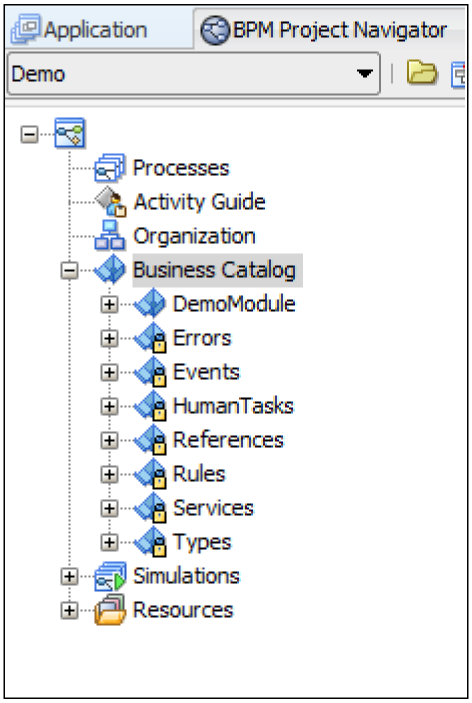


When prompted to name the new module, enter the desired name ex: "**DemoModule**" and click OK.

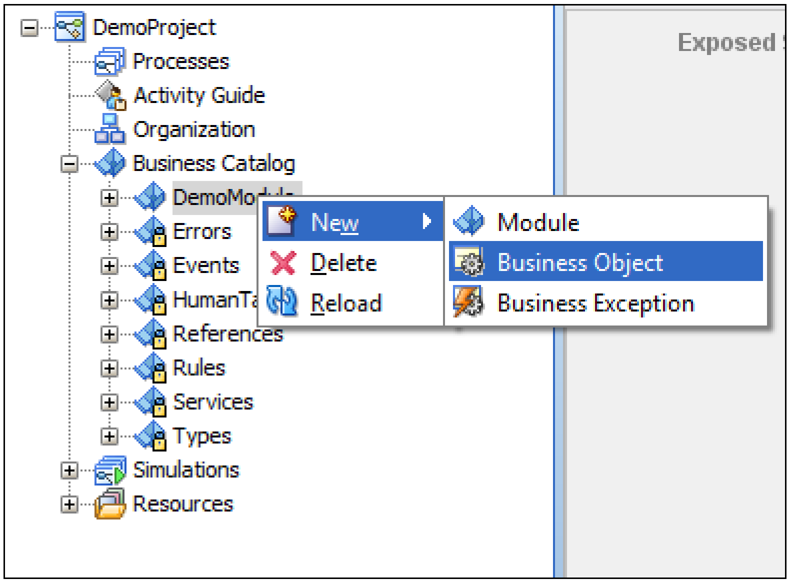


Module: DemoModule

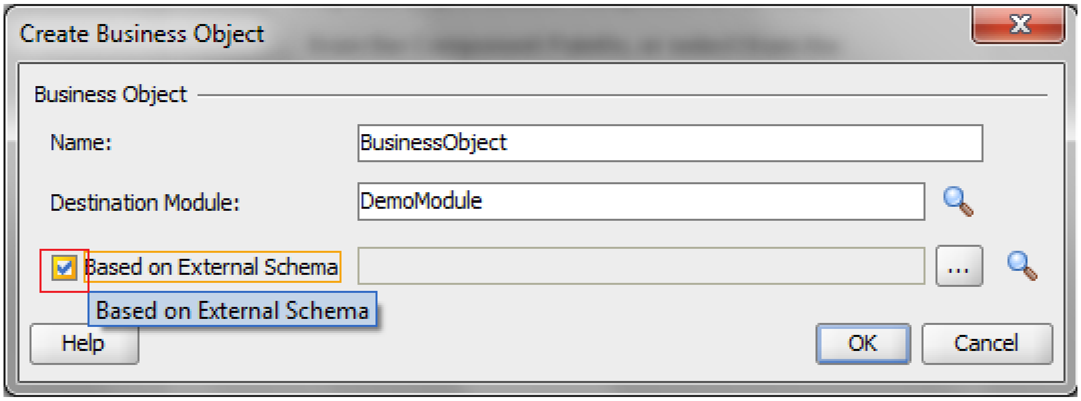
The **DemoModule** module now appears beneath the Business Catalog node.



Right click the **DemoModule** module and select **New** > **Business Object**.



In the Create Business Object window, enter desired Name ex: “**BusinessObject**” and accept **DemoModule** as the Destination Module and check the check box and Click on the Search Icon.

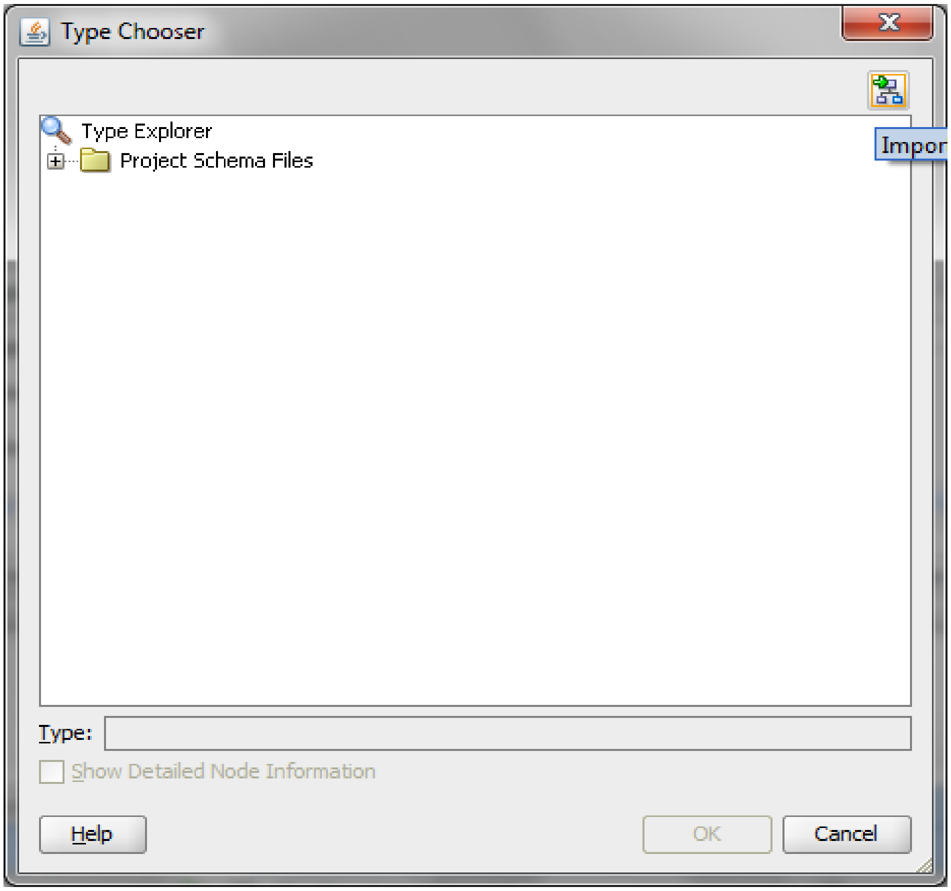


Name: BusinessObject

Destination Module: DemoModule

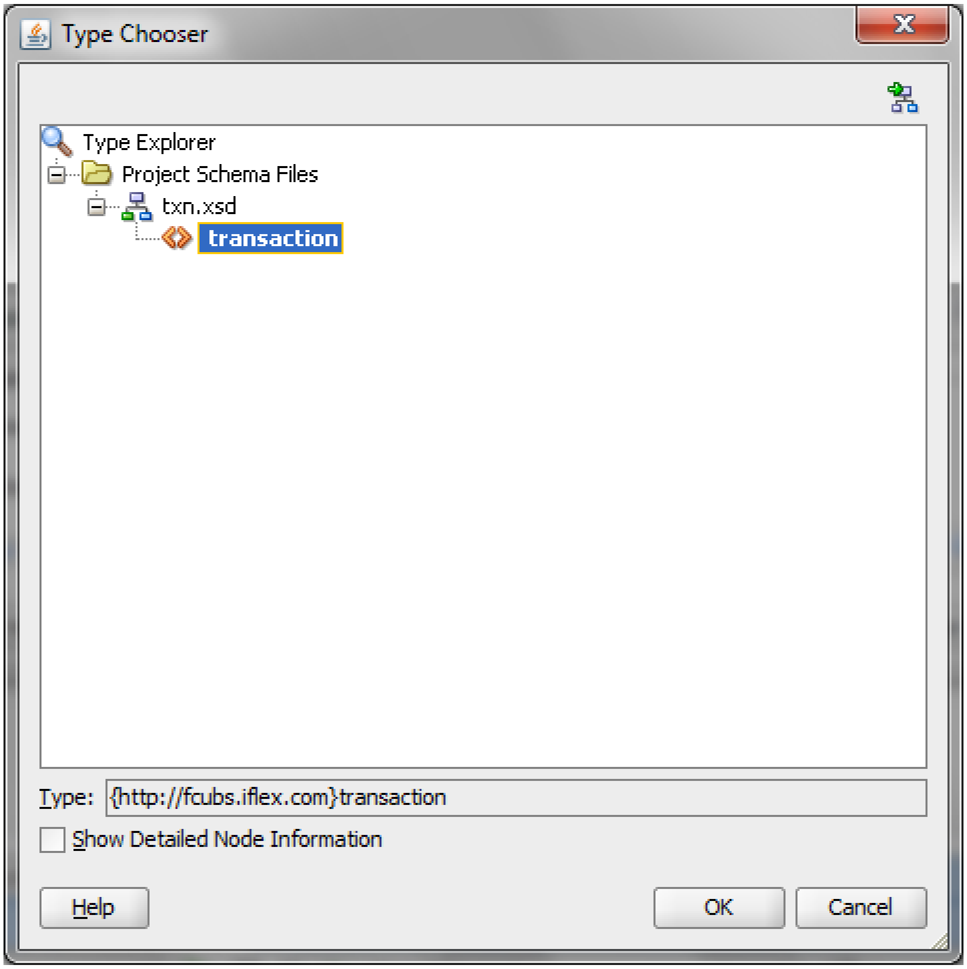
☑ Based on External Schema

On click of Search Icon the following window appears, now click on the Icon at right most Corner to browse the location of the **xsd** that is to be loaded. For example **txn.xsd** in this project is loaded. After loading the **xsd** , it prompts for the copy of the **xsd** to the project . Click Ok. So that the **xsd** is copied to the project.



After loading the xsd, window should appear like the below and select the xsd and Click Ok.

Transaction XSD →



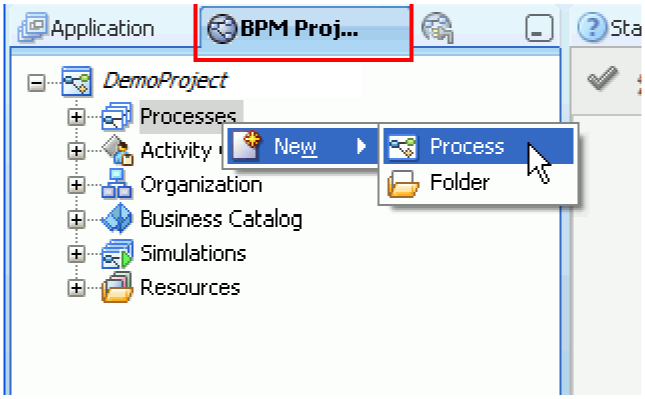
Type: {http://fcubs.iflex.com}transaction

□ Show Detailed Node Information

Click OK again and Save all.

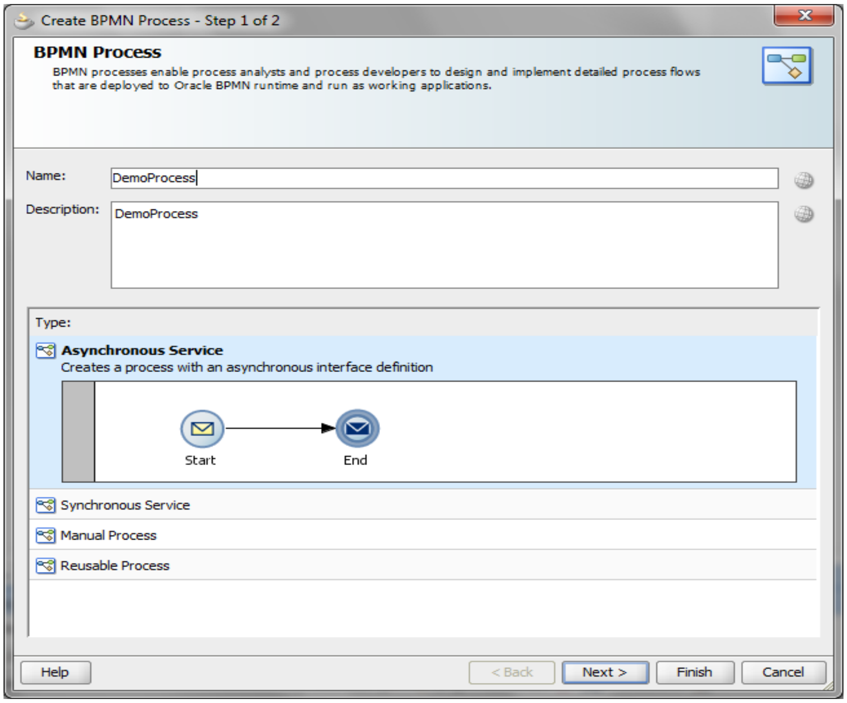
1.5.3 Creating a new process

Step 6: To create a **new** process within this project, first click the **BPM Project Navigator** tab. Then right click on Processes and select **New** > **Process** .



In the BPM Process wizard, select the **Asynchronous Service pattern**.

Click **Next.**


Create BPMN Process - Step 1 of 2

BPMN Process

BPMN processes enable process analysts and process developers to design and implement detailed process flows that are deployed to Oracle BPMN runtime and run as working applications.

Name: DemoProcess

Description: DemoProcess

Type:

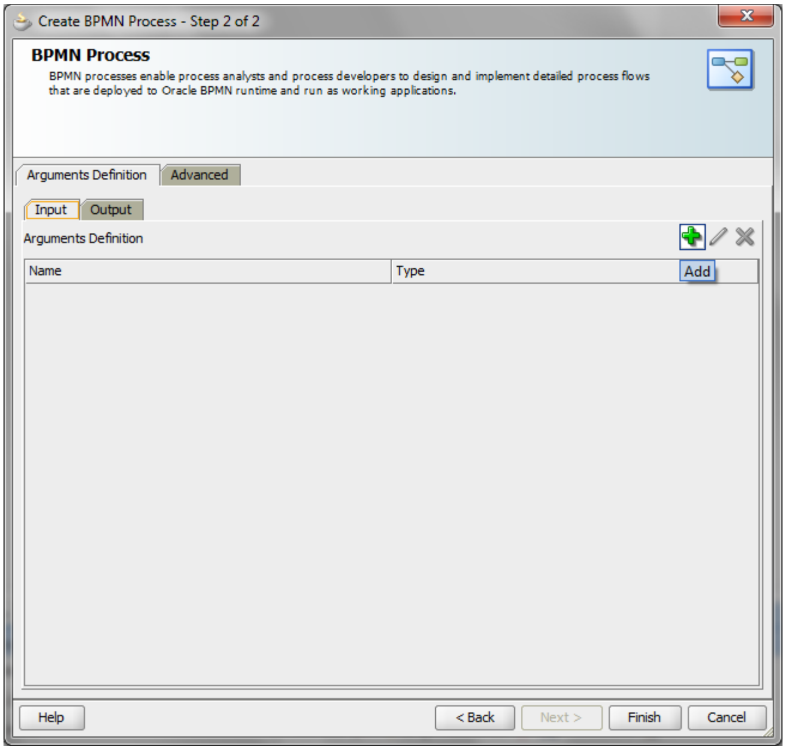
Asynchronous Service
Creates a process with an asynchronous interface definition

Synchronous Service

Manual Process

Reusable Process

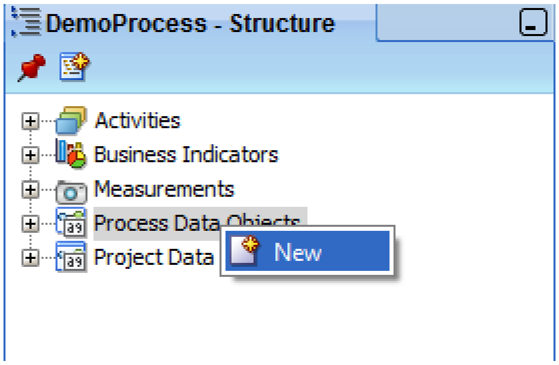
Since we will add the inputs later Click **finish**.


Create BPMN Process - Step 2 of 2
BPMN Process
BPMN processes enable process analysts and process developers to design and implement detailed process flows that are deployed to Oracle BPMN runtime and run as working applications.
Arguments Definition
Advanced
Input
Output
Arguments Definition
Name
Type
Add
Help
< Back
Next >
Finish
Cancel

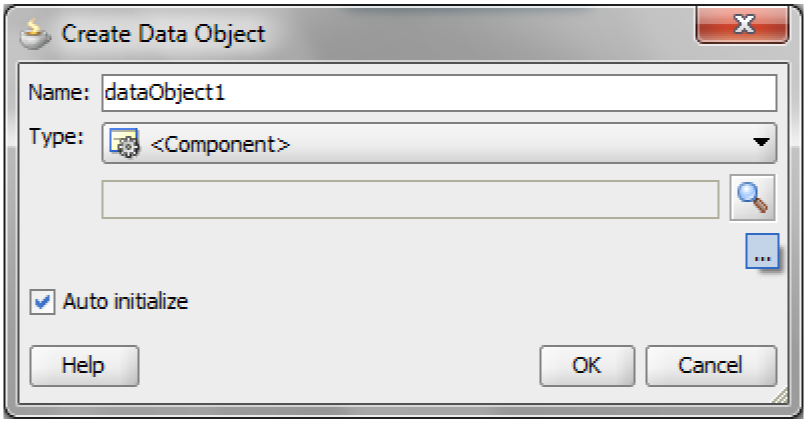
1.5.4 Creating the data objects

Step 7: When a process has been given focus, a detailed outline of its structure appears in the Structure pane in the lower left corner of the JDeveloper window.

Right click on Process **Data Objects** in the Structure pane and select New.



In the Create Data Object popup, enter the Name and click the ellipses button to open another window to search for complex data types.

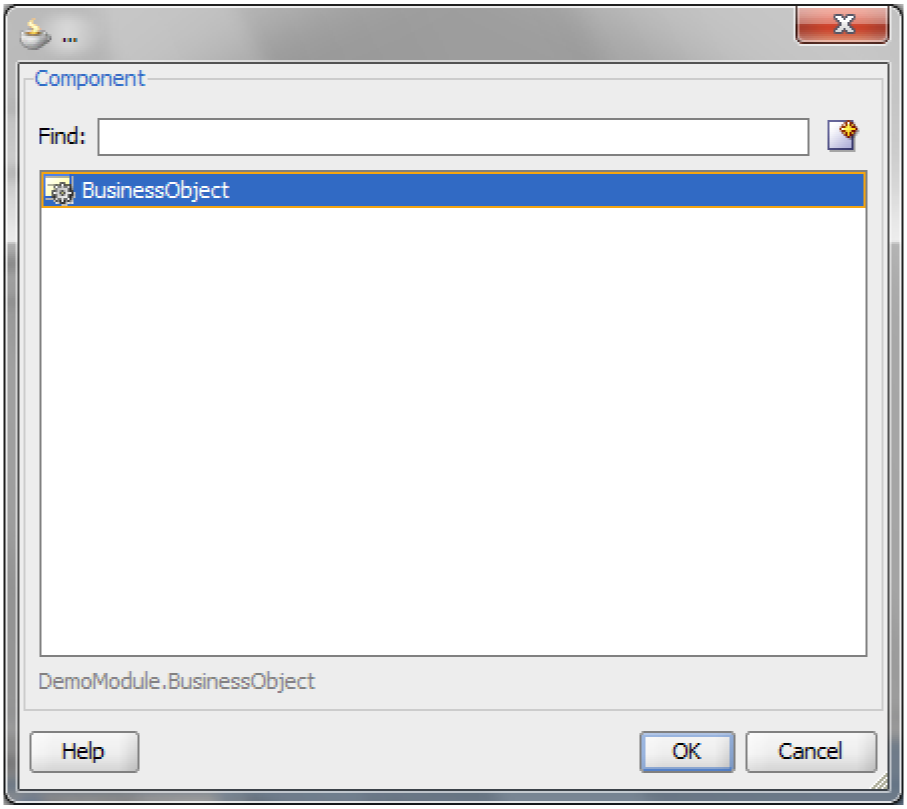


Name: dataObject1

Type: <Component>

☑ Auto initialize

In the Browse Types window, select <**Component**> as the Type and then select **BusinessObject** from the list of components appearing below. Click OK.

