

Building Your First Process with Oracle BPM 11g

This tutorial contains the following sections:

Purpose	
Time to Complete	
Overview	
<u>Scenario</u>	
Software and Hardware Require	rements
<u>Prerequisites</u>	
	Application
Enhancing the Basic Hello Wor	rld Process
E Deploying and Testing the App	<u>lication</u>
<u>Summary</u>	
<u>Resources</u>	

Purpose

This tutorial shows you how to build a simple Hello World application using Oracle BPM Suite 11gR1. It also shows you how to deploy the process to the BPM engine and test it in the BPM Workspace.

Time to Complete

Approximately 2 hours

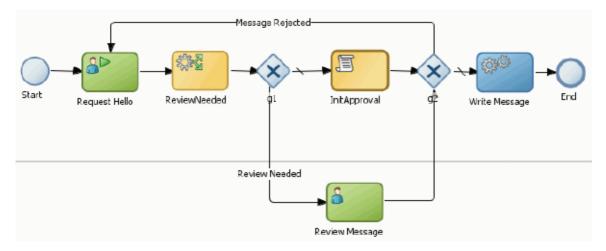
Overview

In this tutorial, you use Studio, the JDeveloper based IDE, to create a simple Hello World process. This process demonstrates the use of a file service, interactive tasks implemented by the human workflow engine, and by conditional branching. The conditions for the conditional branching are determined through the use of a business object and a business rule. You also use a script task to initialize a variable. After building the process, you deploy it to the BPM engine and test it in the runtime environment.

Scenario

There are two roles involved in the Hello World process, the sender of the message, acting in the Requester role, and a reviewer, acting in the Reviewer role. The requester is prompted, through the Request Hello activity, to enter a Hello message, greeting, and a date for the message. After the form is submitted, a business rule is applied to the message content to determine whether the message requires a review, based on the length of the greeting and message fields.

If the message does not require review, the process flows to a script task, which initializes a variable needed by the next task, then the message is sent to the Write Message activity to be written to the file system. If the message requires review, the reviewer is prompted to review the message and either accept or reject it. If the message is rejected, it returns to the Request Hello activity so that the requester can correct the message, otherwise, it goes directly to the Write Message activity for file processing.



Software and Hardware Requirements

In order to perform this tutorial, you must have previously installed Oracle BPM 11gR1 and JDeveloper 11.1.1.3 with both the SOA and BPM extensions. You will also need to have at least one user in the internal LDAP database of your WebLogic server in the OBPM installation in order to map this user to the roles that you define in your Hello World process. You can take care of both of these tasks (installation and seeding of the LDAP database) by performing the Installing Oracle BPM 11g OBE.

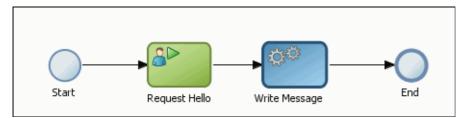
Prerequisites

If you have not yet installed OBPM 11gR1, perform the **Installing OBPM 11g** OBE. Performing this OBE will also seed your LDAP database.

If you already have an OBPM 11gR1 installation, but still wish to seed the Demo Community in the WebLogic server's internal LDAP database, download the <u>zip file</u> containing ANT files needed to perform this task. This is available from OTN as a **SOA 11g Human Workflow** sample code download. You will need to modify some parameters in the ANT build file to match your particular installation. A ReadMe file is included in the zip file to assist you.

Creating the Basic Hello World Application

In this section you create the basic starting point for the Hello World process using the JDeveloper Studio. You add two activities - an interactive activity and a service activity. The end user will be able to enter a Hello message, using the BPM Workspace. The message will be captured in a business object and passed to a file service, which, in turn will write the message to a disk file. Later, you expand upon this to add more complexity to the process.