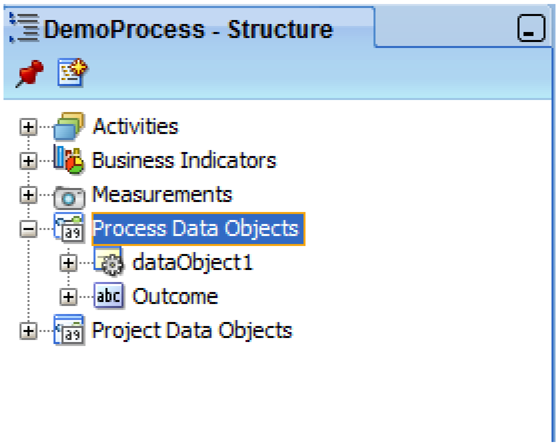


BusinessObject

DemoModule.BusinessObject

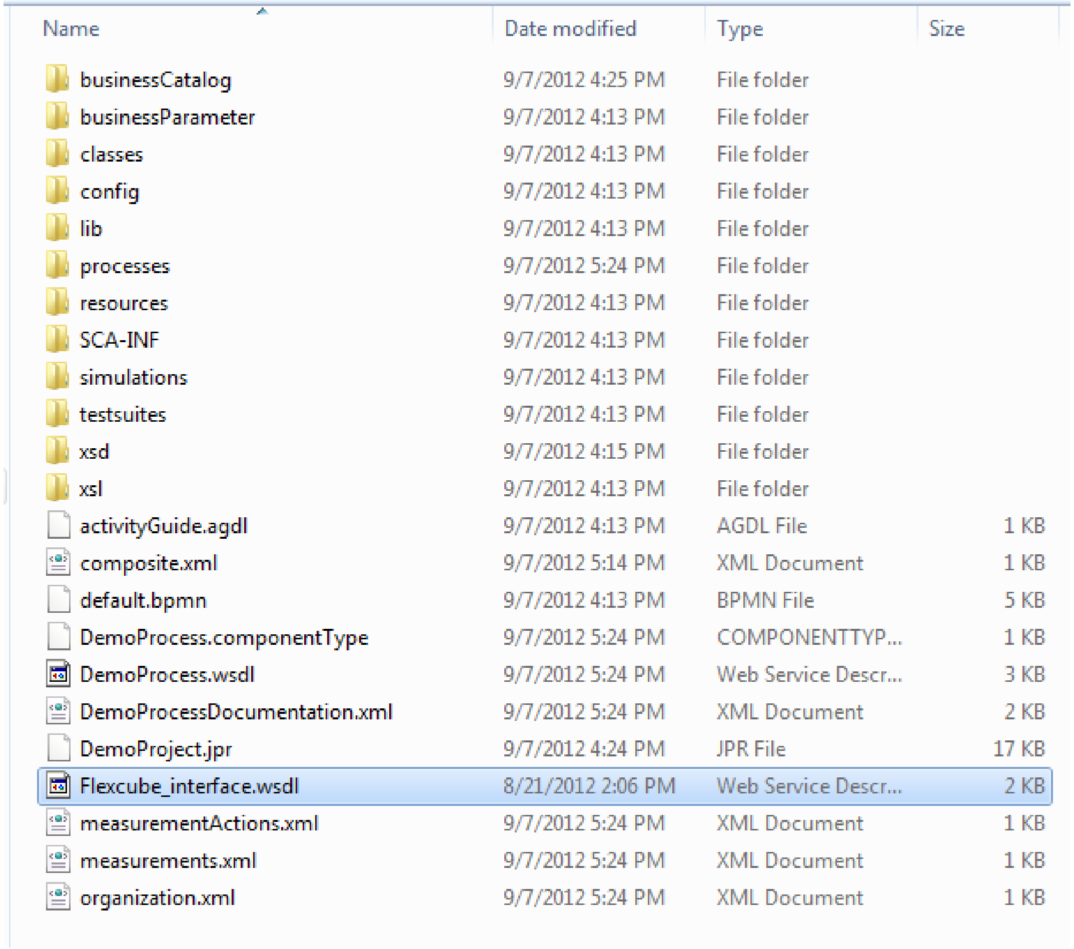
Back in the Create **Data Object** window, click OK again. The data object now appears in the **Structure pane**.

Create another process **data object** of type String to hold the **Outcome**.



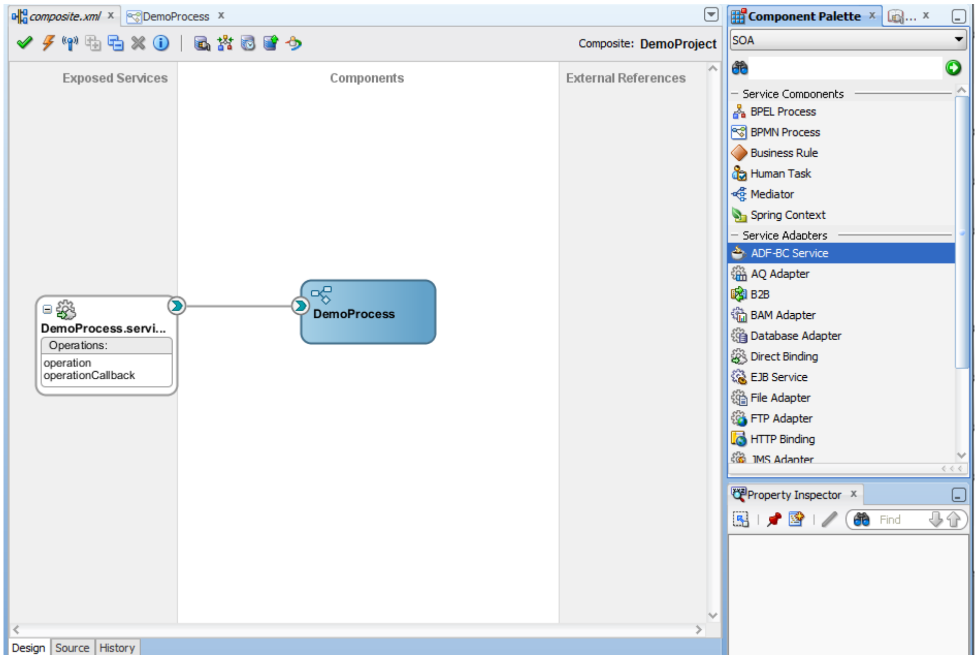
1.5.5 Adding a ADF BC Service Adapter

Step 8: Copy the **Flexcube_interface**.**wsdl** to Project Location

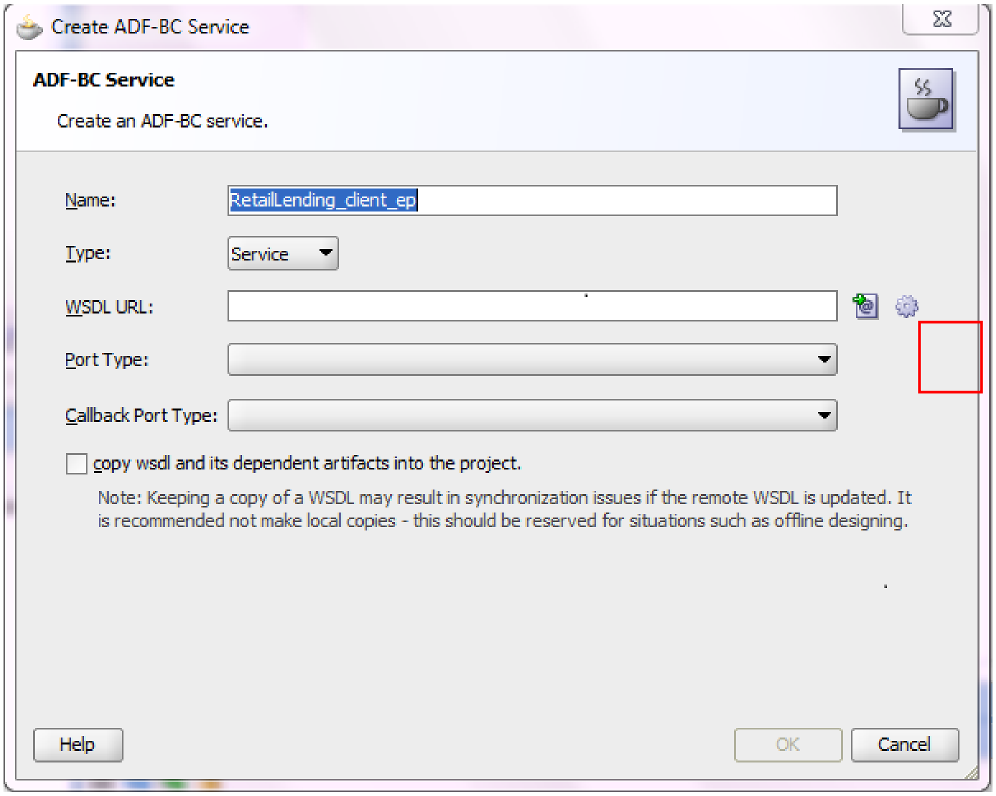


Name	Date modified	Type	Size
businessCatalog	9/7/2012 4:25 PM	File folder	
businessParameter	9/7/2012 4:13 PM	File folder	
classes	9/7/2012 4:13 PM	File folder	
config	9/7/2012 4:13 PM	File folder	
lib	9/7/2012 4:13 PM	File folder	
processes	9/7/2012 5:24 PM	File folder	
resources	9/7/2012 4:13 PM	File folder	
SCA-INF	9/7/2012 4:13 PM	File folder	
simulations	9/7/2012 4:13 PM	File folder	
testsuites	9/7/2012 4:13 PM	File folder	
xsd	9/7/2012 4:15 PM	File folder	
xsl	9/7/2012 4:13 PM	File folder	
activityGuide.agdl	9/7/2012 4:13 PM	AGDL File	1 KB
composite.xml	9/7/2012 5:14 PM	XML Document	1 KB
default.bpmn	9/7/2012 4:13 PM	BPMN File	5 KB
DemoProcess.componentType	9/7/2012 5:24 PM	COMPONENTTYP...	1 KB
DemoProcess.wsdl	9/7/2012 5:24 PM	Web Service Descr...	3 KB
DemoProcessDocumentation.xml	9/7/2012 5:24 PM	XML Document	2 KB
DemoProject.jpr	9/7/2012 4:24 PM	JPR File	17 KB
Flexcube_interface.wsdl	8/21/2012 2:06 PM	Web Service Descr...	2 KB
measurementActions.xml	9/7/2012 5:24 PM	XML Document	1 KB
measurements.xml	9/7/2012 5:24 PM	XML Document	1 KB
organization.xml	9/7/2012 5:24 PM	XML Document	1 KB

Now go to composite.xml drag and drop the ADF-BC Service Adapter from the Component Palette.



Name the Adapter as RetailLending_client_ep.



Create ADF-BC Service

ADF-BC Service

Create an ADF-BC service.

Name: RetailLending_client_ep

Type: Service

WSDL URL:

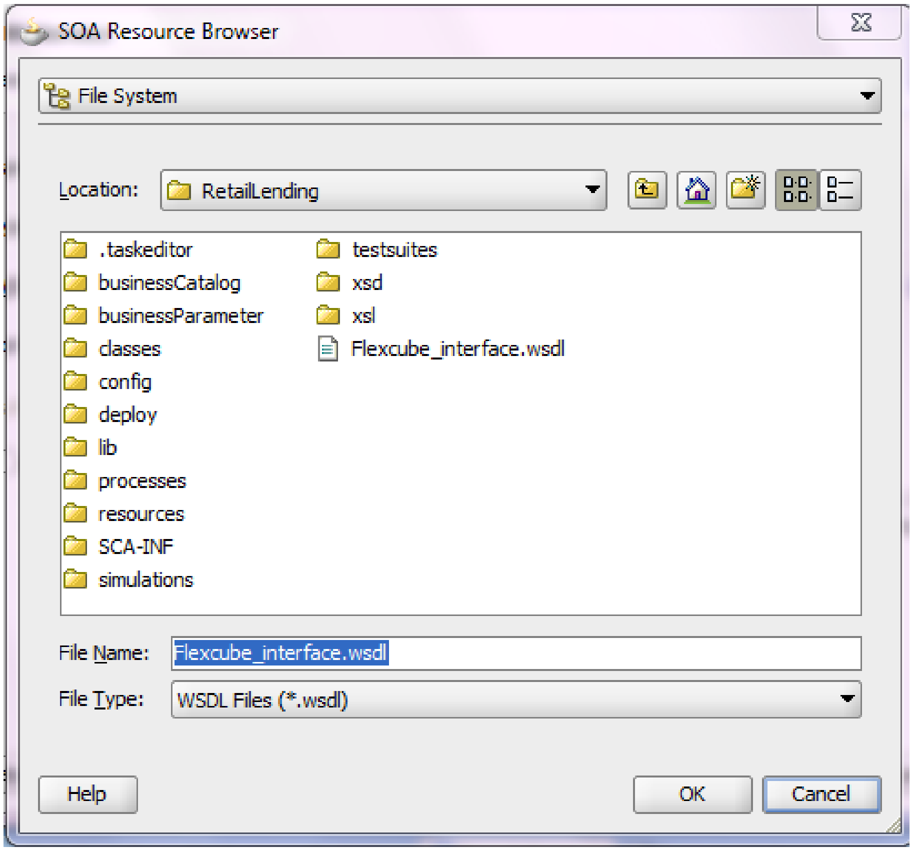
Port Type:

Callback Port Type:

□ copy wsdl and its dependent artifacts into the project.

Note: Keeping a copy of a WSDL may result in synchronization issues if the remote WSDL is updated. It is recommended not make local copies - this should be reserved for situations such as offline designing.

Now Load the **Flexcube_interface.wsdl** from the project location.



SOA Resource Browser

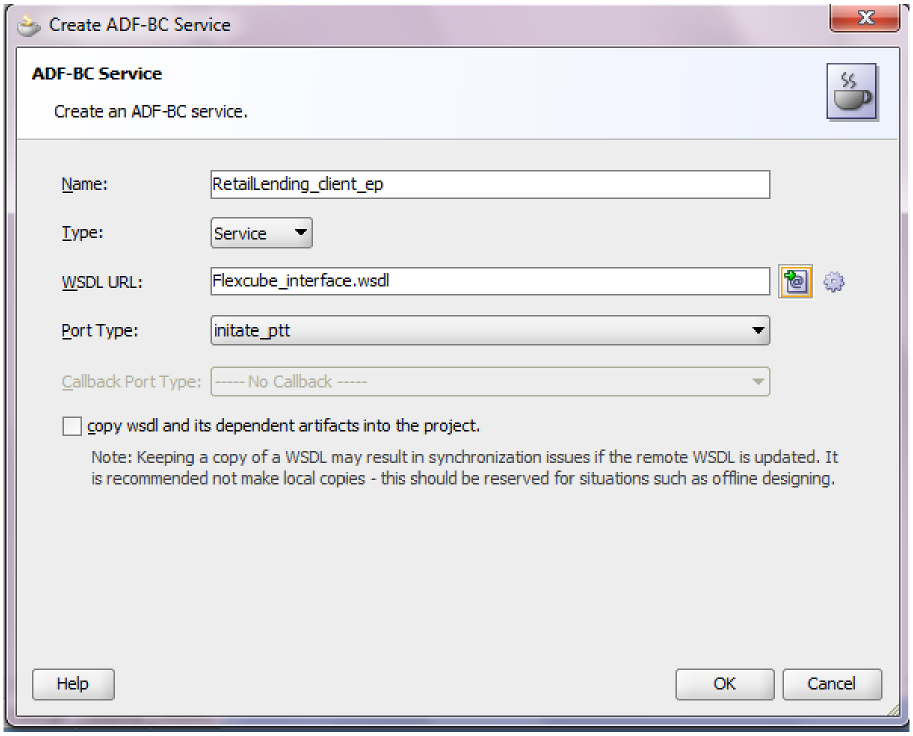
File System

Location: RetailLending

- .taskeditor
- businessCatalog
- businessParameter
- classes
- config
- deploy
- lib
- processes
- resources
- SCA-INF
- simulations
- testsuites
- xsd
- xsl
- Flexcube_interface.wsdl

File Name: Flexcube_interface.wsdl

File Type: WSDL Files (*.wsdl)



Create ADF-BC Service

ADF-BC Service

Create an ADF-BC service.

Name:	RetailLending_client_ep
Type:	Service
WSDL URL:	Flexcube_interface.wsdl
Port Type:	initate_ptt
Callback Port Type:	----- No Callback -----

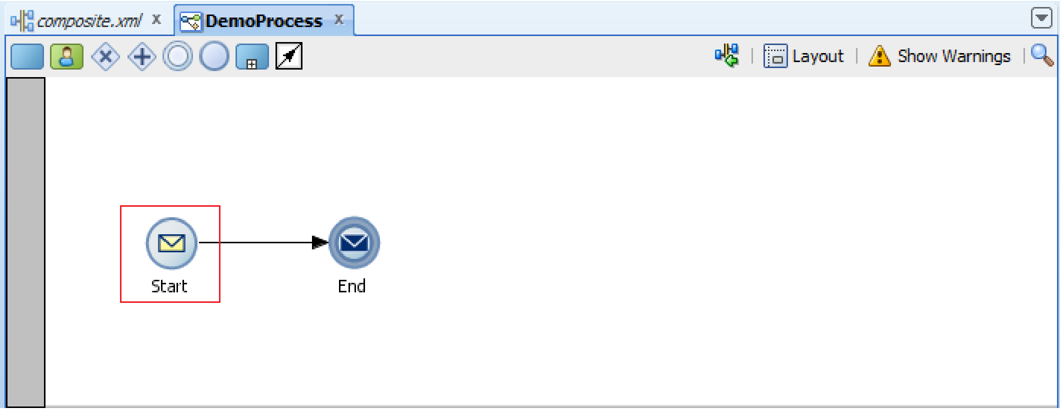
☐ copy wsdl and its dependent artifacts into the project.

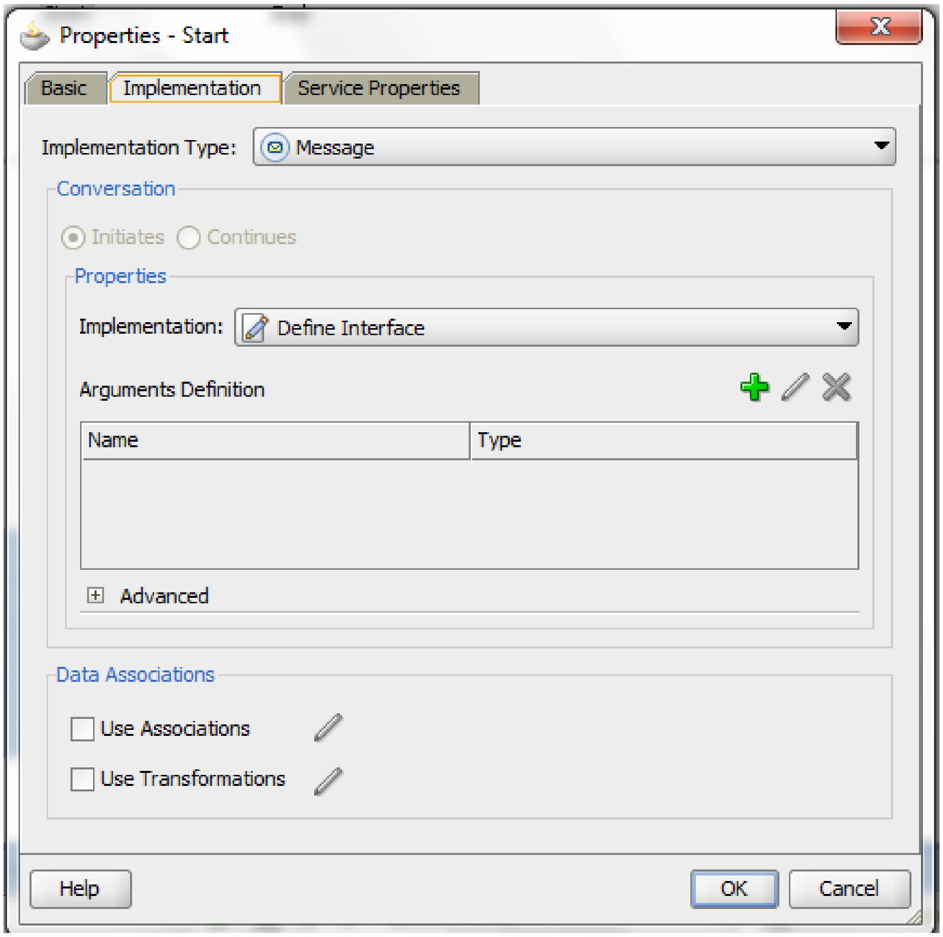
Note: Keeping a copy of a WSDL may result in synchronization issues if the remote WSDL is updated. It is recommended not make local copies - this should be reserved for situations such as offline designing.

Click **OK**

1.5.6 Adding the Created ADF-BC Adapter to the Process

Step 9: Double Click the **Start Event** a Property Window Appears.





Properties - Start

Basic | Implementation | Service Properties

Implementation Type: Message

Conversation

● Initiates ○ Continues

Properties

Implementation: Define Interface

Arguments Definition

Name	Type

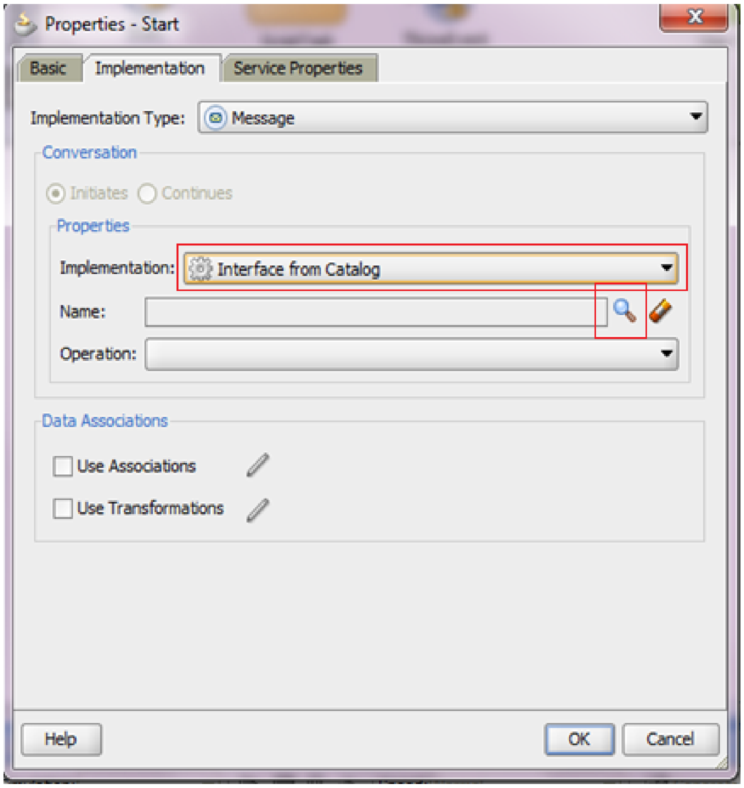
⊞ Advanced

Data Associations

□ Use Associations

□ Use Transformations

Step 10: Now Change the **Implementation** as **Interface from Catalog** and Click the 🔍 icon.