You create several process elements throughout this section of the tutorial. The following naming convention will be used throughout this section:

Name	Description
HelloWorld_OBE	Application name
HelloWorldProject	Main project name
HelloWorld_UI	Project containing user task web form(s)
HelloWorldProcess	Process name

<u>Creating the Process Model</u> <u>Creating the Business Object</u> Implementing the User Task Implementing the File Service

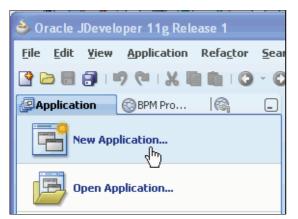
Creating the Process Model

1. Open JDeveloper Studio 11.1.1.3 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:
O Default Role
Enables all technologies.
O Customization Developer
Configures the product for customizing metadata.
O Database Edition
Includes only features for core database development.
O 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
OK Cancel

Close the Daily Tips window.

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



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

2010/22/01/01	Application Name:		
ion Name	HelloWorld_OBE		
1	Directory:		
Settings	C:\;Developer\mywark\HelloWorld_OBE		Browse
	Application Package Prefix:		
	Application Template:		
	Generic Application Creates an application which includes a single project. The pro preconfigured with JDeveloper technologies, but can be custo technologies.	 The second se	de any
	BPM Application Creates a BPM application. The application consists of one BPM has also SOA technology	Aproject.This	project
	Fusion Web Application (ADF) Creates a databound ADF web application. The application cor for the view and controller components (ADF Faces and ADF T another project for the data model (ADF Business Components)	ask Flows), ar	

Click Next.

3. In Step 2 of the Create BPM Application wizard, you create a project for the HelloWorld_OBE application. Enter HelloWorldProject as the Project Name. Notice that BPM and SOA are selected as project technologies by default. Click Finish.

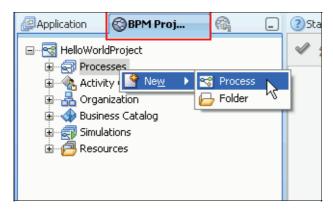
Name your project	BRUTENO BROTOL IL MUTTOR	5
<u>Application Name</u> Project Name	Project Name: HellcWorldProject Directory: C:\JDeveloper\mywork\HelloWorld_OBE\HellcWorldProje	ct Browse
Project SOA Settinos	Project Technologies Generated Components Associated Librar Available: Selected: ADF Business Components ADF Desktop Integration ADF Desktop Integration ADF Faces ADF Library Web Application Support Image: Component Selected: ADF Page Flow Image: Component Selected: ADF Swing Image: Component Selected: Ant Image: Component Selected: Database (Offline) Image: Component Selected: EJB Image: Component Selected: HTML Java Java Image: Component Selected: BPM Technoolgy Image: Component Selected:	ies

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 HelloWorld_OBE application appearing in the drop-down list just above the panel and the HelloWorldProject appearing as the parent node within the panel. The fact that it appears in *italics* indicates that there are unsaved changes.

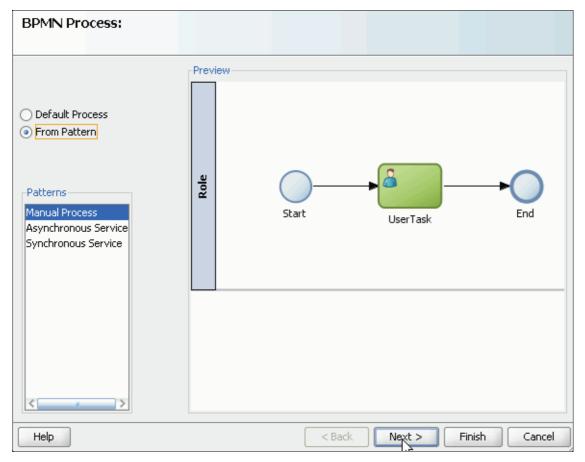
Application 🔗 BF	PM Pro 🦓	
🔁 HelloWorld_OBE		•
Projects	■ 🖓 🖓 • 35 •	
HelloWorldProject SOA Content SoA	ules	

Click the Save All icon on the main toolbar.

4. 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 **From Pattern** radio button, and then the **Manual Process** pattern. Click **Next**.



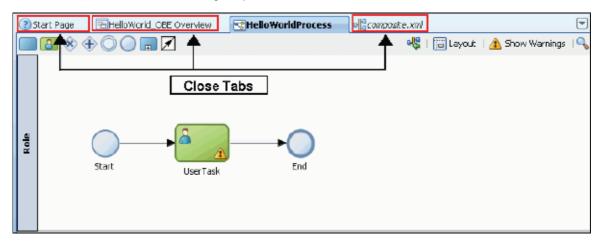
In the next screen, name the process "HelloWorldProcess" and click Finish.

BPMN Process:				
General Advanced Name HelloWorldProcess				
Id: HelloWorldProcess Description				
				٢
Others Author: jmoritz				
Help	< Back	Next >	Finish	Cancel

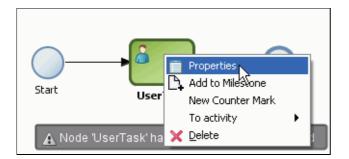
The process model appears in the design editor panel in the middle of the JDeveloper window. The tab name will be same as the name of your new process.

Click the Save All icon again.

You may wish to close the other tabs, as you will not be using these. An X will appear in the upper right corner of the tab when your cursor approaches it. The X will close the tabbed pane. All of these can be easily reopened later from either the menu or one of the navigator panels.



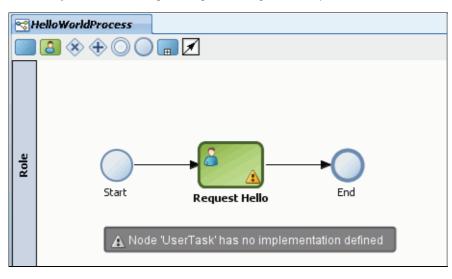
5. Change the name of the user task in the design model. Notice that the model begins and ends with two circular icons. The circle on the left is a Start activity and the circle on the right is an End activity. Connecting the two circles is a line that represents the flow of activities through the process. This is called the sequence flow and sometimes is referred to as the "transition line". Between the Start and End activities is a User Task type activity. Right click on this and select Properties.



When the **Properties** dialog box appears, on the **Basic** tab, change the name of the activity to "Request Hello". Click **OK**.

🌢 Properti	es - UserTask		
Basic Imp	lementation		
Name:	Request Hello	۲	
Description:		۲	

Don't worry about the warning message indicating that no implementation has been defined. You will do this later.



Click Save All.

6. Add the Component Palette to the JDeveloper window by selecting View > Component Palette from the menu. The palette will appear in the right pane of the window.

Select **BPM** from the drop-down list at the top of the Component Palette, then expand the **Activities** accordion panel as shown below.

Component Palette	🕞 R	_
BPM		-
8 0		٢
Activities		
– Default		
Activity		
🚌 Business Rule		
Event Subprocess		
付 Manual		
Receive		
🛐 Script		
醛 Send		
🛞 Service		
🔚 Subprocess		
- Interactive		
Complex		
🛃 FYI		
🚭 Group		
💫 Initiator		
陷 Management		
[User		

7. Add a service activity to the process. You'll need to first make room for another activity on the sequence flow.

Click on the **End** activity and drag it to the right, dropping it on the right side of the design panel, allowing enough room for another activity icon to fit between the **Request Hello** activity and the **End** activity.



Now click the **Service** activity icon in the Component Palette and drag it to the Sequence flow between **Request Hello** and **End**. Drop it there. Notice that the transition line turns blue when the drop target area approaches the line.

Important: The transition line *must be blue* when you drop the object in order for the transition line to be connected to the activity.

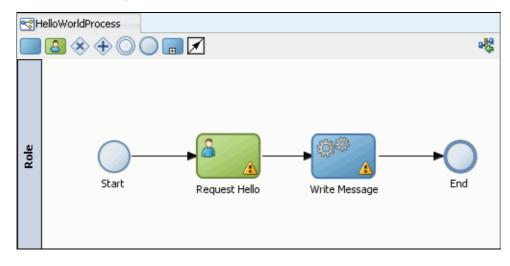


When you drop it, the **Properties** dialog box for the activity opens. On the **Basic** tab, change the activity name to "Write Message".

े Properti	ies - ServiceTask	X
Basic Imp	olementation	
Name:	Write Message	۲
Description:		۲
🗄 Sampling	Point	

Click OK.