Properties - Start

Implementation Type: Message

Conversation

○ Initiates ○ Continues

Properties

Implementation: Interface from Catalog

Name:

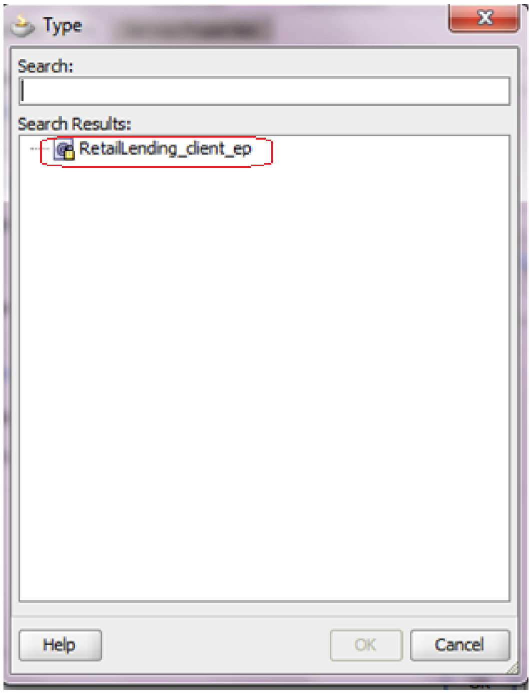
Operation:

Data Associations

□ Use Associations

□ Use Transformations

Select the Adapter from the Window



Type

Search:

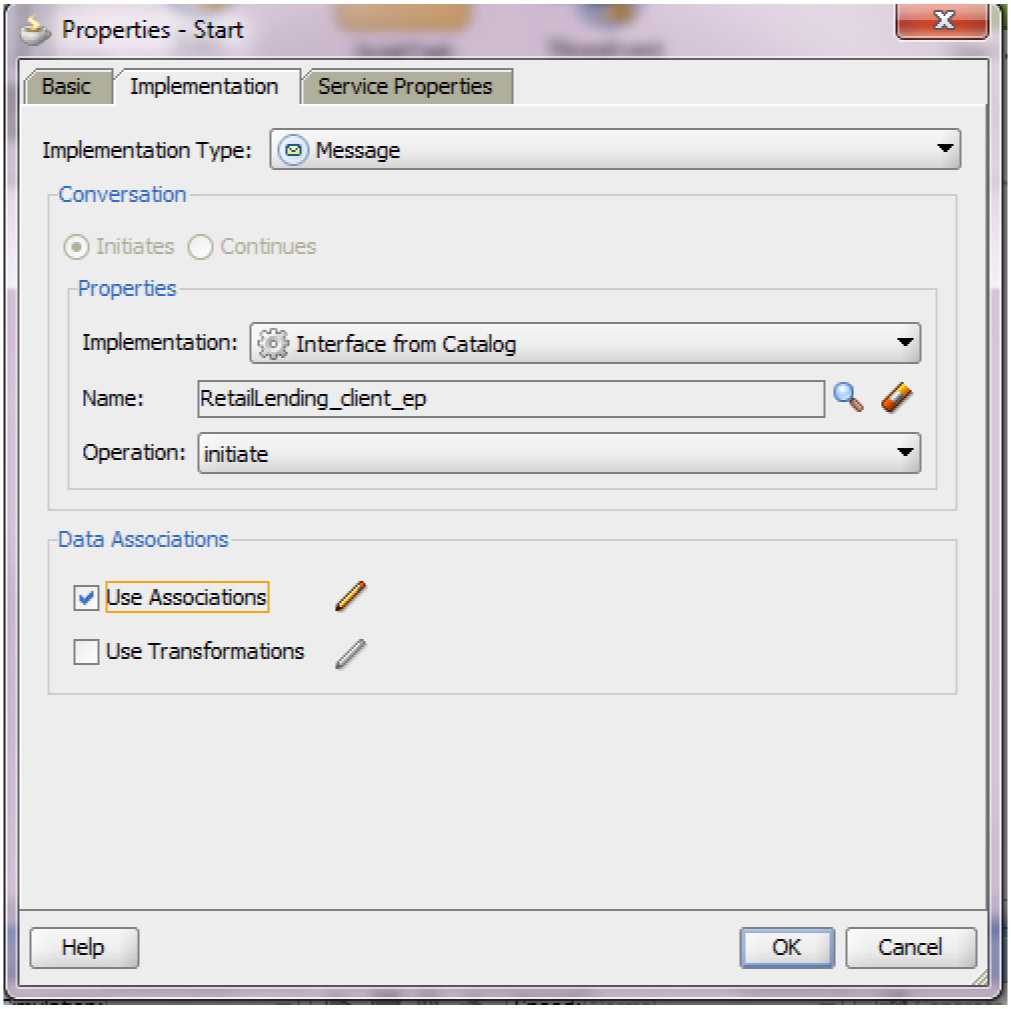
Search Results:

RetailLending_client_ep

Click OK.

1.5.7 Assigning Inputs to the Start Node in the Process

Step 11: Check the **Use Association** Check Box and Click the ✎ icon.



Properties - Start

Implementation Type: Message

Conversation

● Initiates ○ Continues

Properties

Implementation: Interface from Catalog

Name: RetailLending_client_ep

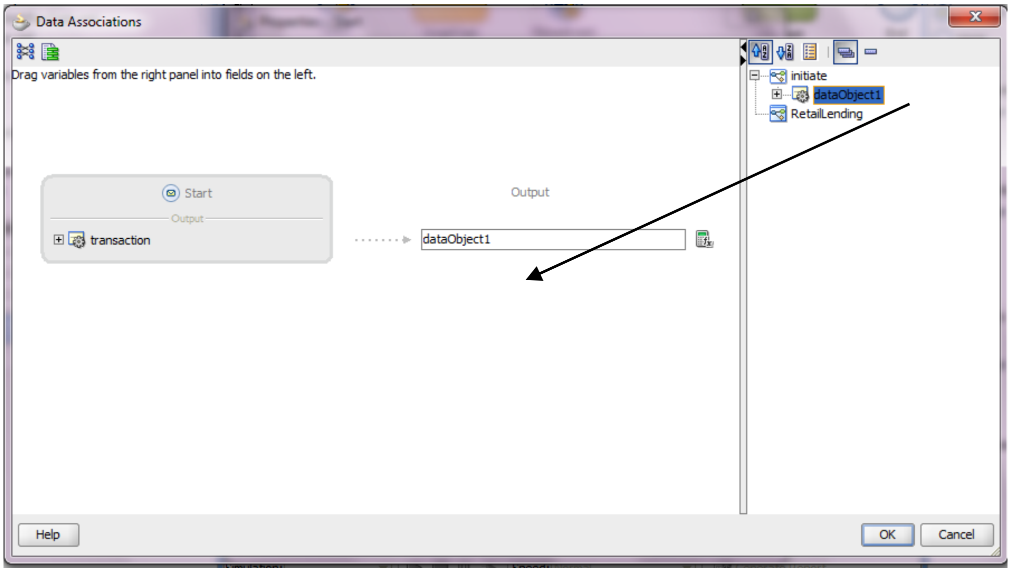
Operation: initiate

Data Associations

☑ Use Associations

□ Use Transformations

After Clicking the icon a **Data Association** Window will appears .Drag the **business object** from the left corner to the text box mapping to the **Output**.

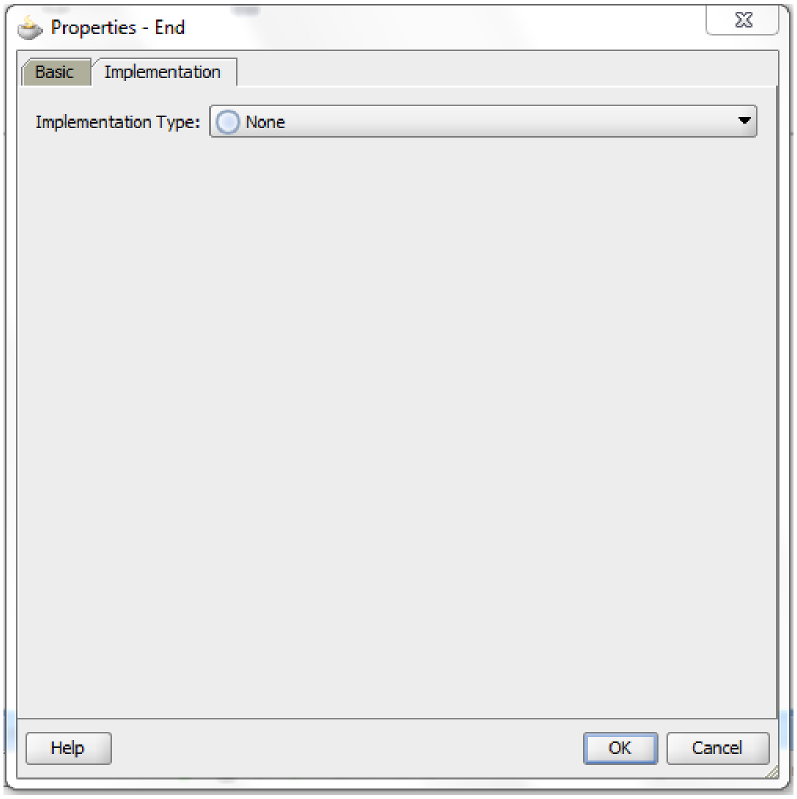


Data Associations

Drag variables from the right panel into fields on the left.

dataObject1

Step 12: Set the End Node as None by selecting **implementation type** as **none**.

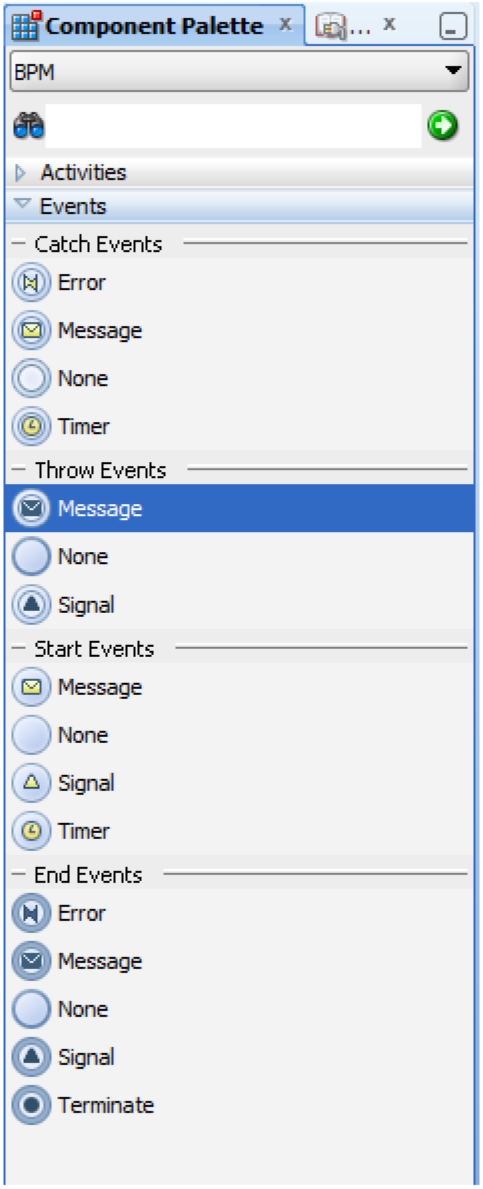


Properties - End

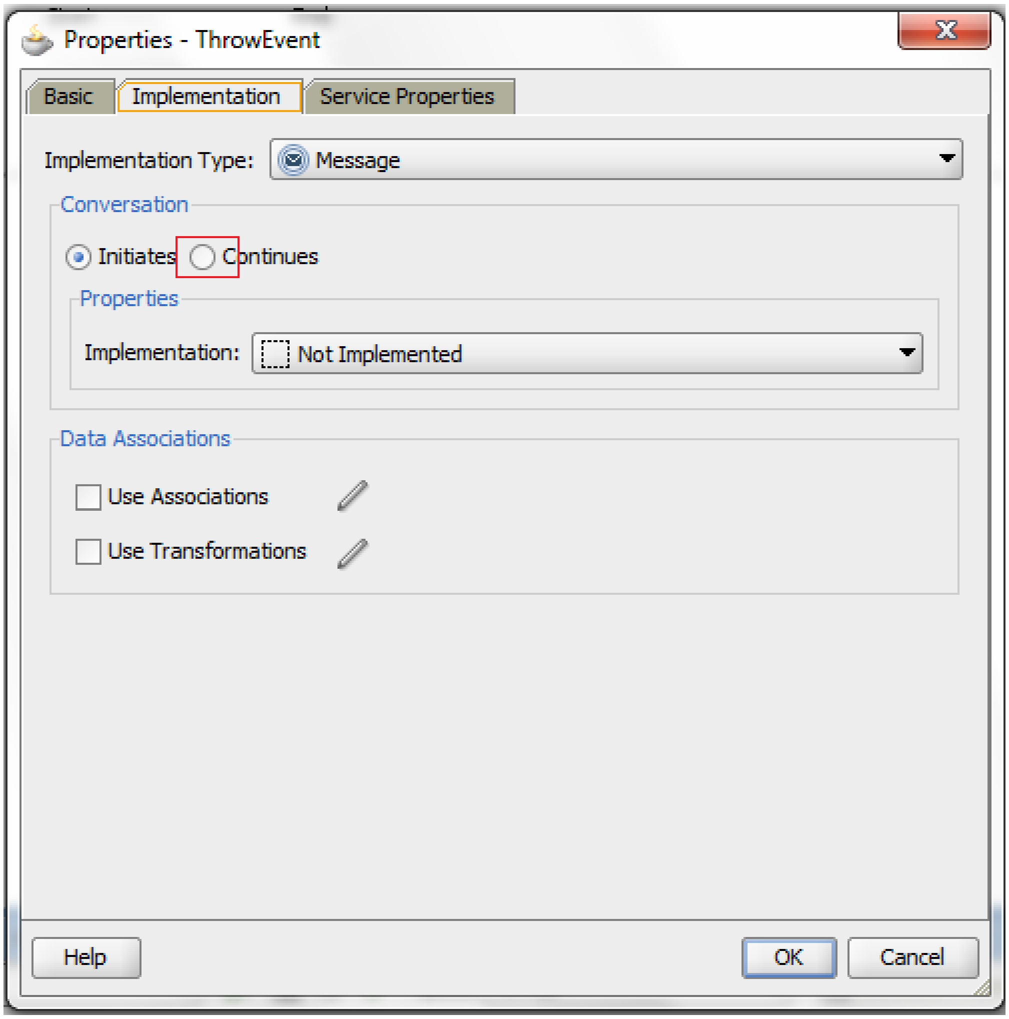
Implementation Type: None

1.5.8 Adding a Throw Event to the Process

Step 13: Drag and Drop the Message Throw Event from Events Pane in Component Palette to the Process.



A Properties window appears. Now select the Continues Radio Button the Throw Event will be automatically Implemented.



Implementation Type: Message

Conversation

● Initiates ○ Continues

Properties

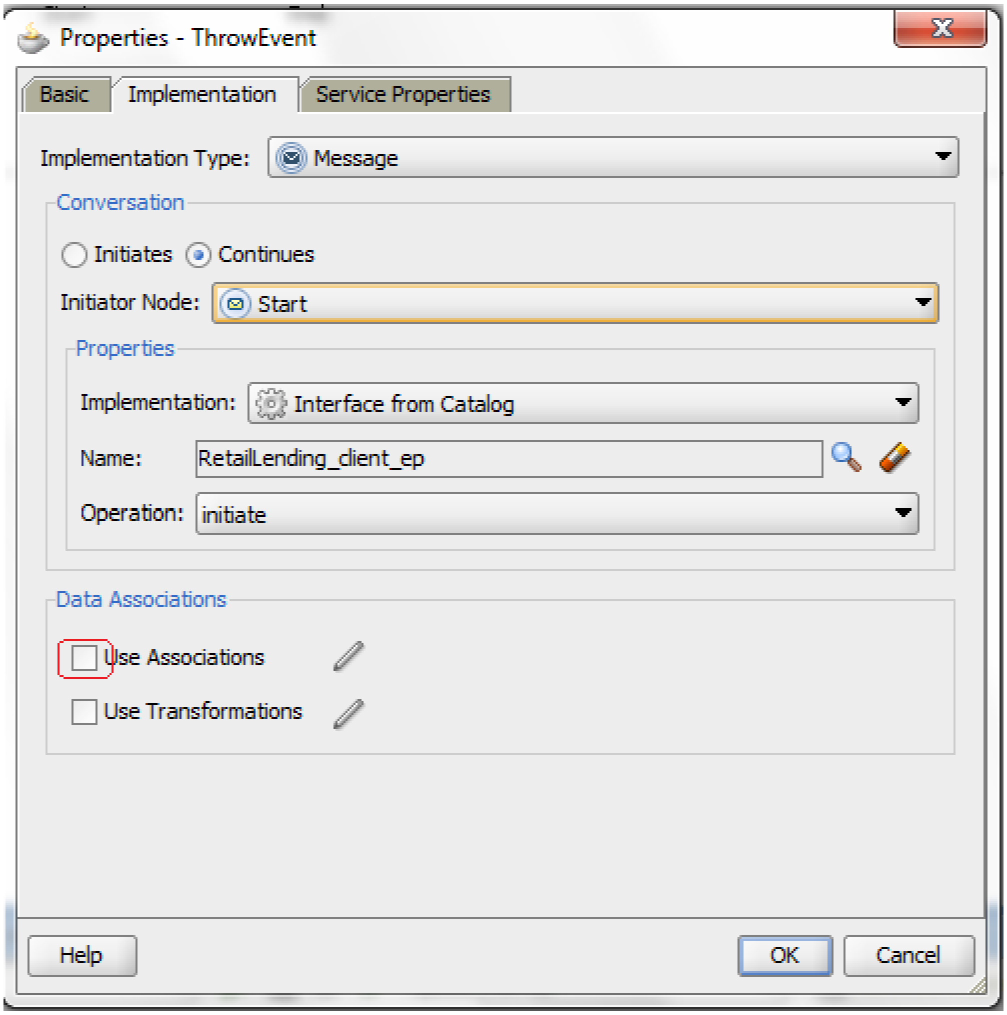
Implementation: Not Implemented

Data Associations

□ Use Associations

□ Use Transformations

Now click the **association** check box.



Properties - ThrowEvent

Implementation Type: Message

Conversation

○ Initiates ● Continues

Initiator Node: Start

Properties

Implementation: Interface from Catalog

Name: RetailLending_client_ep

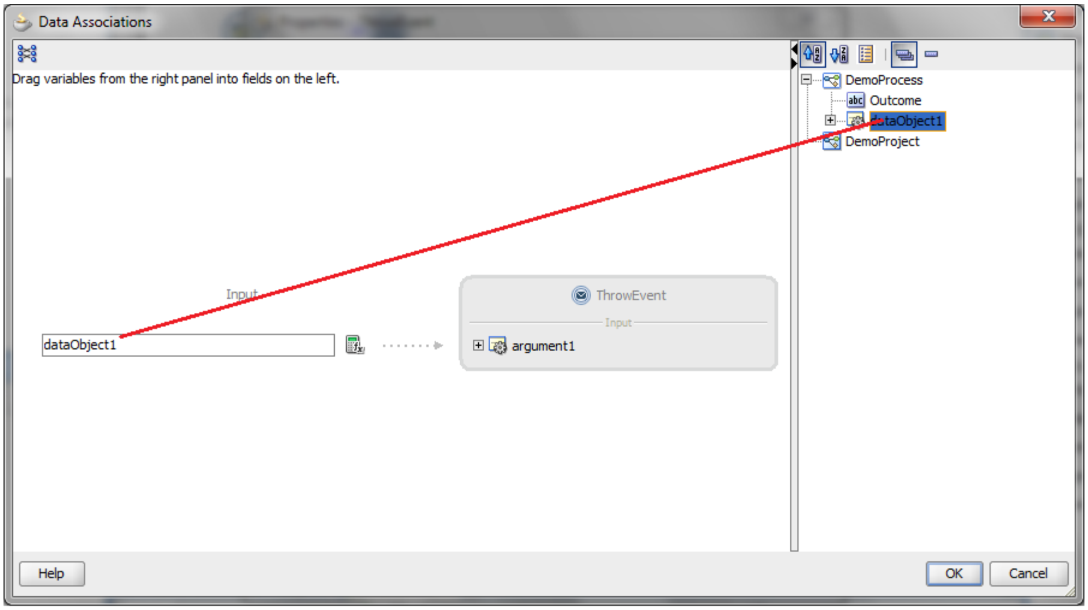
Operation: initiate

Data Associations

□ Use Associations

□ Use Transformations

After Clicking the ✎ icon a **Data Association** Window will appears .Drag the **business object** from the left corner to the text box mapping to the **Input**.

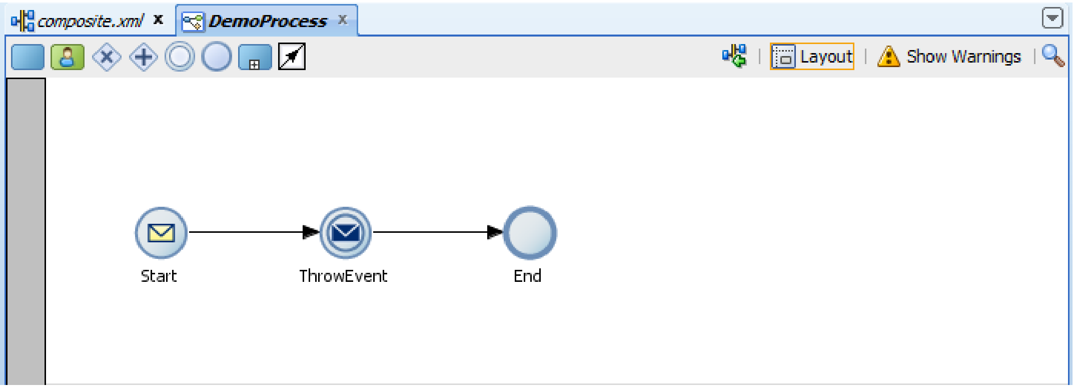


Drag variables from the right panel into fields on the left.

Input: dataObject1

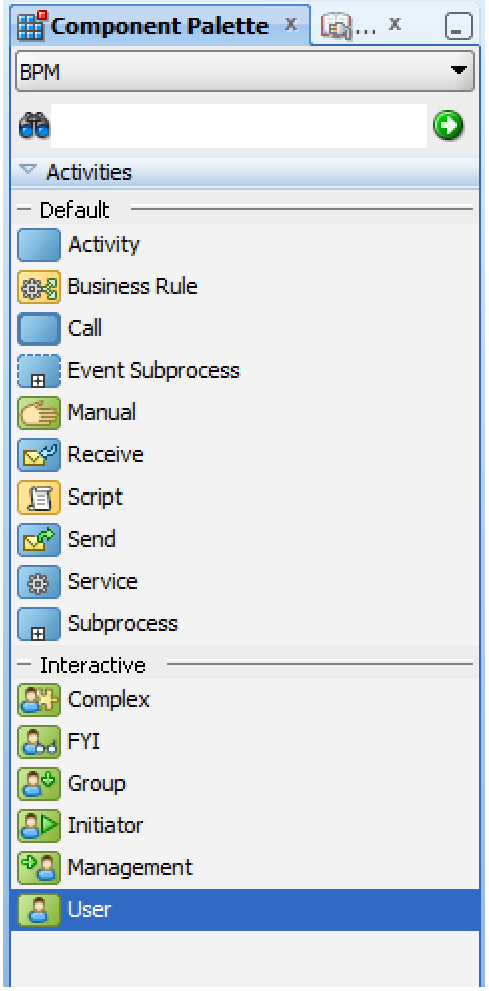
Click **OK**.

Now our Process Look Likes this

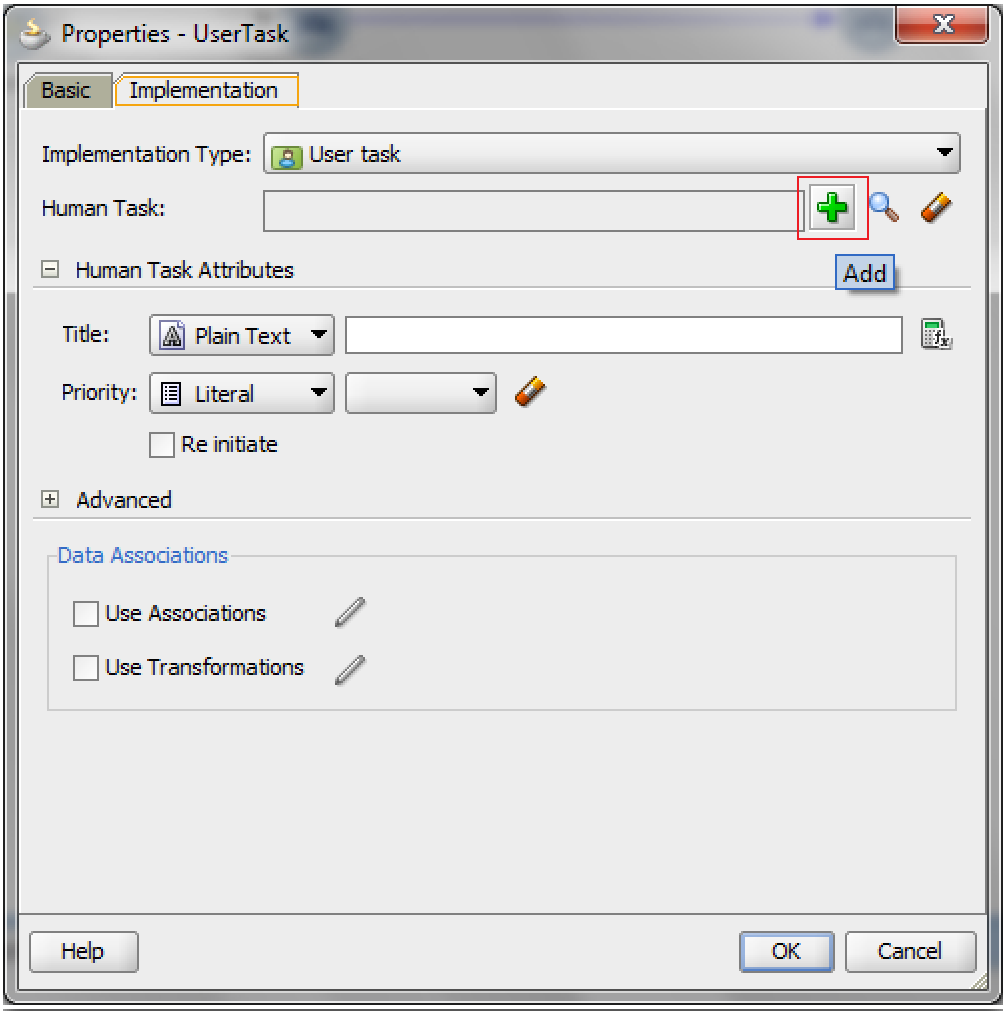


1.5.9 Creating and Implementing Human Tasks

Step 14: Expand the Activities pane in the Component Palette and from the Interactive section, click and drag a User activity, dropping it onto the sequence flow between Throw and End events.



The **user task** properties window will be opened. In the basic tab enter an appropriate name and then go to **implementation tab** and click add Icon of the **Human Task** column.



Properties - UserTask

Implementation Type: User task

Human Task:

Title: Plain Text

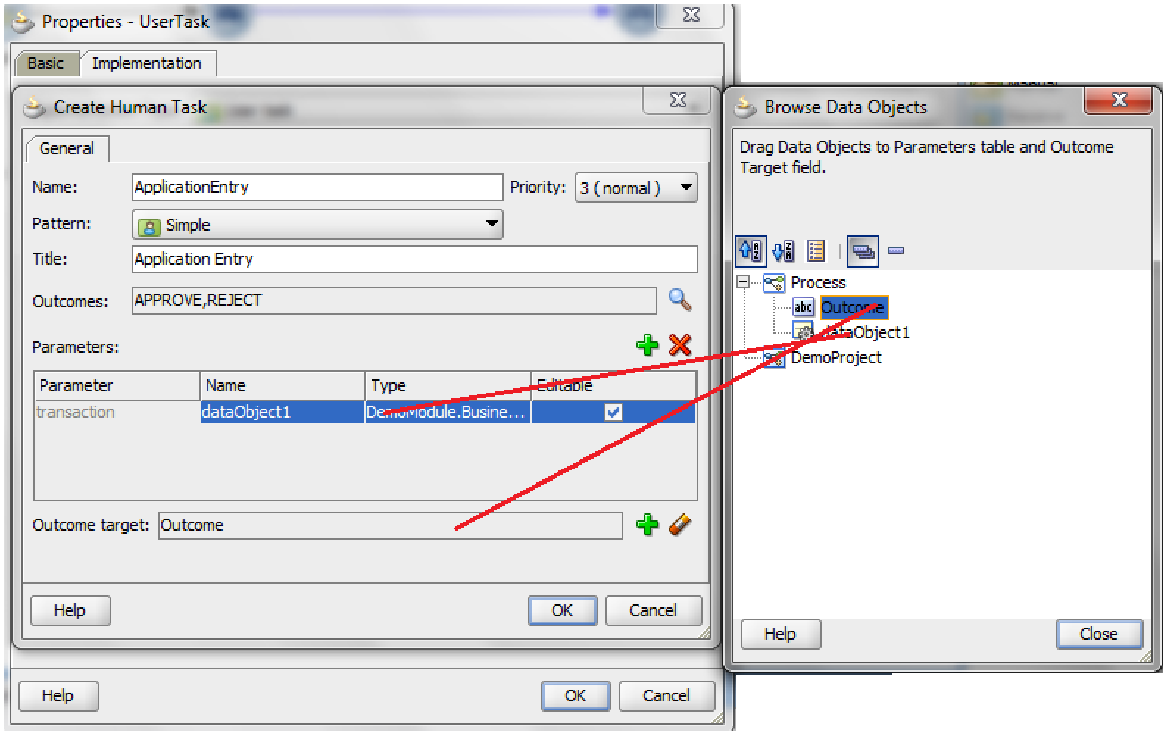
Priority: Literal

□ Re initiate

□ Use Associations

□ Use Transformations

Create **human task** window is opened, Change the Name and title accordingly, Click on the add Icon and map **parameter** and **outcome** target respectively and Check the editable field in the parameter slab.



Properties - UserTask

Create Human Task

Name: ApplicationEntry

Priority: 3 (normal)

Pattern: Simple

Title: Application Entry

Outcomes: APPROVE,REJECT

Parameters:

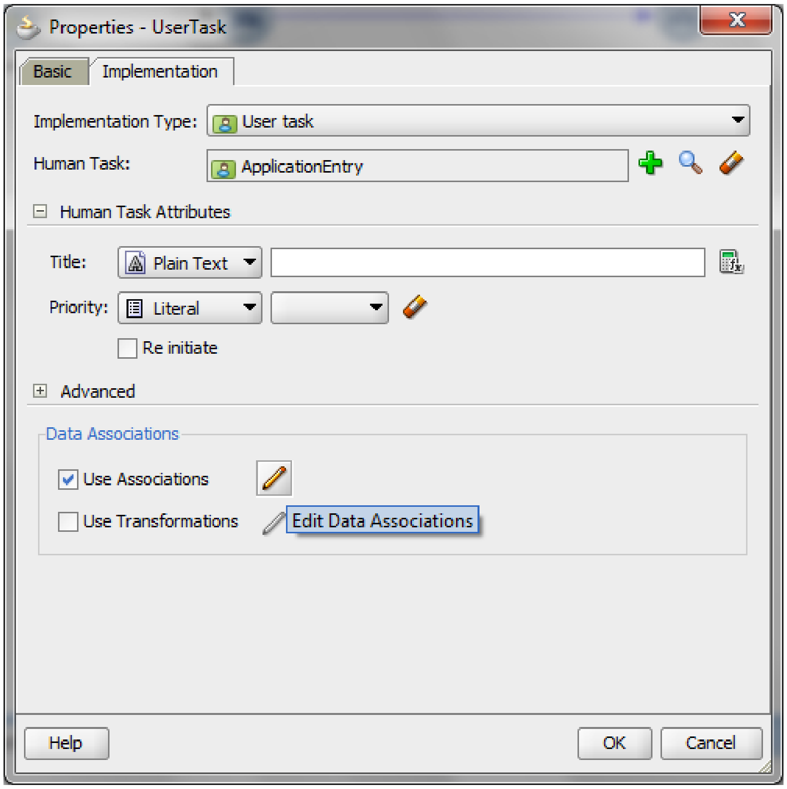
Parameter	Name	Type	Editable
transaction	dataObject1	DemoModule.Busine…	☑

Outcome target: Outcome

Browse Data Objects

Drag Data Objects to Parameters table and Outcome Target field.

Click OK and check the **Use Associations** in the **User Task** properties window.



Properties - UserTask

Implementation Type: User task

Human Task: ApplicationEntry

⊟ Human Task Attributes

Title: Plain Text

Priority: Literal

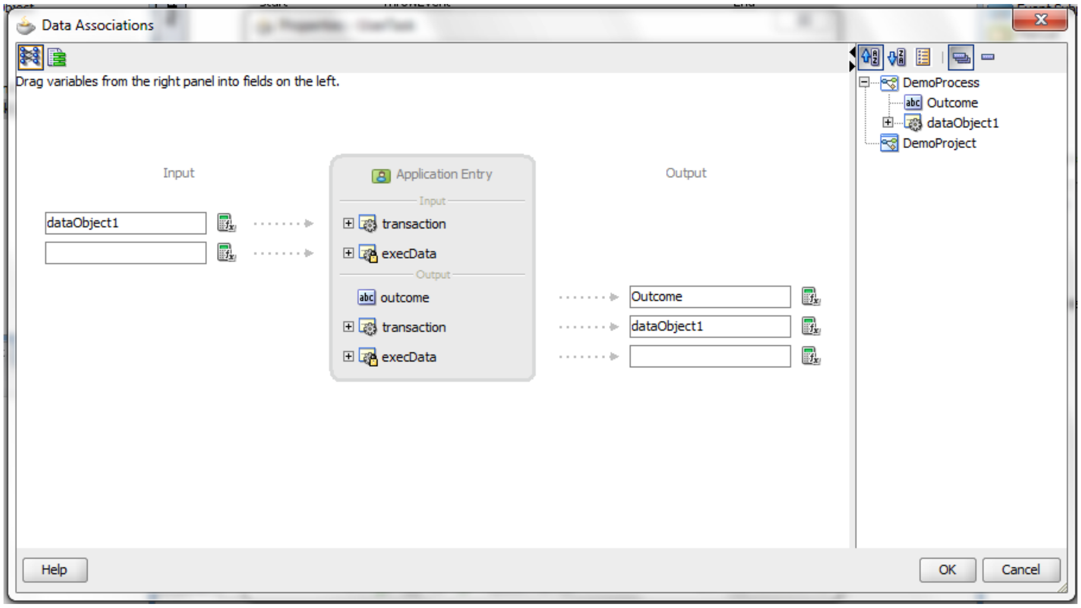
☐ Re initiate

⊞ Advanced

Data Associations

☑ Use Associations

☐ Use Transformations



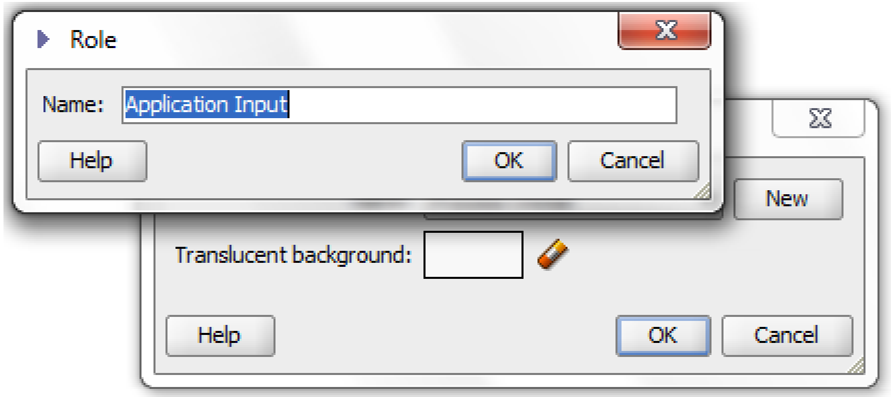
Data Associations

Drag variables from the right panel into fields on the left.

Input: dataObject1 → Application Entry Input: transaction, execData

Application Entry Output: outcome → Outcome; transaction → dataObject1; execData →

Click Ok and a **role properties** window appears.

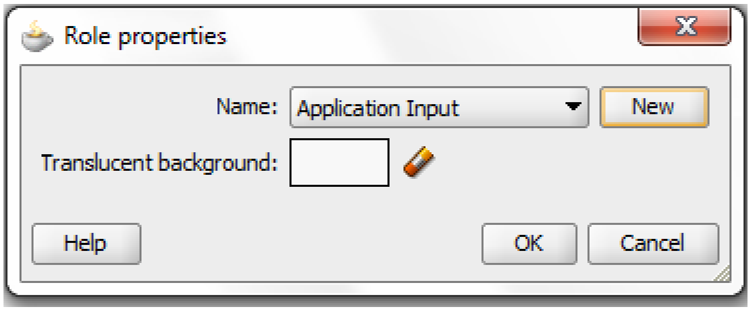


Role

Name: Application Input

Translucent background:

Add a Name to the **Role** Click Ok and Save all.

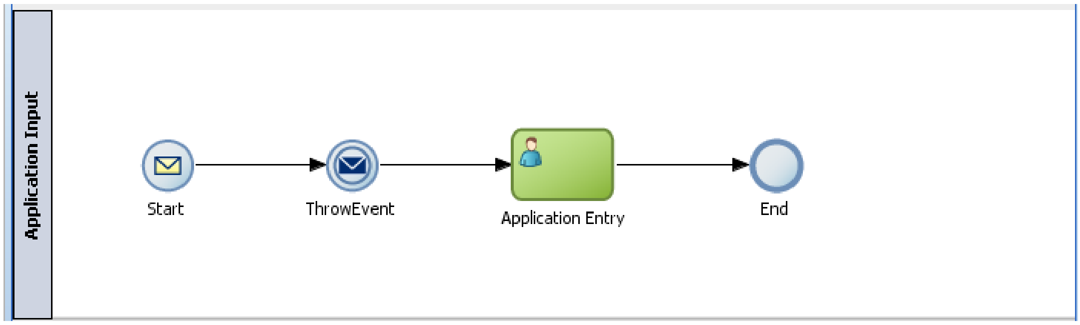


Role properties

Name: Application Input

Translucent background:

Now our Process Look Likes this:



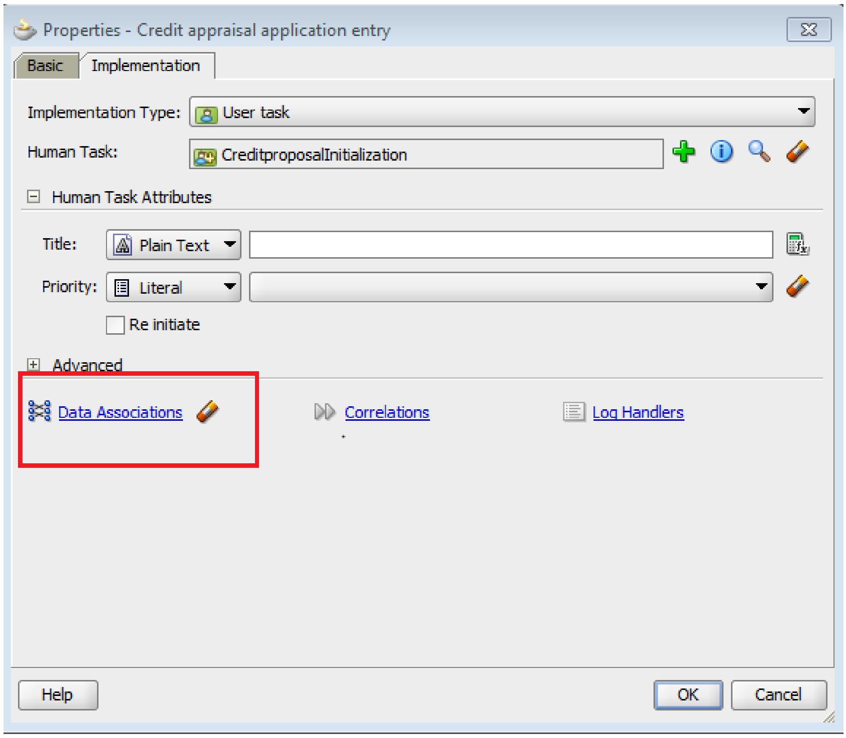
1.5.10 How to get the Conversation Id

To get the infra generated conversation id in the process, the below steps needs to be performed in the **first human task** of the process.

This conversation id is considered as **Application Number in each process**. Conversation id will be **unique** for each task.

Step 1: On double click on the first human task, you will get the below property window.

In property window, go to the implementation tab and click on the highlighted data association.


Properties - Credit appraisal application entry

Implementation Type: User task

Human Task: CreditproposalInitialization

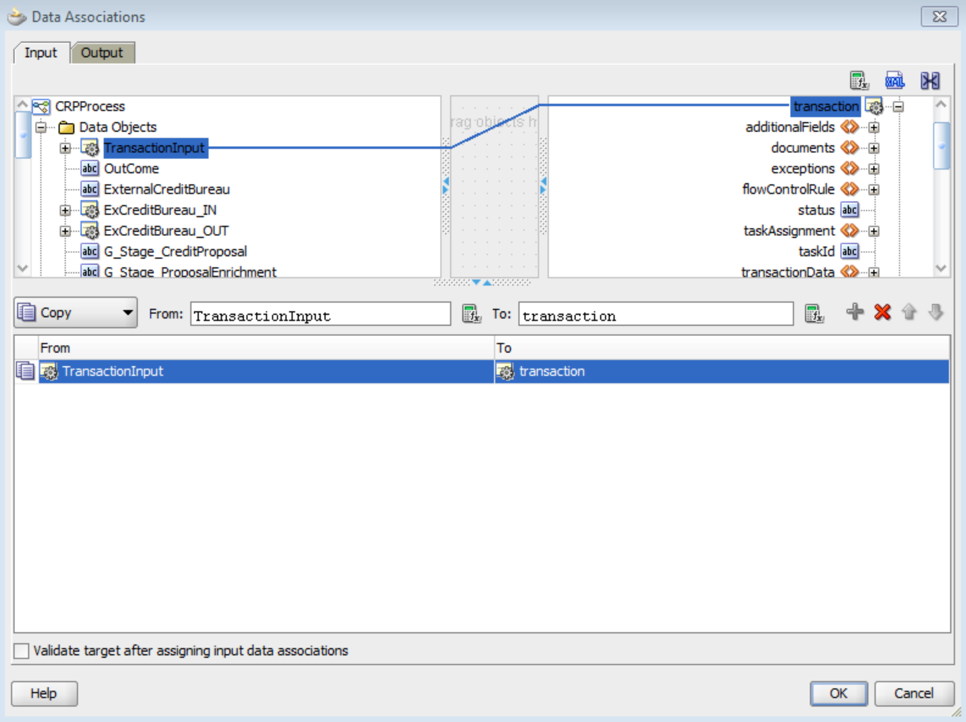
Human Task Attributes

Title: Plain Text

Priority: Literal

□ Re initiate

On clicking the data association, the below input/output window will be opened.

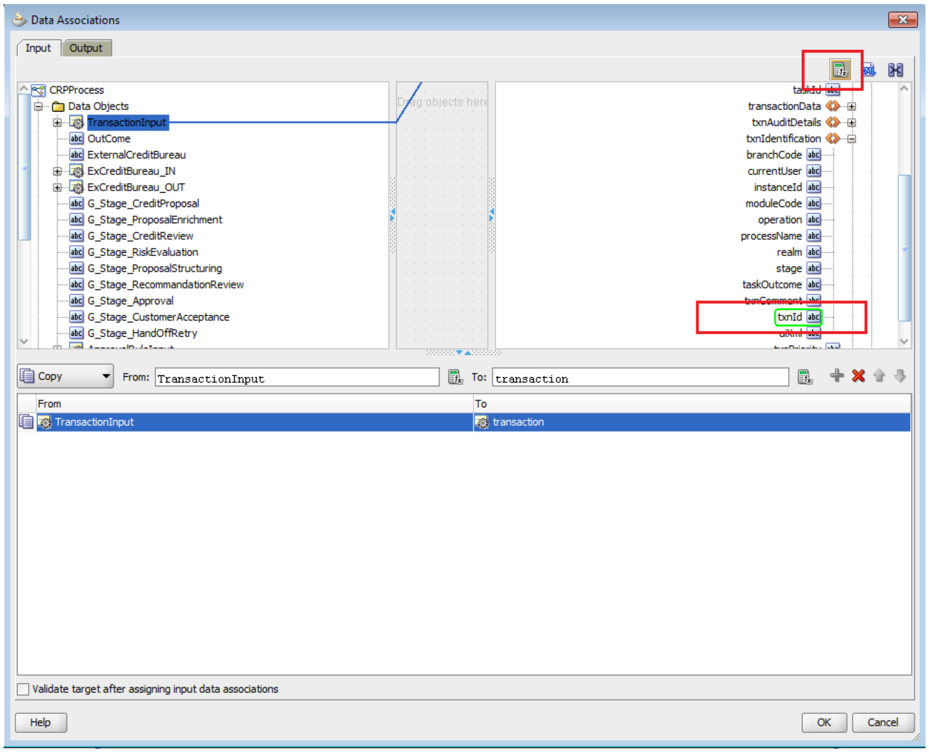


From: TransactionInput To: transaction

From	To
TransactionInput	transaction

□ Validate target after assigning input data associations

Step 2: Drag the highlighted expression icon to the target (**txnId**) then Expression builder window will get opened.