8. Click Save All. Your process model should now look similar to this.



Creating the Business Object

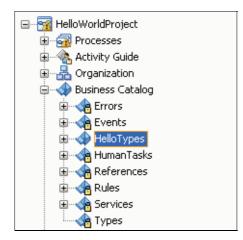
1. Now you will create a business object capable of storing multiple pieces of data, related to the message that the user enters in the Request Hello activity. This object will be populated when the user enters the message. It will then be passed to the Write Message activity so that the message can be written to a file.

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

Application SBPM	Proj	R		HelloWorldProcess
🖃 📲 HelloWorldProject				🔲 🖪 🛞 🕂 🤇
🗄 🕣 🔂 Processes				
🗄 🕀 Activity Guide				
🗄 📲 Organization				
🗄 📣 Business Catalo	ą.			
🗄 🕣 🕣 Simulations	Ne <u>w</u>	•	🚸 Moo	
🗄 🗃 Resources	🝓 <u>R</u> eload		🐻 Busi	iness OSject
	_		🌉 Busi	ness Exception

When prompted to name the new module, enter "HelloTypes" and click OK.

The HelloTypes module now appears beneath the Business Catalog node.



2. Right click the HelloTypes module and select New > Business Object.

🖨 📣 Business Catalog	
🗄 😪 🙀 Errors	
🖶 😪 Events 🔤 .	
🕀 📣 HelloTypes	
😥 🛶 HumanTas 🎴 New 🔹 🕨 Module	art
🕀 🙀 Reference 💥 Delete 🛛 🐼 Business	s Object 🔪 👘
🕀 🙀 Rules 🛛 💀 Reload 🛛 🥵 Business	s Exception 🕅
🗈 😪 Services	
Types	
🗄 🖙 🛃 Simulations	
🗄 🔁 Resources	

In the Create Business Object window, enter "HelloObject" as the Name and accept HelloTypes as the Destination Module. Click **OK**.

Create Business Object		X
Business Object		
Name:	HelloObject	
Destination Module:	HelloTypes	
Based on External Schema		Q,
Help	OKCanc	el

The HelloObject editor now opens in the editor.

3. Add the following three attributes to HelloObject:

Attribute Name	Туре
date	Time
greeting	String
message	String

The following instructions describe how to create the **date** attribute.

Click the plus sign next to the Attributes section of the Business Object editor as shown below:

	s 🔯 HelloObject	
Business Object		
Description:]
Documentation:	Edit	
🗄 Details		
Attributes		4

The **Create Attribute** popup appears. Enter "date" as the Name value. Select **Time** as the Type from the dropdown list. Click **OK**.

🔷 Create Attribute	
Name: date	
Type: Time	•
Help	OK Cancel

The date attribute now appears in the Attributes section of the Business Object editor.

Continue working in this way to create the other two attributes. When you finish, the Business Object editor should look like this:

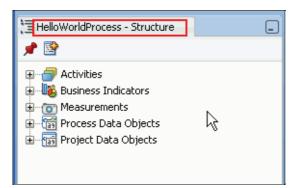
😪 Hello World Process 🛛 🗔	HelloObject		
Business Object			
Description:			
Documentation: Edit			
🛨 Details			
Attributes			4
Time date		 ×	û 🕹
∃ String greeting		×	û 🎝
		×	∲ IJ

Click Save All and close the HelloObject tab in the editor panel.

4. Now create a process *data object* of type HelloObject so that you can use it in your process.

If the **HelloWorldProcess** tab is still open in the editor panel, click anywhere in the design editor to put the focus on the HelloWorldProcess. (If it is not open, select it within the BPM Project navigator by expanding **HelloWorldProject** > **Processes** to find it)

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.

🗄 HelloWorldProcess - Structure
📌 🖹
🕀 🖅 🗇 Activities
🗄 📲 👪 Business Indicators
🗄 🐨 📷 Measurements
Process Data Object
🗄 📲 Project Data Objects 上 New

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

🍲 Create Data Object	
Name: hello	
Type: String	
🖌 Auto initialize	
Help	OK Cancel

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

💩 Browse Types	
Type: <component></component>	•
-Component	
HelloObject	
HelloTypes.HelloObject:	
Help	OK Cancel

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



Click Save All.

Implementing the User Task

1. Every interactive activity must be bound to a task service to provide its implementation. In the case of a User Task (such as Request Hello), it must be bound to a *Human Task* type task service. You will create the Human Task in this step.