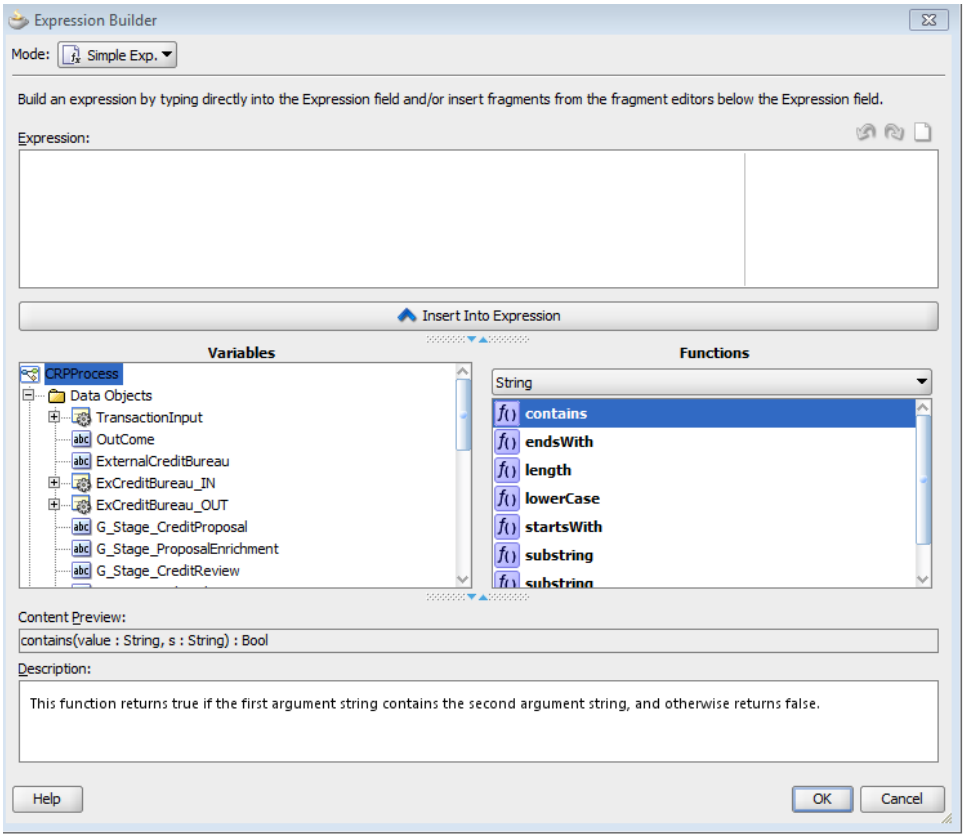


From: TransactionInput To: transaction

From	To
TransactionInput	transaction

□ Validate target after assigning input data associations

Expression builder window



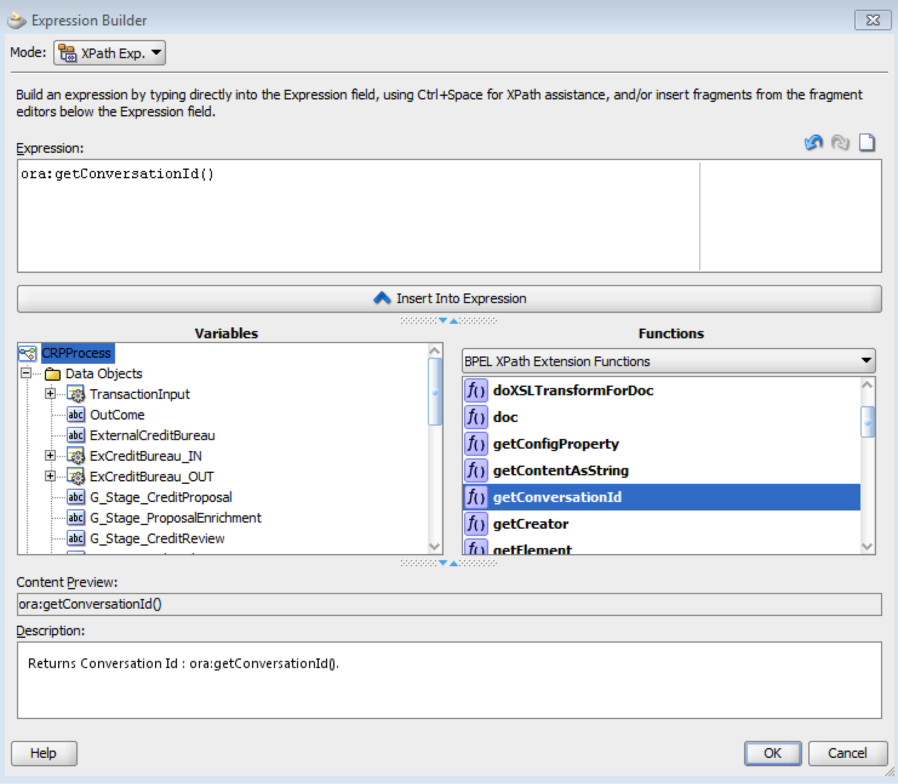
Mode: Simple Exp.

Build an expression by typing directly into the Expression field and/or insert fragments from the fragment editors below the Expression field.

Content Preview: contains(value : String, s : String) : Bool

Description: This function returns true if the first argument string contains the second argument string, and otherwise returns false.

Add the “ora:getConversationId()“ function from the BPEL Xpath Extension Function list



Expression Builder

Mode: XPath Exp.

Build an expression by typing directly into the Expression field, using Ctrl+Space for XPath assistance, and/or insert fragments from the fragment editors below the Expression field.

Expression:

```
ora:getConversationId()
```

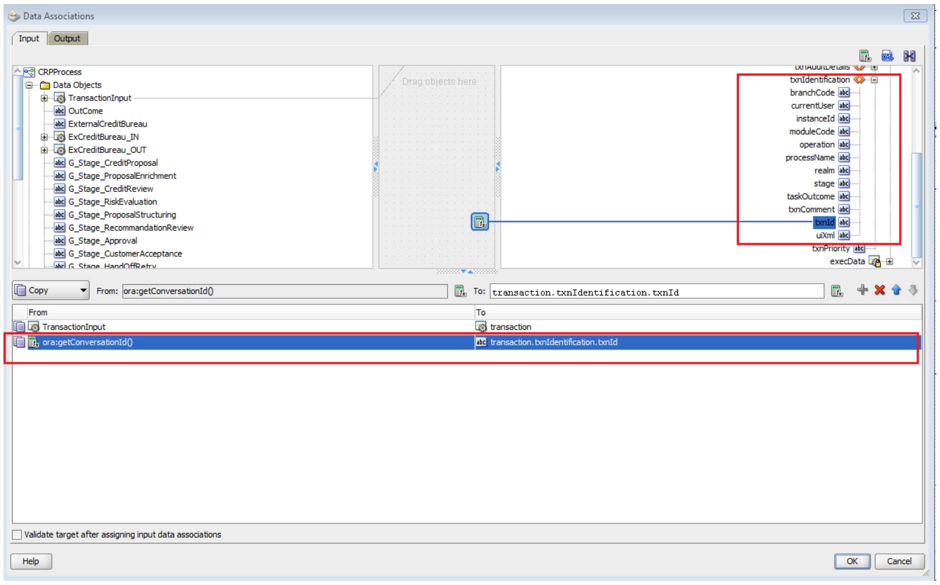
Functions: BPEL XPath Extension Functions

Content Preview: ora:getConversationId()

Description:

Returns Conversation Id : ora:getConversationId().

Finally Expression will be added in data association window.



Data Associations

Copy From: ora:getConversationId() To: transaction.txnIdentification.txnId

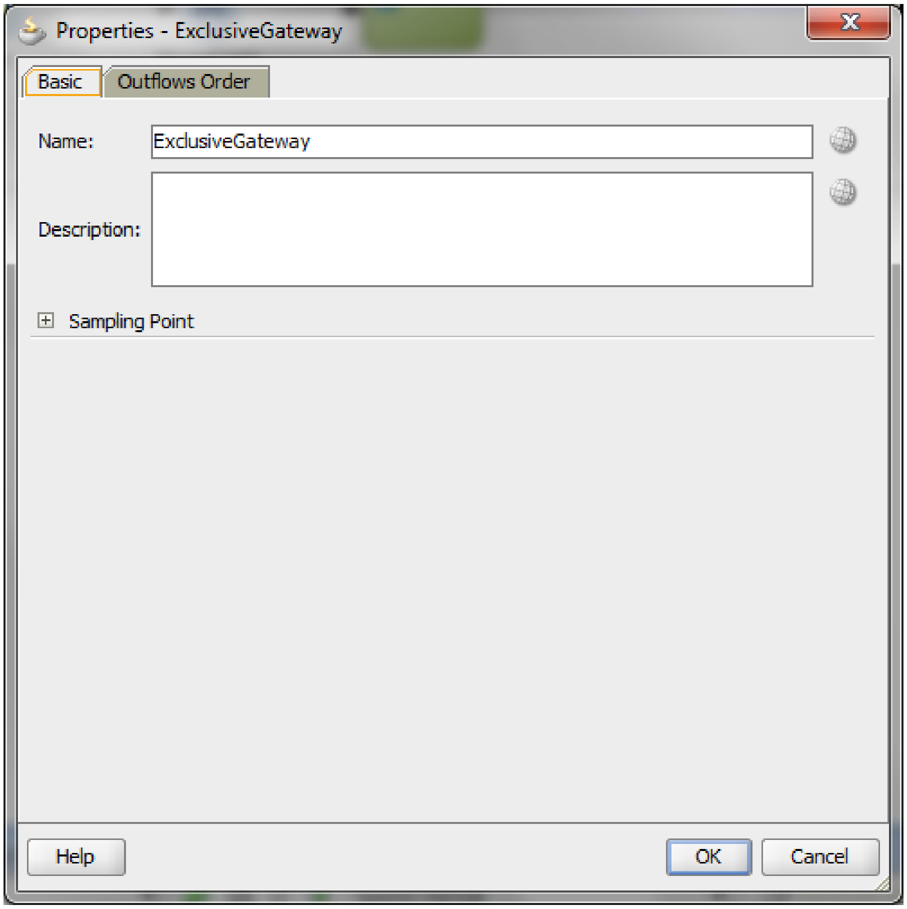
From	To
TransactionInput	transaction
ora:getConversationId()	transaction.txnIdentification.txnId

☐ Validate target after assigning input data associations

1.5.11 Adding Gateways to the Process

Step 15: Expand the **gateway pane** in the **Component Palette** and click and drag a **Exclusive gateway**, dropping it onto the sequence flow between **Appenty**(**humantask**) and **End events**.

Properties window opens if required change the name and click OK.



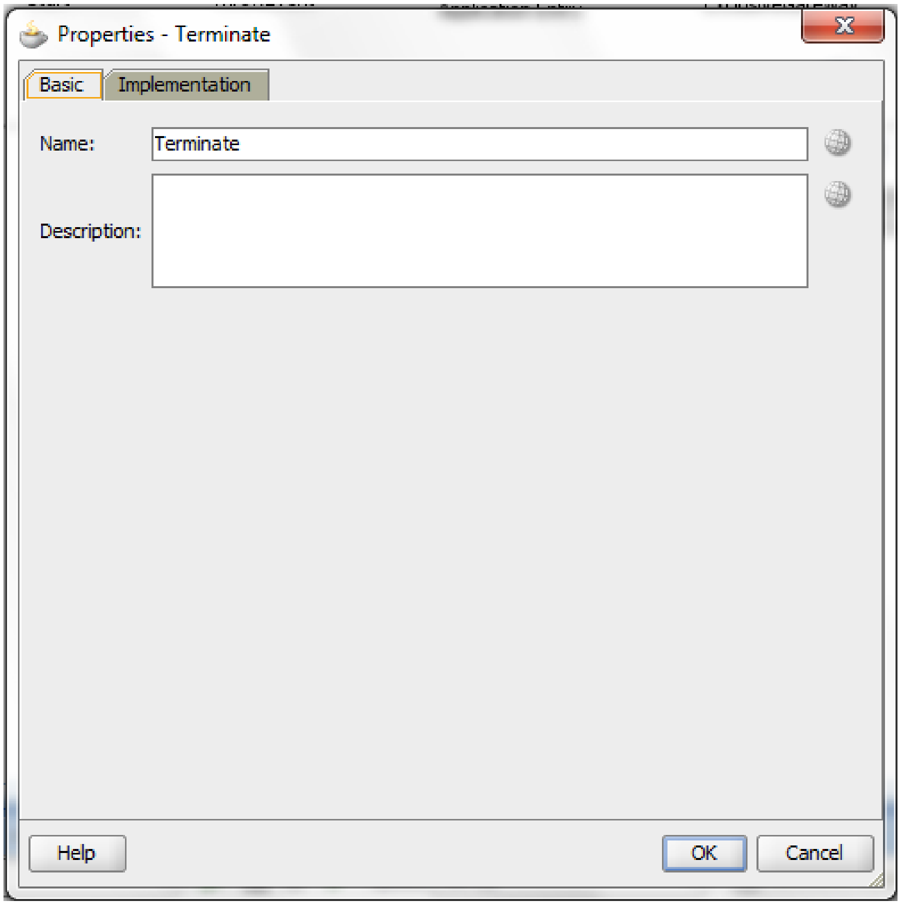
Properties - ExclusiveGateway

Name: ExclusiveGateway

Description:

Step 16: Expand the **Events pane** in the **Component Palette** and from the **End events** section, click and drag a **Terminate**, **dropping** it onto the Process editor.

Properties window opens if required change the name and click OK.

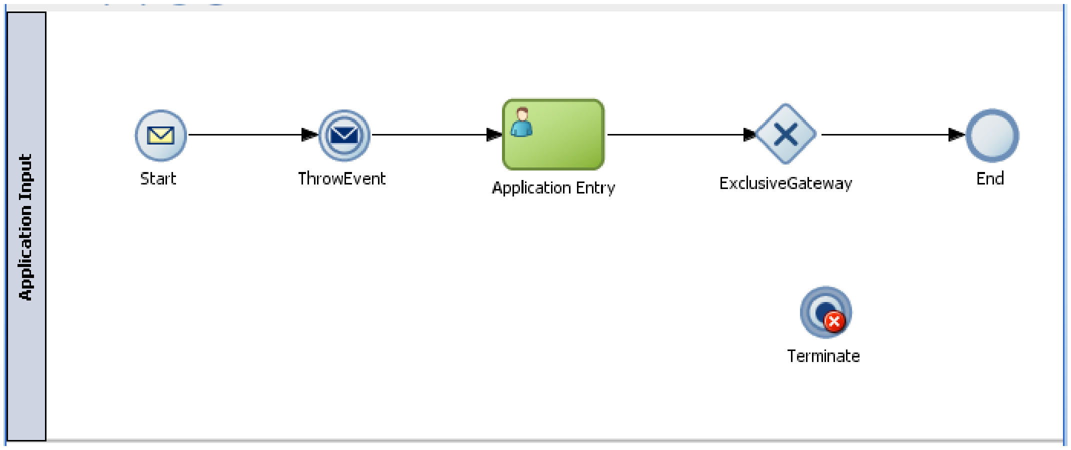


Properties - Terminate

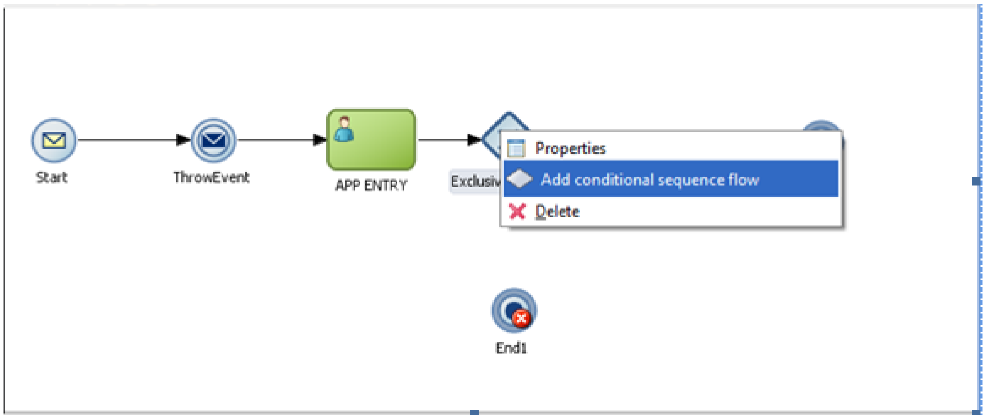
Name: Terminate

Description:

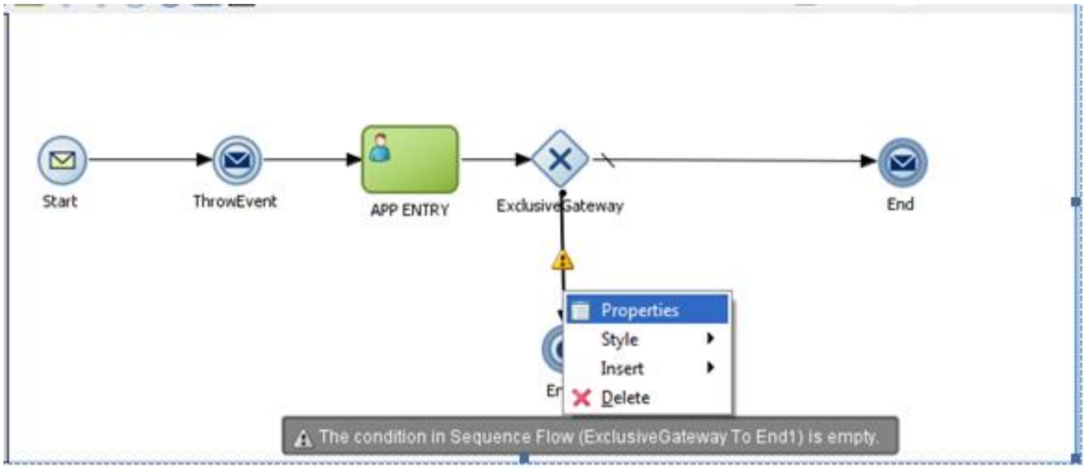
Now our Process looks likes this:



Right click on the **gateway** and select the **Add Conditional Sequence** flow, and connect the gateway to the **Terminate event**.

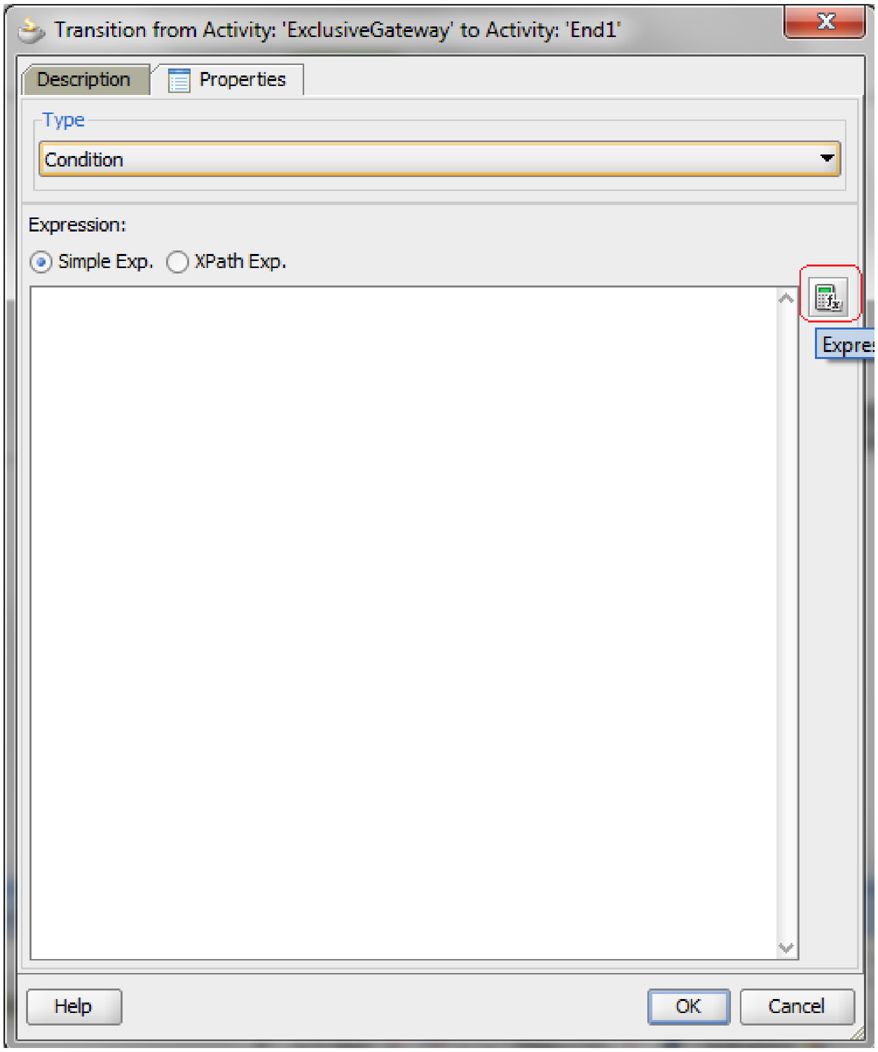


Right click on the **Sequence flow**, and select **properties**.



The condition in Sequence Flow (ExclusiveGateway To End1) is empty.

The **Sequence flow property** window will be opened, enter the name if required and go to properties tab, Click on the Expression Builder.