Right click the Request Hello activity in the design editor and select Properties to open the Properties window.

Click the Implementation tab. Next to the Human Task field, click the plus sign button as shown below.

🔷 Properties - Request Hello			×
Basic Implementation			
Implementation Type: 📵 User task			-
Human Task:	÷	Q	<i>i</i>
Human Task Attributes			

In the Create Human Task dialog box, enter or select the following values for fields in the top portion of the dialog:

Field	Value		
Name	SayHello		
Pattern	Initiator		
Value	Please Enter a Hello Message		
Outcomes	Submit <this auto="" choose="" for="" is="" pattern="" selected="" the="" when="" you=""></this>		

Add a parameter by clicking the plus icon above and to the right of the Parameters panel.

춸 Create H	luman 1	ask		×
General				
Name:	SayHello			Priority: 3 (normal) 🔻
Pattern:	😰 Initi	ator		-
Title:	Please E	nter a Hello Message		
Outcomes:	SUBMIT			۹.
Parameters:				+ ×
Parameter		Name	Туре	Editable
Outcome targ	get:			
Help				OK Cancel

This opens the Data Object window displaying available data objects that you can drag into the Parameters panel. Click on the **hello** data object and drag it into the parameters panel. Select the **Editable** checkbox for the new parameter.

े Create H	luman Task			🕹 Data Object 🛛 🔀
General				Drag Data Objects to Parameters table and Outcome Target field.
Name:	SayHelo	F	Priority: 3 (normal) 🔻	
Pattern:	👜 Initiator		-	
Title:	Please Enter a Hello Mess	sage		E
Outcomes:	SUBMIT		Q	HelloWorldProject
Parameters:			× 🗣 🕺	
Parameter	Name	Түре	Ecitable 📕	
HelloObject	hella	HeloTypes.HeloObj	. 🗹	
Outcome targ	iet:			
Help			CK Cancel	
				Help dose

Close the Data Object window and then click OK in the Create Human Task window..

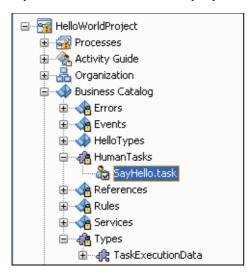
You are returned to the Properties window for the Request Hello activity. Click $\ensuremath{\text{OK}}$.

🕹 Properties - Request Hello	X
Basic Implementation	
Implementation Type: [] User task	•
Human Task: 😰 SayHello	- 🕂 🔍 🥜
Human Task Attributes	
Title: Plain Text 💌	
Priority: 3 (normal) 💌	
Re initiate	
Advanced	
-Data Associations	
Use Associations Type: Simple V	
Use Transformations	
Help	OK Cancel

Click Save All.

2. Now you must provide a form for the user to enter the Hello message, ensuring that the form is linked to the hello data object.

In the BPM Project Navigator, expand **Business Catalog > Human Tasks**. Here you see the **SayHello.task** object. This is the human task that you just defined.



Double click it to open it in the editor. When it opens, click the **Create Form** drop-down list in the editor toolbar on the left and select **Auto-Generate Task Form** ...

😪 HelloWorldProcess	🏠 SayHello.task	6
🧏 🞯 Create Form 🔻	52	
Auto-Generati	e Task Form	
🚆 🛛 Launch Task F	orm Wizard 🤟 🗌	
🤟 vata	Lask Lice	Plain Text 🗾 Please Enter a Hello Message
酇 Assignment	Description:	
🅮 Presentation		
🚳 Deadlines	Outcomes:	SUBMIT
Notification	Priority:	3 (Normal) 💌
🗞 Access	Category:	By expression 💌
🗲 Events	Owner	Application Role - HelloWorldProject.ProcessOwner Static - Q
	Application Context:	OracleBPMProcessRolesApp

The **Create Project** window opens. It is necessary to create a separate project to contain UI elements (ADF forms). Enter HelloWorld_UI as the Project Name and accept the default directory. Click **OK** when finished.

🍲 Create Project		×
Enter the name and dire projects to organize you	ctory of the new project. Use ır files.	
Project Name: HelloWorld_UI		
Directory:	- 	
Developer\mywork\Hello	oWorld_OBE\HelloWorld_UI	Browse
Help	ок	Cancel

It will take several seconds (even up to a minute, depending on resources) to create the form and open the necessary editor. Eventually, you will see the editor shown below (partial view). The highlighted section is the portion that will be visible to the end user. Notice the **Date**, **Greeting**, and **Message** fields.

gend info				
2 8 2771	ETAILS???			toolbar
ASSIGN		EXPIRATION_DATE???	. ,	21 TASK_NUMBER?1
???CREAT	OR??? #{creator. inputValue}	???ACQUIRED_BY???	#{acquiredBy. i∩put∀alue}	???PRIORITY??
CREATE_D/	??? #{createdDate. \TE??? inputValue}	???DUE_DATE???		???STATE??
UPDATE_D/	??? #{updatedDate. \TE??? inputValue}	???OUTCOME???	#{ actionDisplayName. input∀alue}	
2??CONTEN	TS???			
Date	#{date.inputValue}		•	
	#{greeting.input∀alue}			

Select the **Message** field as shown above. When you do so, the properties for this field appear in a panel in the lower right corner of the JDeveloper window. Expand the **Appearance** accordion panel within this panel and change the value of the **Rows** property to **5**.

🛱 Input Text - #{bindings.message.hints.la 🤅 🤅				
🔣 i 🏓 🔮	10	/ 🖪 🤇	ið Find 🕹 🕄	
🖃 Common				
o Id:	it2			~
Rendered:	<de< td=""><td>efault>(tru</td><td>Je) 🔻</td><td>~</td></de<>	efault>(tru	Je) 🔻	~
Label:	#{b	indings.me	ssage.hints.label}	~
📵 Value:	#{bindings.message.inputValue}			~
🖃 Appearanc	e			
Columns:		#{binding	s.message.hints.display	 ~
Rows:		5		\sim
Secret:	<default>(false)</default>		~	
Wrap:	<default></default>		~	
ShowRequir	ired: $\overline{\langle}$ default>(false) \checkmark		$ $ \sim	
Changed:		<pre>cdefault`</pre>	> (false) 🔷 👻	

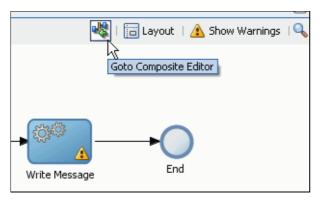
Tab out of the field so that the change will take effect in the form.

ITS???		
#{date.input∀alue}		
#{greeting.inputValue}		
#{message.inputValue}	^	
	#{date.input∀alue} #{greeting.input∀alue}	#{date.inputValue} #{greeting.inputValue}

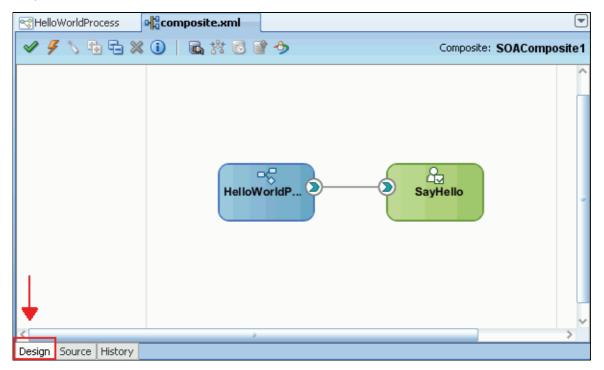
Click Save All. Close all tabs in the editor except for the HelloWorldProcess tab.

Implementing the File Service

1. Create the implementation for the **Write Message** service activity using the **SOA Composite** editor. Click the **Goto Composite Editor** button on the Design Editor toolbar as shown below.



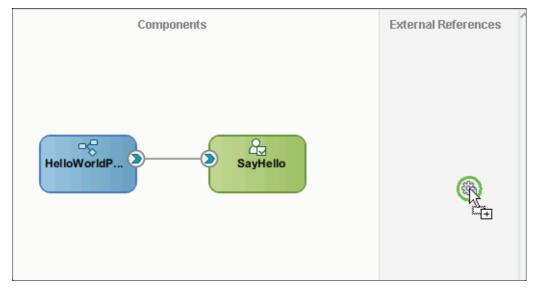
The SOA Composite editor opens. To view the design editor, rather than the XML source, click the **Design** tab at the bottom left margin of the Composite Editor panel. The HelloWorldProcess BPMN component and the SayHello human task component are shown in the composite editor. These are considered components of the SOA composite.