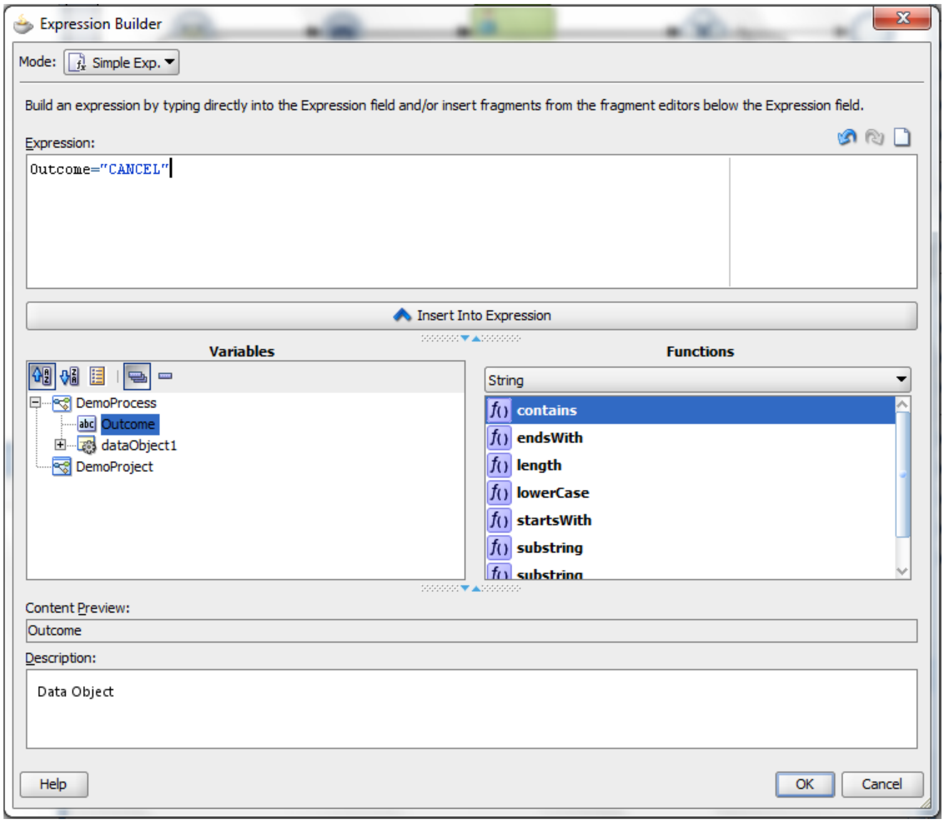
Transition from Activity: 'ExclusiveGateway' to Activity: 'End1'

Type: Condition

Expression:

● Simple Exp. ○ XPath Exp.

And Build the condition for the sequence flow by selecting the object from the list and click **insert into expression** or by dragging and drop the **object** in the expression tab.



Expression Builder

Mode: Simple Exp.

Build an expression by typing directly into the Expression field and/or insert fragments from the fragment editors below the Expression field.

Expression:

```
Outcome="CANCEL"
```

Content Preview: Outcome

Description: Data Object

Click Ok and again Click OK.

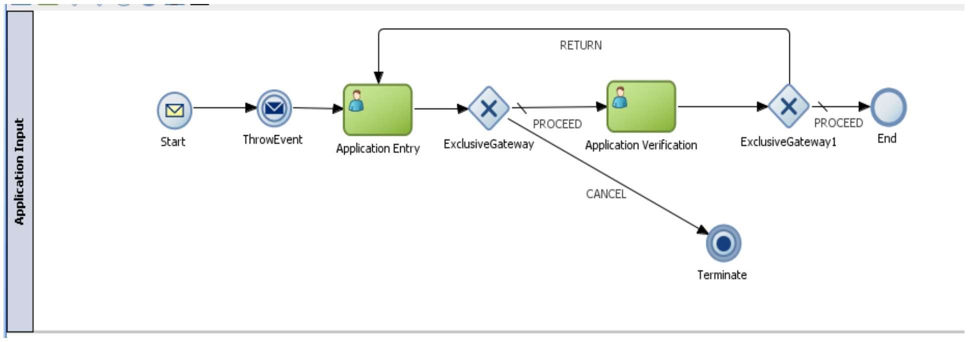
Click Save All.

Create another user activity and implement the humantask properties.(follow the same steps as done for the useractivity (**Application Entry**)).

Create another gateway event and connect the conditional end to the first human task(app entry).

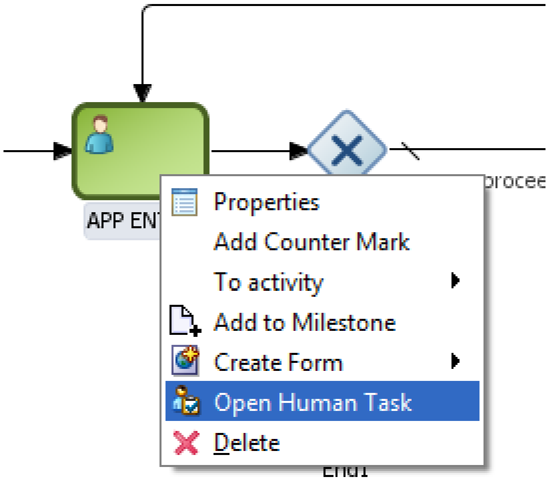
Click Save all.

Now the process looks like this:

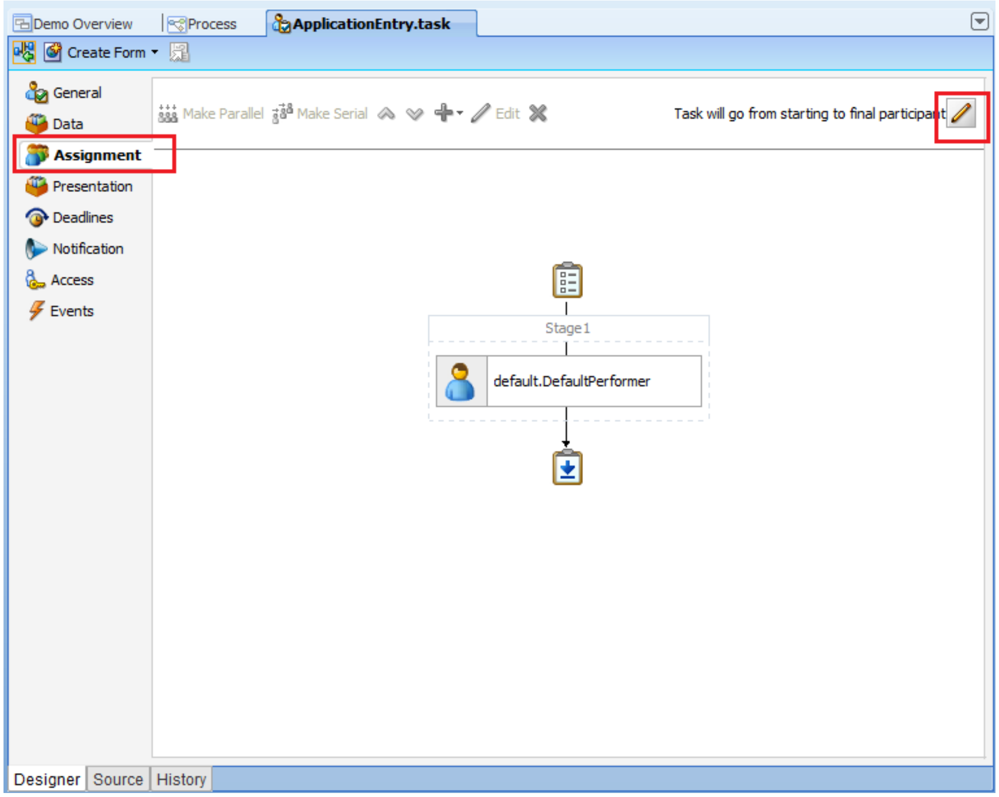


1.5.12 Mapping Oracle FLEXCUBE Roles to Human Task

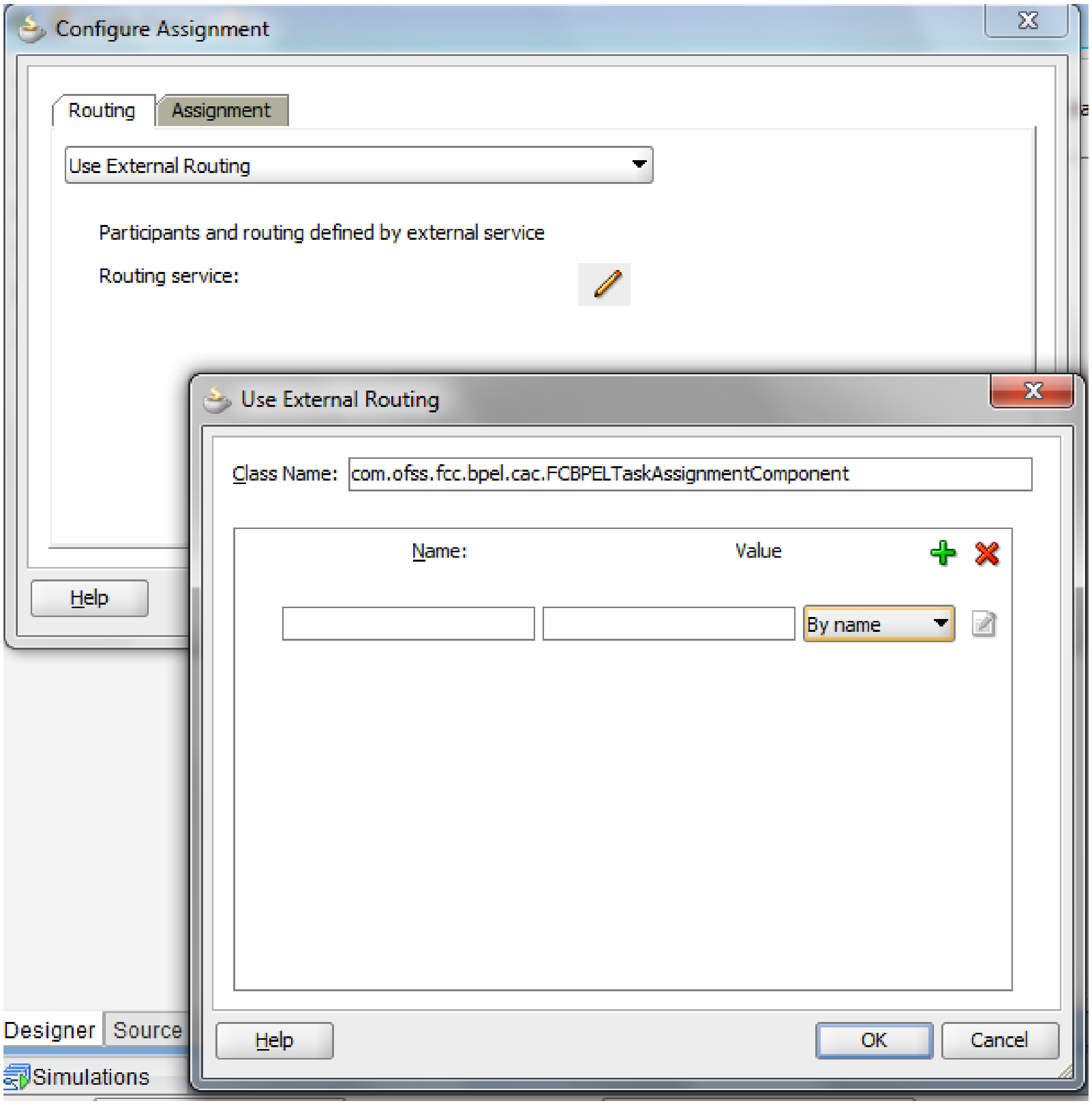
Step 17: Right click on the **human task** and select Open **Human Task**, it opens in new tab.



Click Assignment tab and click Edit Icon.



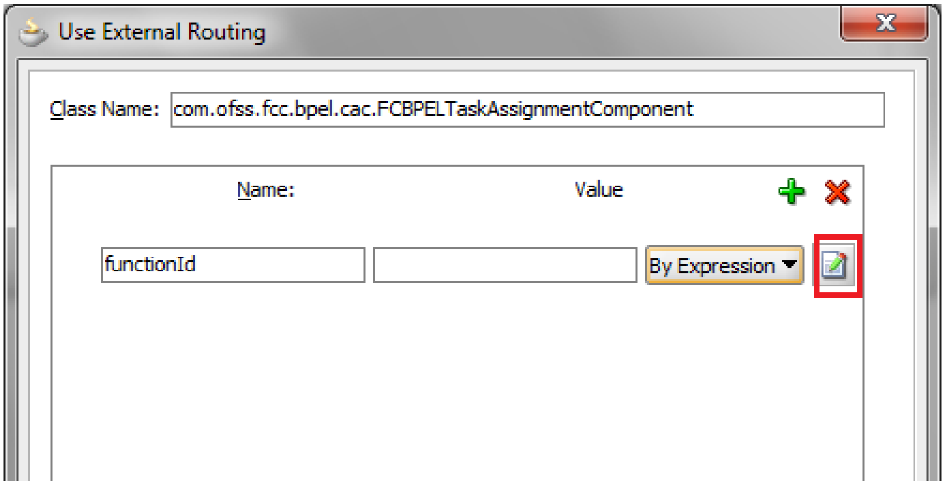
Configure Assignment window is opened, select Use External Routing from the select box and click on edit Icon, Use **External Routing** window will be opened, now enter the class name as **(com.ofss.fcc.bpel.cac.FCBPELTaskAssignmentComponent).**



Class Name: com.ofss.fcc.bpel.cac.FCBPELTaskAssignmentComponent

Enter the **name** and select by **expression** from the select box and click on Edit Icon beside it.

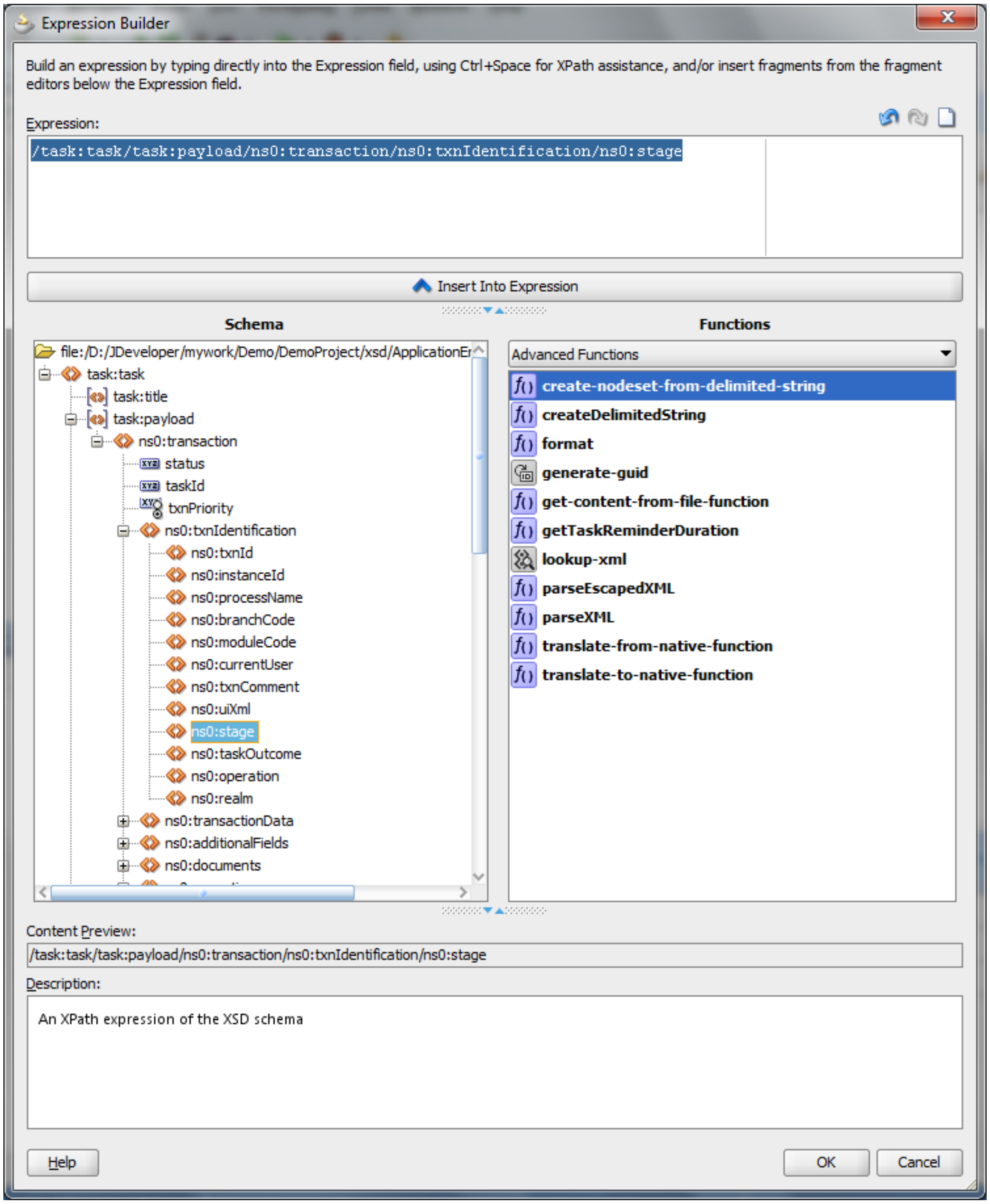
STOP The name specified here is Case sensitive.



Class Name: com.ofss.fcc.bpel.cac.FCBPELTaskAssignmentComponent

Name:	Value	
functionId		By Expression

Select the stage from the **task**:**payload** and Click insert into expression and Click OK.



Expression Builder

Build an expression by typing directly into the Expression field, using Ctrl+Space for XPath assistance, and/or insert fragments from the fragment editors below the Expression field.

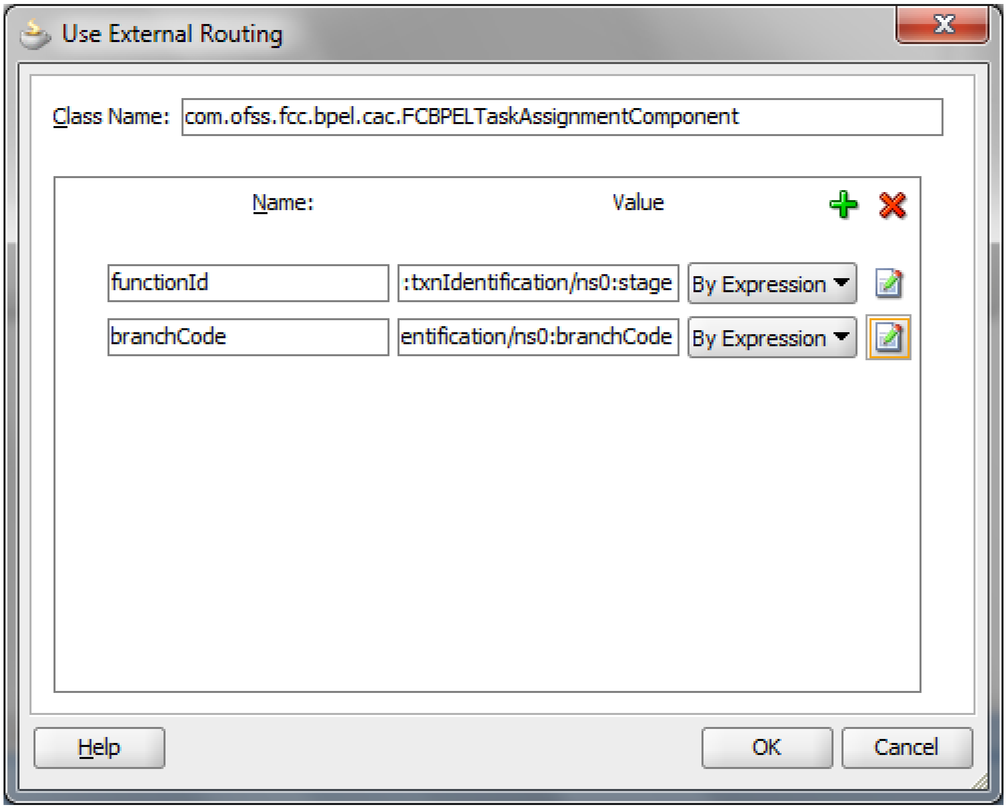
Expression:

```
/task:task/task:payload/ns0:transaction/ns0:txnIdentification/ns0:stage
```

Content Preview: /task:task/task:payload/ns0:transaction/ns0:txnIdentification/ns0:stage

Description: An XPath expression of the XSD schema

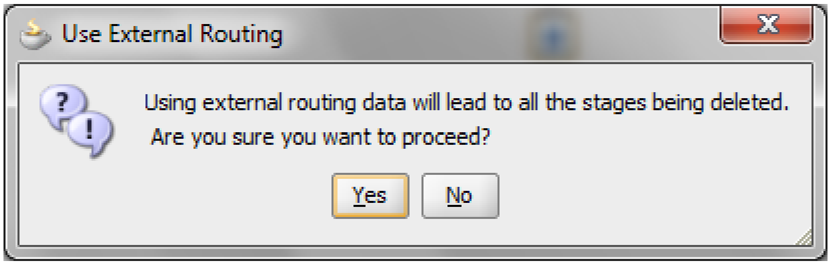
Add another element and likewise map the **branchcode** and click ok .



Class Name: com.ofss.fcc.bpel.cac.FCBPELTaskAssignmentComponent

Name:	Value	
functionId	:txnIdentification/ns0:stage	By Expression
branchCode	entification/ns0:branchCode	By Expression

Click ok



Using external routing data will lead to all the stages being deleted.
Are you sure you want to proceed?

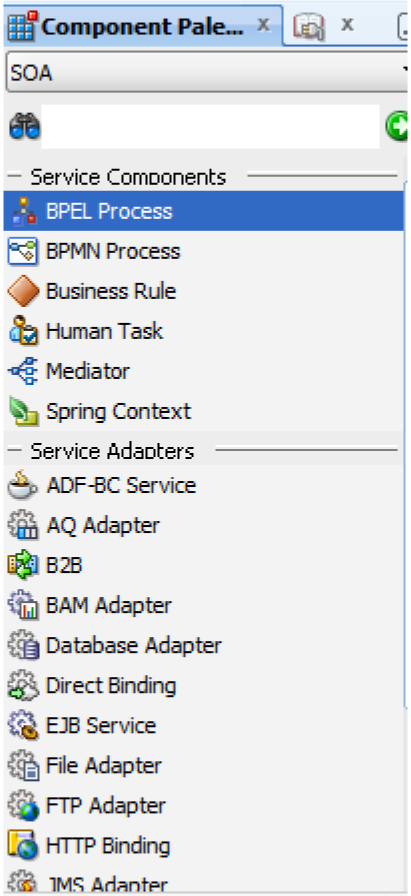
Click Yes and Save all.