2. Click the File Adapter service adapter from the Component Palette. Notice that the Component Palette is now showing SOA components, by default.

Component Palette	Re	_
SOA		-
6 0		\odot
- Service Components		<u> </u>
Å BPEL Process		
😪 BPMN Process		
🧼 Business Rule		
🇞 Human Task		
🐗 Mediator		
🍉 Spring Context		1
- Service Adapters		
造 ADF-BC Service		
🏭 AQ Adapter		
🔯 B2B		
🛍 BAM Adapter		
🍘 Database Adapter		
🍪 Direct Binding		
🔞 EJB Service		
豄 File Adapter		
🚳 FTP Adapter		×

Drag and drop the File Adapter into the External Reference column of the editor.



The Adapter Configuration Wizard opens when you drop it.

3. Click Next on the Welcome page of the Adapter Configuration Wizard.

On the Service Name page of the wizard, name the service MessageWriter. Click Next.

Service Name	
Enter a Service Name.	
Service Type: File Adapter	
Service Name: MessageWriter	

On the Adapter Interface page, select Define from operation and schema (specified later) . Click Next.

Adapter Interface	01010101010101010449098546	X
	fined by a wsdl that is generated using the operation name and schema(s) specified later i adapter interface may be defined by importing an existing WSDL.	n
	operation and schema (specified later)	
 Import an ex 	xisting WSDL	
WSDL URL:		2
Port Type:	▼	
Operation:	~	

On the Operation page, select **Write File**. The Operation Name will be pre-populated with the name **Write**. Accept this value and click **Next**.

Operation	010101010101010104049494949494
system, a Write File operation that contents of a file, and a List Files op	rations. There is a Read File operation that polls for incoming files in your local file creates outgoing files, a Synchronous Read File operation that reads the current peration that lists file names in specified locations. Specify the Operation type and on per Adapter Service may be defined using this wizard.
Operation Type: <u>R</u> ead File <u>Write File</u> <u>Synchronous</u> <u>List Files</u>	Read File
Operation Name: Write	

On the File Configuration page, select **Physical Path** as the **Directory specified**. Enter a dot ('.') for the **Directory for Outgoing Files (physical path)** field. Also enter a **File Naming Convention** of:

Hello_%SEQ%.xml

File Configuration			*
Specify the parameters for the Writ	e File operation.		
Directory specified as Directory for Outgoing Files (physic	vsical Path Logical Name al path):		
Ŀ			Browse
File Naming Convention (po_%SEQ	%.txt); Hello_%SEQ%.xm	1	
Append to existing file			
Write to output file when any of t	hese conditions are met		
✓ Number of Messages Equals:	1	<u> </u>	
Elapsed Time Exceeds:	1	minutes	
File Size Exceeds:	1000	kilobytes	*

Click Next.

In the Messages page, you determine *what* should be written to the file. Click the magnifying glass icon next to the **URL** field to open the Type Chooser popup. Expand **Project Schema Files** and **HelloObject.xsd** to find and select **HelloObject**. as shown below. Click **OK** to accept the selection and return to the Messages window of the wizard.

Messages	
defines the messa	e for the Write File operation. Specify the Schema File Location and select the Schema Element that ges in the outgoing files. Use the Browse button to find an existing schema definition. If you check by then you do not need to specify a Schema.
Message Schema	
Native format	translation is not required (Schema is Opaque)
URL	
Schema Element	
	🕹 Type Chooser 🛛 🔣 📝 🛁
	😤 🗃 🚩
	C Type Explorer
	😑 🛃 HelloObject vsd
	HelloObject
	😥 🚠 SayHelloPayload.xsd
	AsyHelloWorkflowTask.xsd AsyNetReference
	Project WSDL Files
	Type: {http://xmlns.oracle.com/bpm/bpmobject/HelloTypes/HelloObject}Hell
Help	Show Detailed Node Information
	Help OK Cancel

Click Next in the Messages window.