Repeat the same steps for every human task in the process.

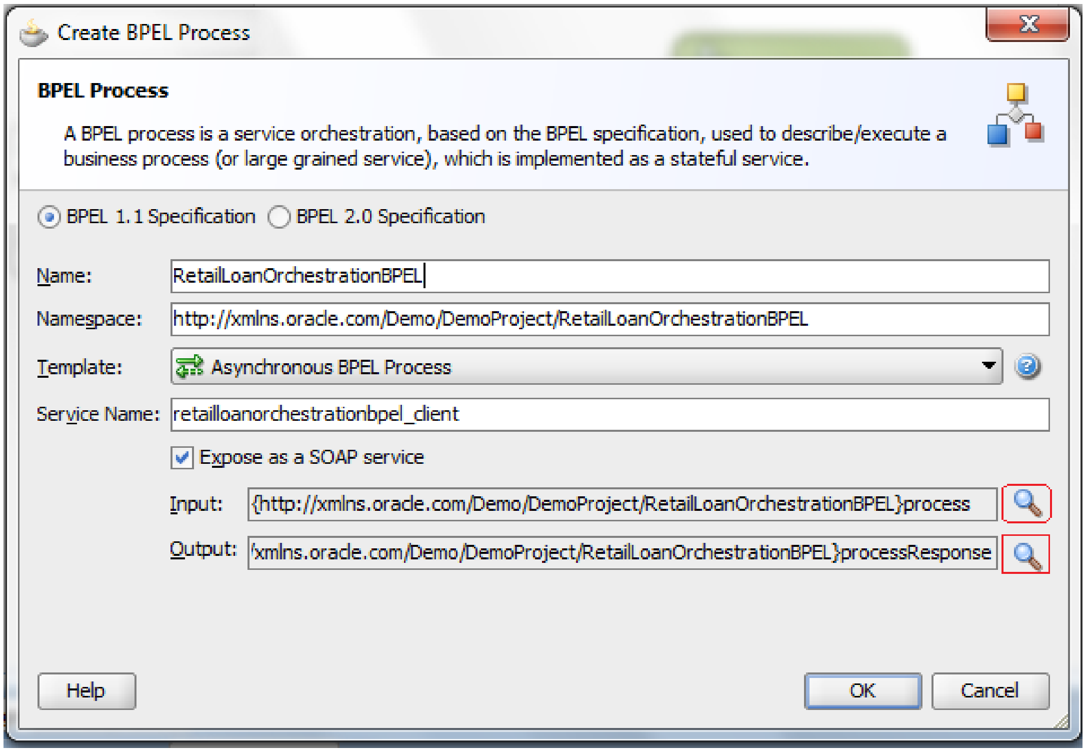
1.5.13 Creating and Implementing system tasks

Step 18: Drag and Drop BPEL Process to the Composite.xml

STOP A Service Call can be **Implemented** by **Service Adapters** also.



A Window appears Rename the Process with Desired name ex: **RetailLoanOrchestrationBPEL.**



Create BPEL Process

BPEL Process

A BPEL process is a service orchestration, based on the BPEL specification, used to describe/execute a business process (or large grained service), which is implemented as a stateful service.

● BPEL 1.1 Specification ○ BPEL 2.0 Specification

Name: RetailLoanOrchestrationBPEL

Namespace: http://xmlns.oracle.com/Demo/DemoProject/RetailLoanOrchestrationBPEL

Template: Asynchronous BPEL Process

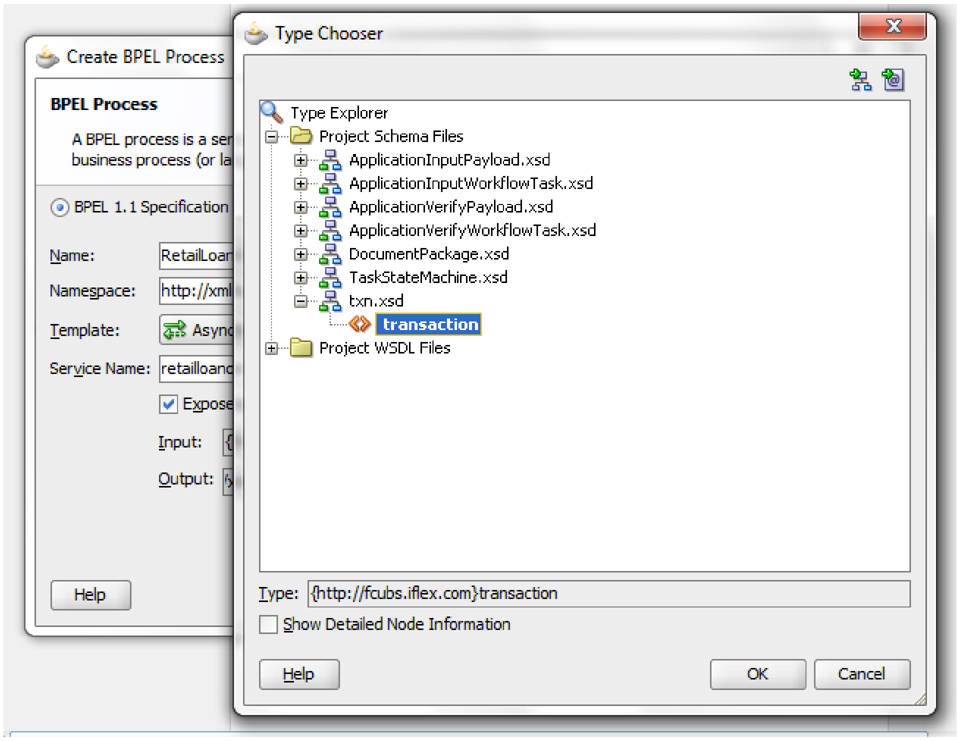
Service Name: retailloanorchestrationbpel_client

☑ Expose as a SOAP service

Input: {http://xmlns.oracle.com/Demo/DemoProject/RetailLoanOrchestrationBPEL}process

Output: /xmlns.oracle.com/Demo/DemoProject/RetailLoanOrchestrationBPEL}processResponse

Click the **Input** and **Output** and chose the element type.



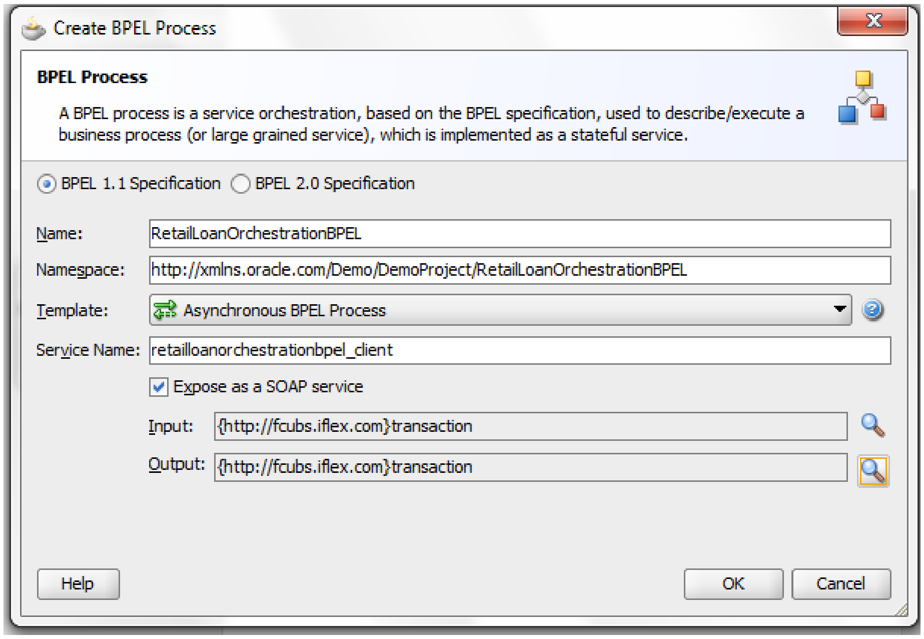
Type Chooser

Type Explorer
- Project Schema Files
  - ApplicationInputPayload.xsd
  - ApplicationInputWorkflowTask.xsd
  - ApplicationVerifyPayload.xsd
  - ApplicationVerifyWorkflowTask.xsd
  - DocumentPackage.xsd
  - TaskStateMachine.xsd
  - txn.xsd
    - transaction
- Project WSDL Files

Type: {http://fcubs.iflex.com}transaction

□ Show Detailed Node Information

BPEL Process screen is as follows:



Create BPEL Process

BPEL Process

A BPEL process is a service orchestration, based on the BPEL specification, used to describe/execute a business process (or large grained service), which is implemented as a stateful service.

● BPEL 1.1 Specification ○ BPEL 2.0 Specification

Name: RetailLoanOrchestrationBPEL

Namespace: http://xmlns.oracle.com/Demo/DemoProject/RetailLoanOrchestrationBPEL

Template: Asynchronous BPEL Process

Service Name: retailloanorchestrationbpel_client

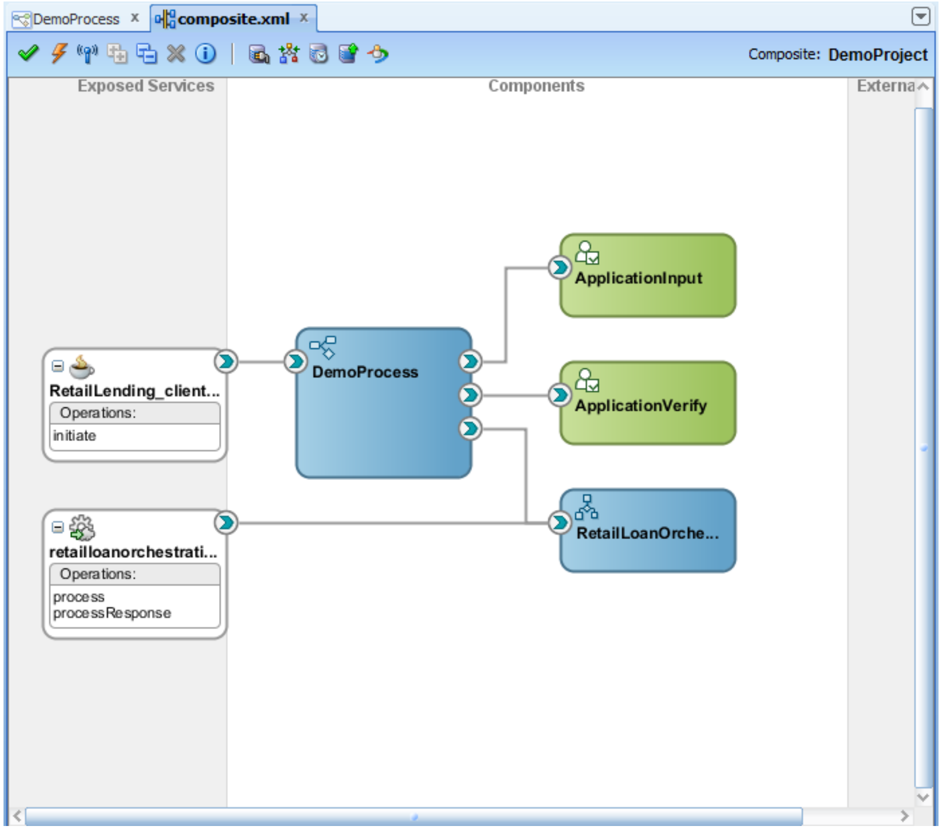
☑ Expose as a SOAP service

Input: {http://fcubs.iflex.com}transaction

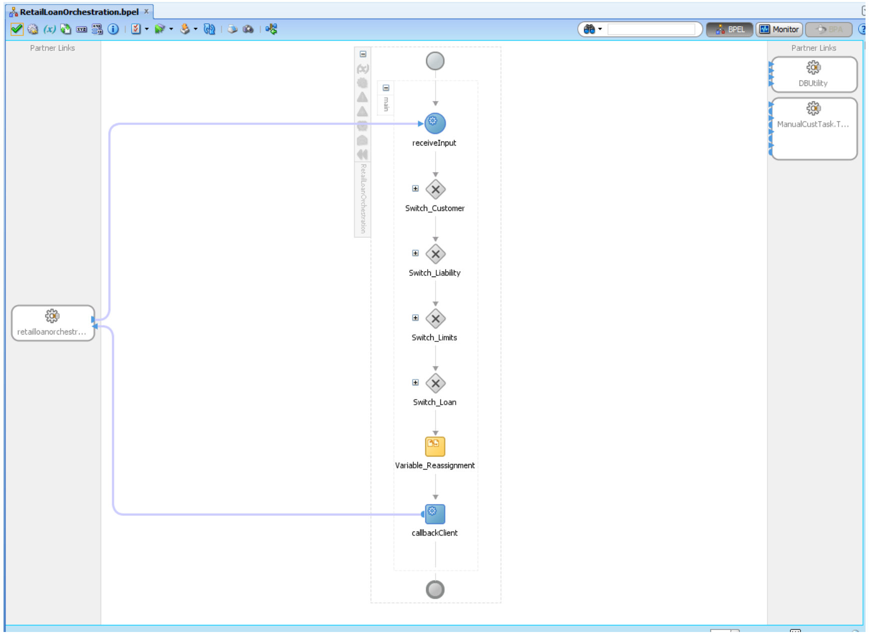
Output: {http://fcubs.iflex.com}transaction

Click Ok

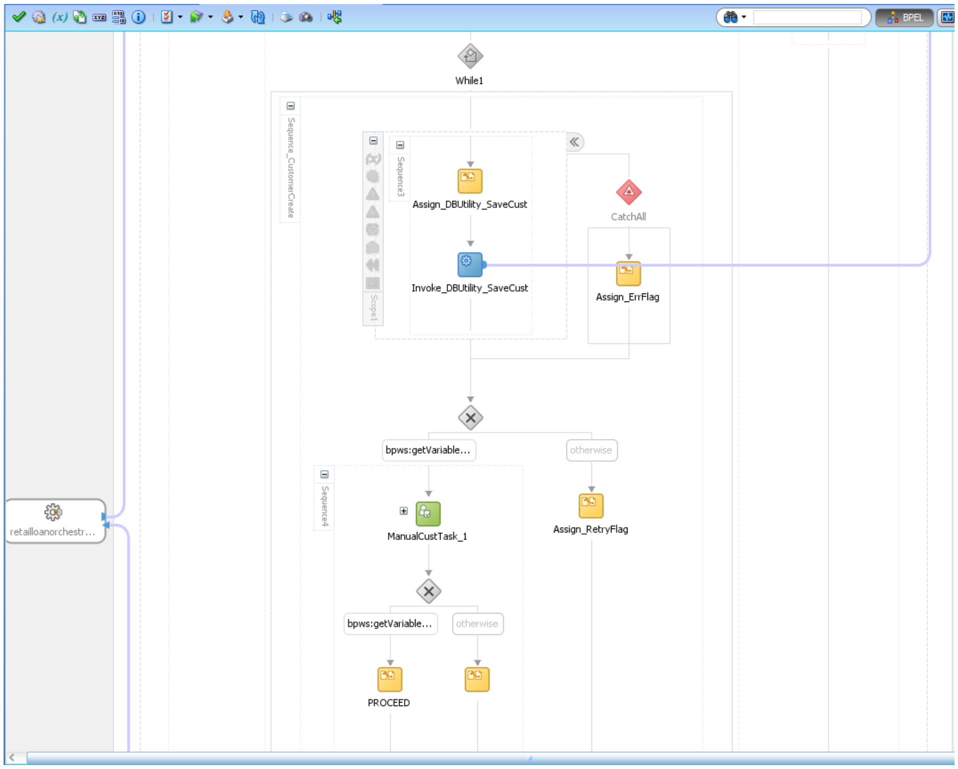
Now the **Composite.xml** looks like this



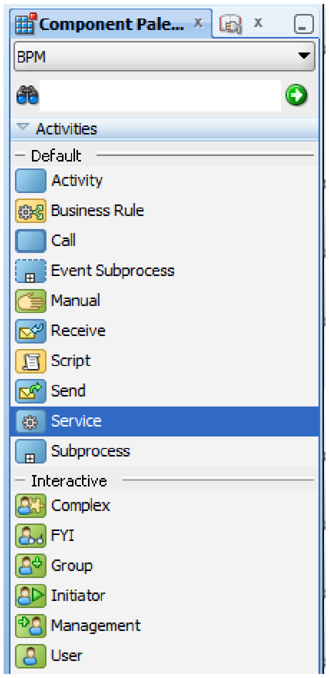
The **RetailLoanOrchestration**.**bpel** looks like this after Implementation.



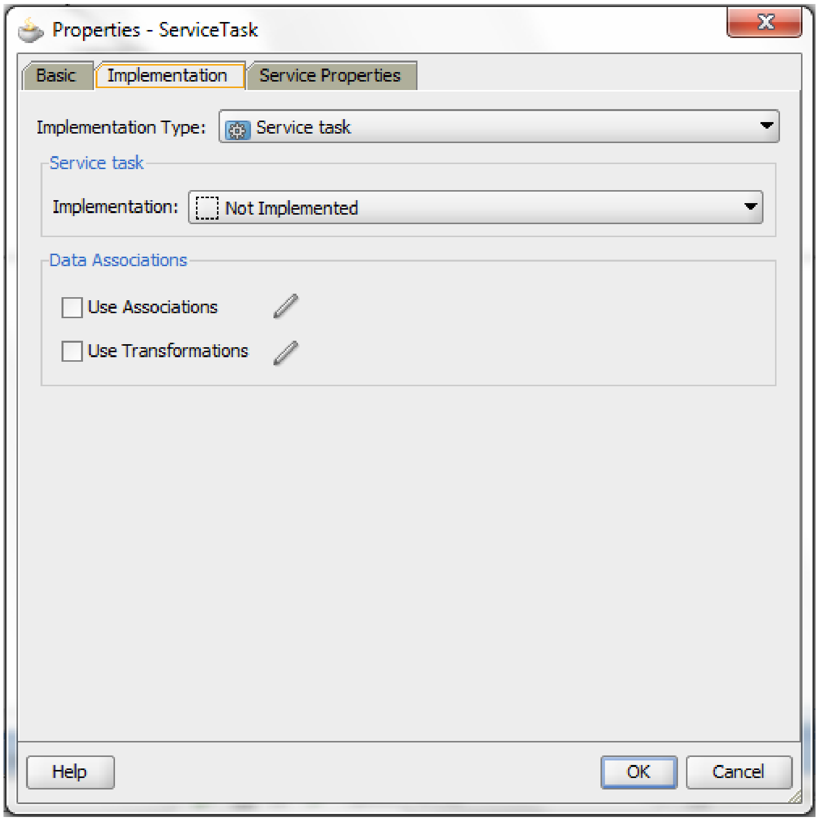
In each **Switch node** of **RetailLoanOrchestration**.bpel process **DButility Bpel process** is called if it fails a **manual retry task** is initiated to book the **RetailLoan**



Now go to the Process and add a **Service Task** from **component palette.**



Window appears as below:


Properties - ServiceTask

Basic | Implementation | Service Properties

Implementation Type: Service task

Service task

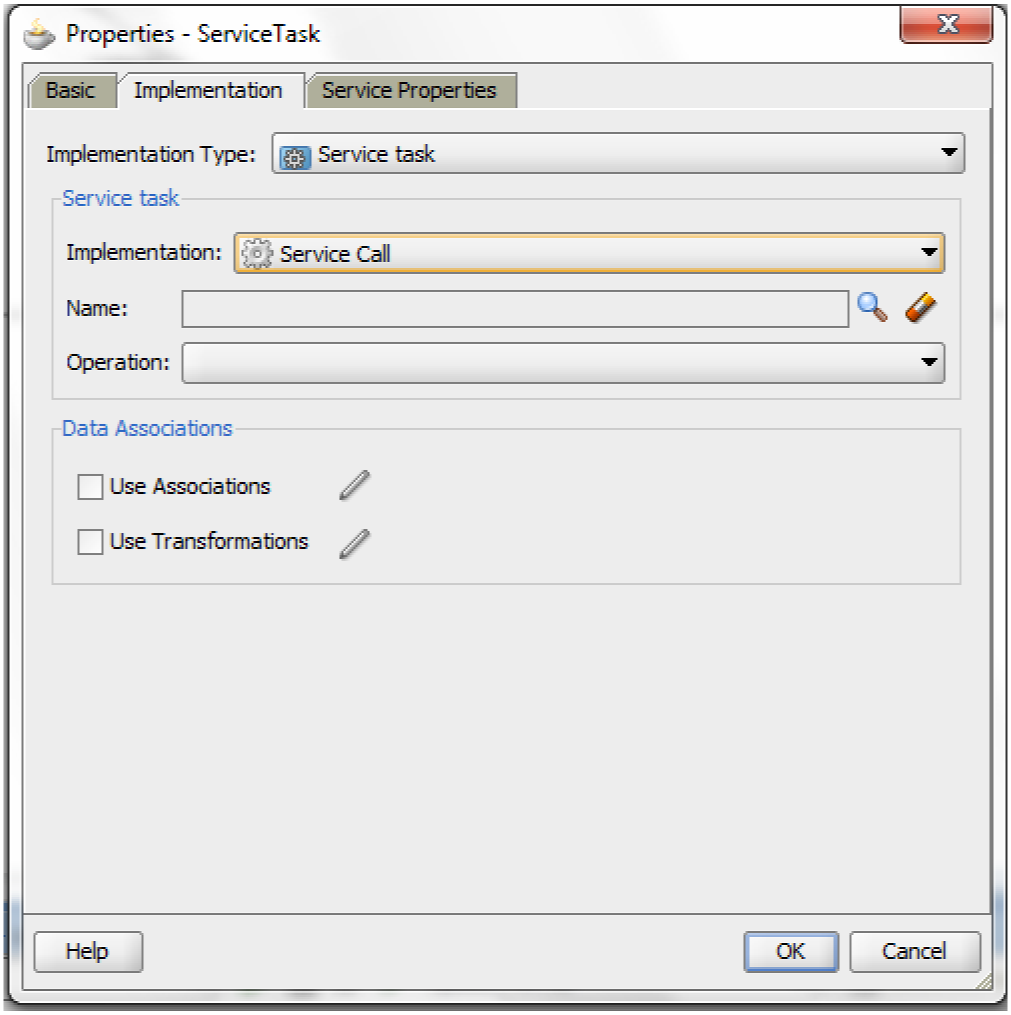
Implementation: Not Implemented

Data Associations

□ Use Associations

□ Use Transformations

Now Select the **Implementation type** as **Service Call** and click the 🔍 icon.



Properties - ServiceTask

Implementation Type: Service task

Service task

Implementation: Service Call

Name:

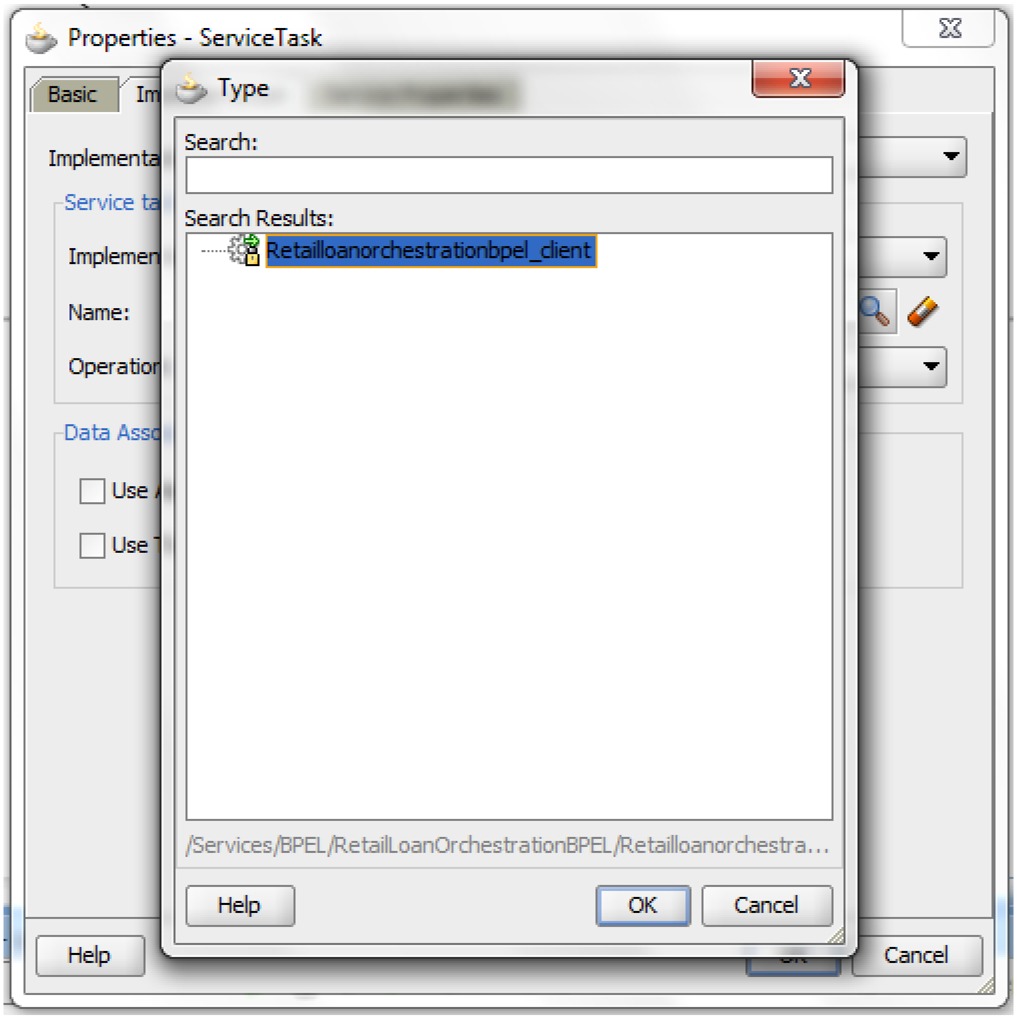
Operation:

Data Associations

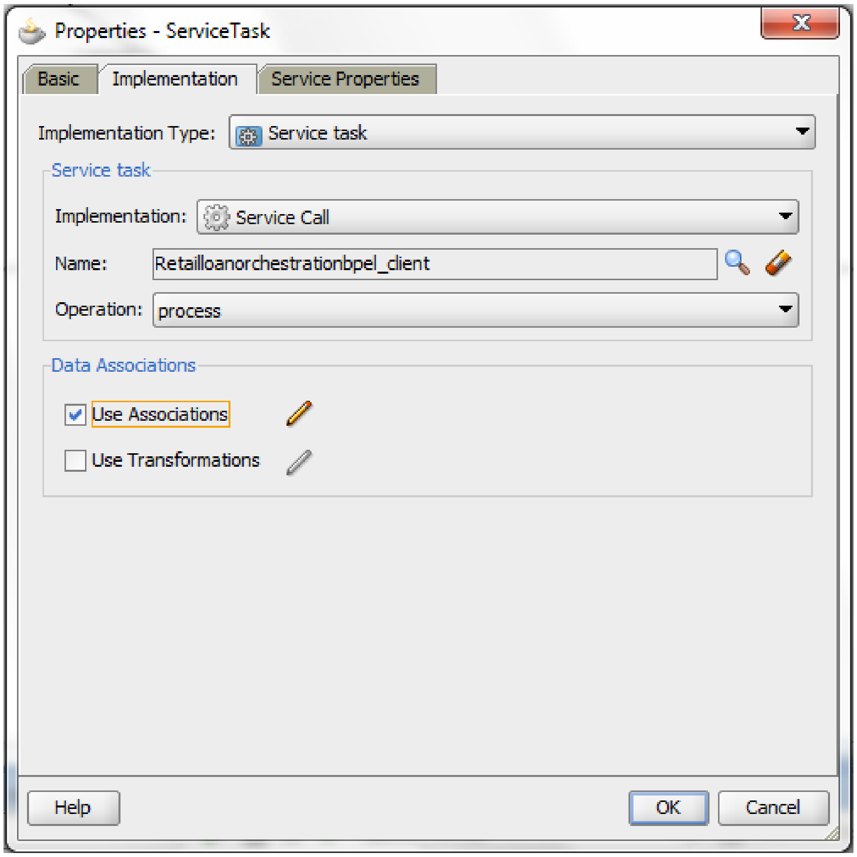
□ Use Associations

□ Use Transformations

Window appears as below from that select the service **RetalLoanOrchestrationBPEL** which is listed.



Click **Use association** check box and ✎ icon.



Implementation Type: Service task

Implementation: Service Call

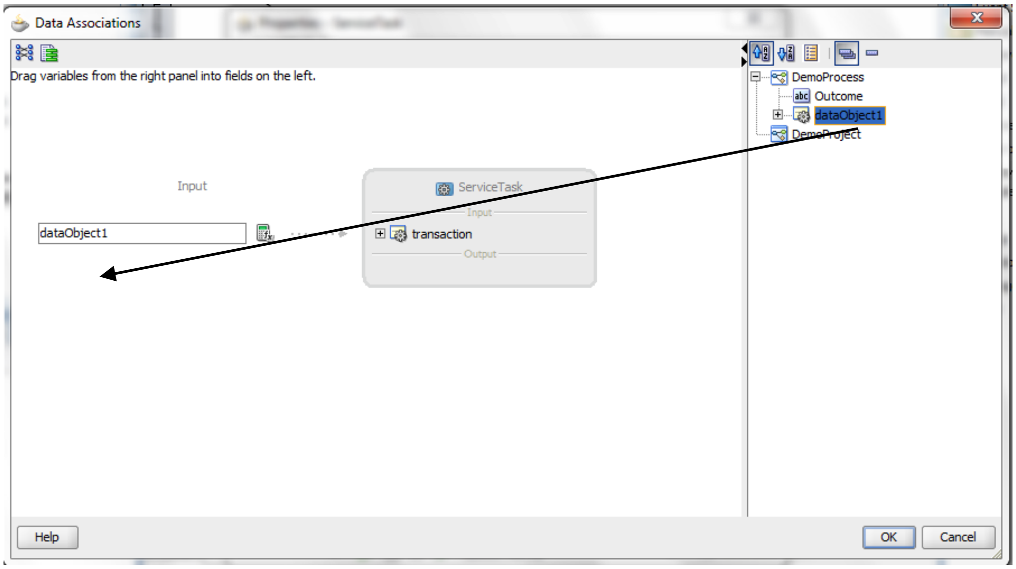
Name: Retailloanorchestrationbpel_client

Operation: process

☑ Use Associations

□ Use Transformations

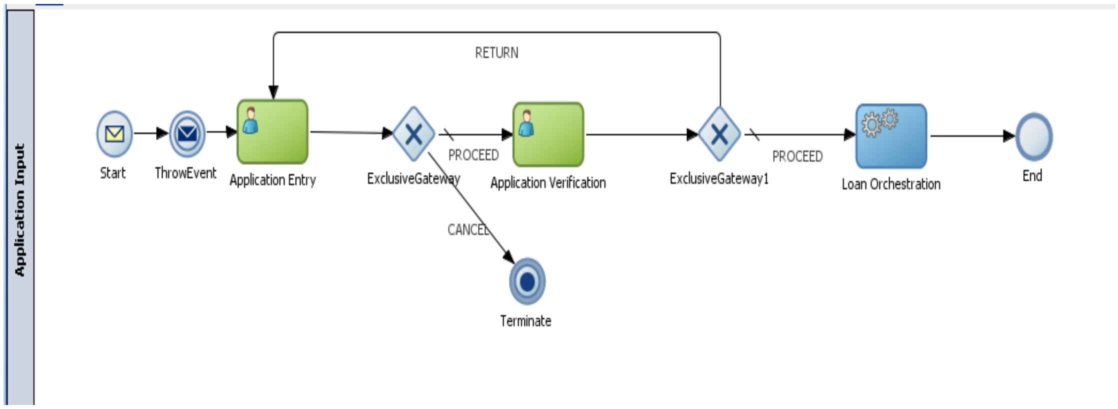
Map the **data association** and click ok.



Drag variables from the right panel into fields on the left.

Input: dataObject1

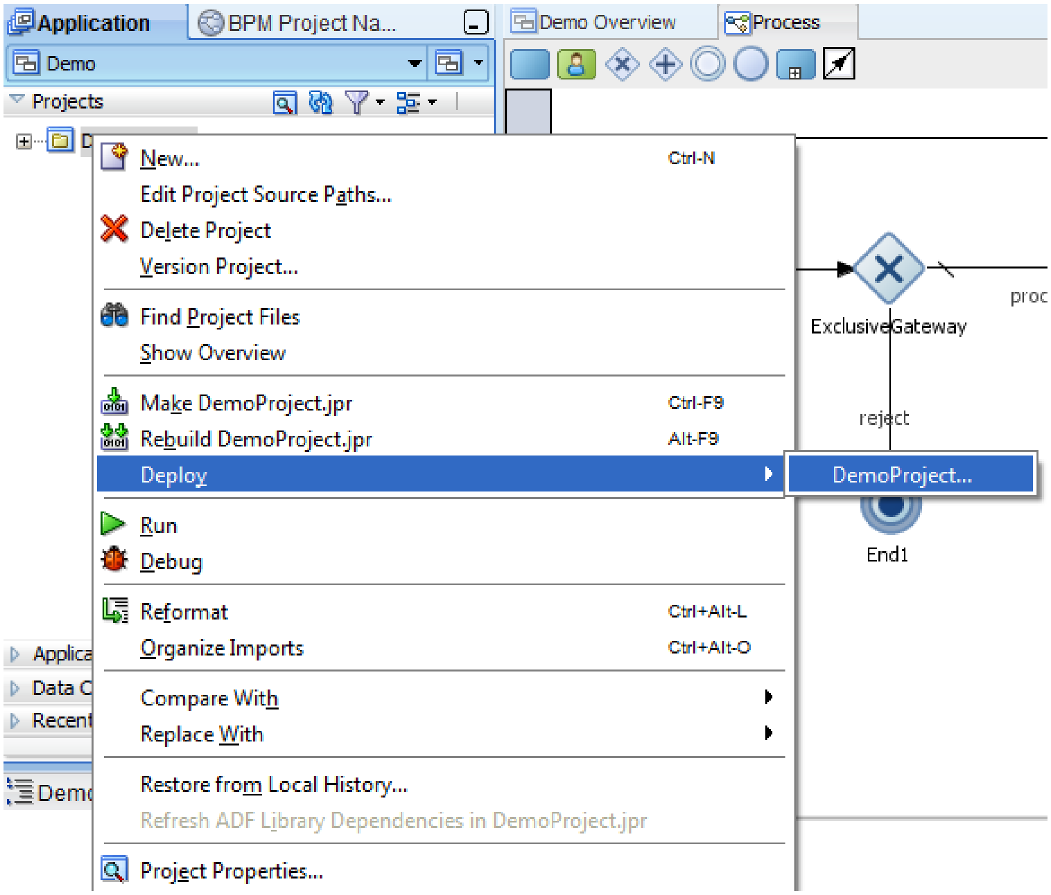
Now the process looks like this:



1.6 Deploying the Process

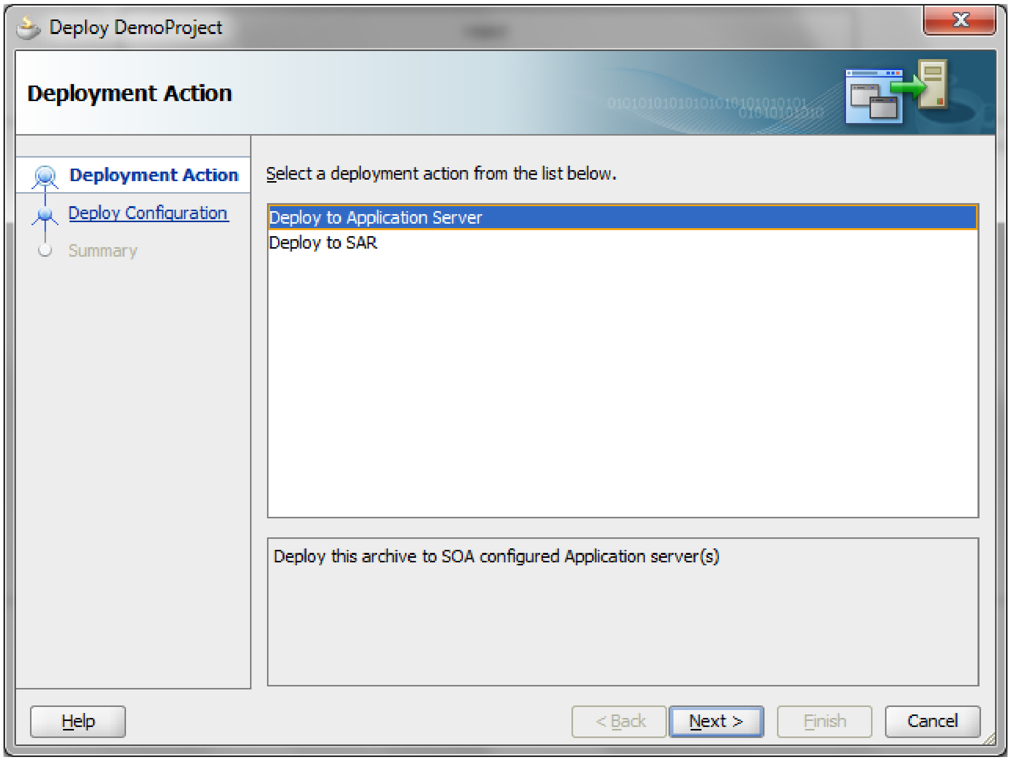
Step 19: Deploy the **DemoProject**.

In the **Application Navigator**, right click **DemoProject** and select **Deploy > DemoProject**.



The Deploy **DemoProject** wizard opens.

In the **Deployment Action** page of the wizard, select Deploy to Application Server and click **Next**.



Select a deployment action from the list below.

Deploy to Application Server

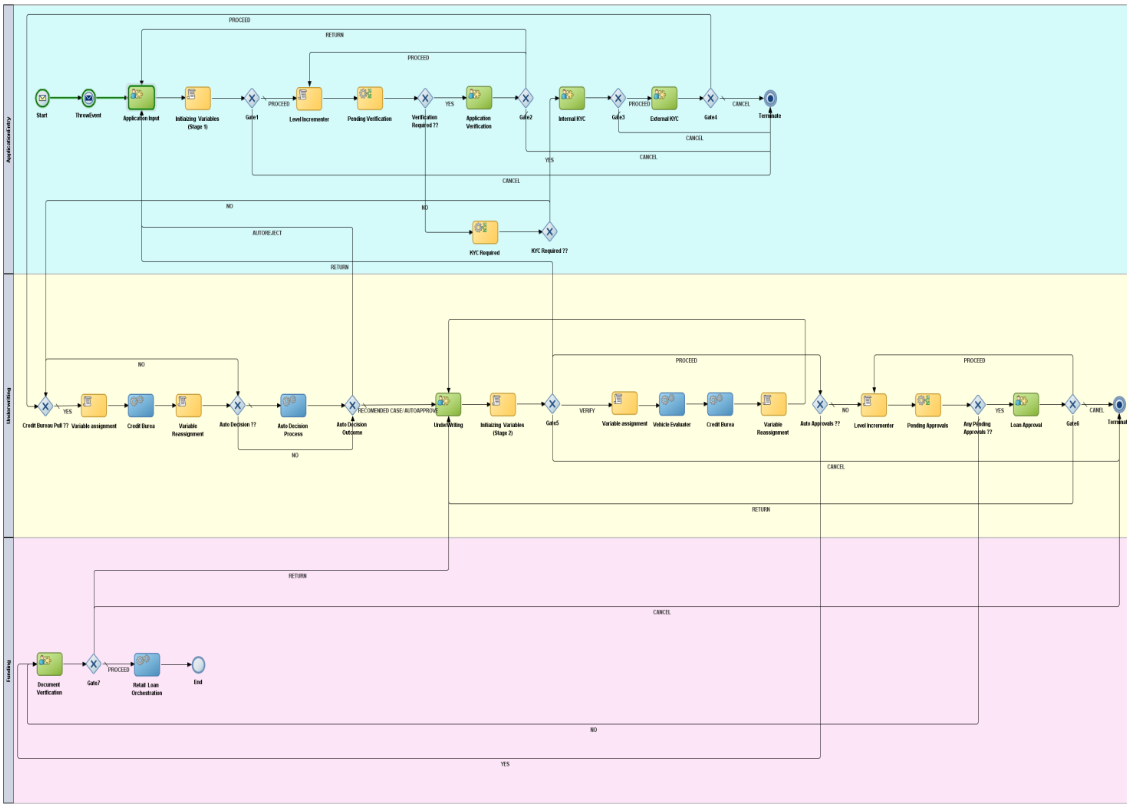
Deploy to SAR

Deploy this archive to SOA configured Application server(s)

In the **Deploy Configuration** page, click the **Overwrite any existing** composites with the same revision IDcheckbox and click Next.

1.7 Retail Lending BPMN Process

1.7.1 Retail Lending BPMN Process Flow Diagram



1.7.2 Guidelines followed in Retail Lending Process flow

- single data object named **transactioninput** which contains the **transaction.xsd**
  txn.xsd
  is used throughout the Process Flow to map the Inputs and Outputs.
- Global Fault Handling Standard are maintained across the Process Flow.
- The Fault Handling is done separately in another BPEL Process called **RetalLoanOrchestrationBPEL.**
- **Swimlanes** are used to differentiate the different Stages in the Process.
- Usage of **Flexcube_interface.wsdl** Flexcube_interface.wsdl to initiate the Process rather than using other wsdl files.
- **com.ofss.fcc.bpel.cac.FCBPELTaskAssignmentComponent** class is used in all Human Tasks to map the Flexcube Roles .

- Naming Conventions are followed as per the Retail Lending Flow Diagram.
- The DBUtility Call and Manual Retry Task are done separately in RetalLoanOrchestrationBPEL BPEL Process.

1.7.3 Naming Conventions Followed in Retail Landing Process Flow

Processes

- RetailProcess(BPM)
- RetailLoan OrchestrationBPEL(BPEL)

RetailProcess(BPM)

Human Tasks:

- ApplicationInput
- ApplicationVerify
- InternalKYCTask
- ExternalKYCCkeck
- UnderWriting
- UnderWriting_Approval
- FinalVerification

Business Rules:

- VerifyAppRules
- KYCRules
- LoanApprovalRules

Exposed Services:

- RetailLending_client_ep

External Reference:

- VehicleEvaluater
- CreditBureau
- SelectDecisionDBAdapter

Task Name/Event Name	Input	Output
Start Event	-	**transactioninput**
ThrowEvent	**transactioninput**	-
User Tasks(all)	**transactioninput**	**transactioninput**
VerifyAppRules	**VRule_IN**	**VRule_ OUT**
KYCRules	**KYC_IN**	**KYC_ OUT**

LoanApprovalRules	**UnderWrite_IN**	**UnderWrite_ OUT**
Credit Burea	**ExCreditBureau_IN**	**ExCreditBureau_OUT**
Auto Decision Process	**AutoDecision_IN**	**AutoDecision_OUT**
Vehicle Evaluater	**VEvaluator_IN**	**VEvaluator_OUT**

OUTCOME	String
ExternalCreditBureau	String
AutoDecisionReqd	String
AutoDecisionOutput	String

RetailLoan OrchestrationBPEL(BPEL)

Human Tasks:

- ManualCustTask

Exposed Services:

- retailloanorchestration_client_ep

External Reference:

- DBUtility

Task Name/Event Name	**Input**	**Output**
RetailLoan OrchestrationBPEL	**inputvariable**	**outputvariable**
DBUtility	**Invoke_DBUtility_SaveCust_initiate_InputVariable_1**	**Invoke_DBUtility_SaveCust_initiate_OutputVariable**

1.8 Acronyms and Abbreviations

RL	Retail Lending
BPMN	Business Process Model and Notation
BPEL	Business Process Execution Language
SOA	Service-Oriented Architecture

1.9 References

Retail Loan Origination Oracle FLEXCUBE Universal Banking Release 12.0 [May] [2012]

http://docs.oracle.com/cd/E14571_01/doc.1111/e15176/model_bus_procs_bpmpd.htm

BPMN Process Flow Definition Guide
[December] [2017]
Version 12.0.4.7.11

Oracle Financial Services Software Limited
Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India

Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax:+91 22 6718 3001
www.oracle.com/financialservices/

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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 failsafe, 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.

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.

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.