Messages				01010101010101	010191919191919191		Es
defines the message	for the Write File oper es in the outgoing files. then you do not need	Use the Brows	se button to				
Message Schema-							
Native format t	ranslation is not require	ed (Schema is (Opaque)				
URL	businessCatalog/Hello	Types/HelloOb	ject.xsd				Q 🔅
<u>S</u> chema Element	HelloObject					-	

Click Finish in the final page of the wizard. The service is created and appears in the SOA Composite editor.

Click Save All and close the Composite editor tab.

4. Now you must wire the service implementation you just created to the **Write Message** activity in the BPM process. You need to be in the BPM Process design editor for this.

Note: If necessary, open the **HelloWorldProcess** in the design editor by double clicking on it in the **BPM Project Navigator**.

Right click on the Write Message activity and select Properties.

\$	
	Properties
Write Me	To activity 🗟 🔹 🕨
	New Counter Mark
	🗙 <u>D</u> elete

In the Properties window, click the Implementation tab.

Select **Service Call** as the **Implementation**. Click the magnifying glass icon next to the **Name** field to browse for a service. The **Type** popup window appears, displaying your **MessageWriter** service. Select it and click **OK**.

🕹 Properties - Write Message	×
Basic Implementation	
Implementation Type: 🔯 Service task 🔹	
Service task	
Implementation: Service Call	
Name:	
Operation:	
Data Associ 🍲 Type	
Use As Search:	
Use Tr:	
/Services/Externals/MessageWriter:	
Help Cancel	

Back in the Properties window, notice that the Operation field has now been automatically set to write.

In the **Data Associations** panel, click the **Use Associations** checkbox, then click the pencil icon next to it. This opens an editor that allows you map data into the activity. In this case, you want to map the *hello* object into the activity so that the MessageWriter service can write the message.

🔶 Properties - Wr	ite Message	×
Basic Implementat	ion	
Implementation Type:	Service task	-
Implementation:	🕃 Service Call	-
Name: Messag	eWriter	م ي
Operation: write		-
Data Associations		

In the **Data Associations** window that opens next, drag the hello data object from the right column over to the **Inputs** field on the left side of the window. Notice that this maps the hello data object to the helloObject expected as input to the activity's service implementation.

🔶 Data Associations	$\overline{\mathbf{X}}$
24 🖹	🖓 👯 📃 I 📼 📼
Drag variables from the right panel into fields on the left.	HeloWorldProcess
Help	OK Cancel

Click OK in the Data Associations window to save the mapping and return to the Properties window.

Click **OK** in the Properties window.

Click Save All.

Enhancing the Basic Hello World Process

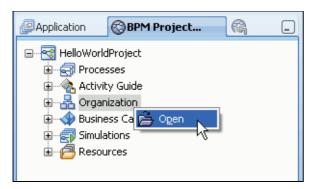
In this section, you enhance the basic process you just built by adding a review capability for the message entered by the user. You add another business object to hold review-related data and use this business object in conjunction with a business rule that tests the length of the greeting and message. You also add another human interaction that allows the user with the Reviewer role to review the entered message and accept or reject the message. Then you change the process model, itself, so that it conditionally branches to the Review Message activity and potentially back to the Request Hello activity if the message was rejected. A script task will also be needed in order to initialize a variable used in the conditional branching logic.

You begin the enhancement by changing the name of the role assigned to the user who executes the Request Hello activity to something more meaningful. Then you add a new role for the user who reviews the message.

Adding the ReviewNeeded Business Object Adding a Business Rule Defining a Decision Table for the Business Rule Adding the Review Message Human Interaction Adding Conditional Branching Adding a Script Task

Adding the ReviewNeeded Business Object

1. Change the name of the **Role** role. In the BPM Navigator, right click on the **Organization** node beneath **HelloWorldProject** and select **Open**.



The Organization editor opens in the center panel. Select the Role role and click the pencil icon to edit it.

品	Organization	
	Roles	^
:	Name	4
	Process Owner	
	Role	
		Edit

The Role popup window appears. Enter the name **Requester** and click OK.

Role		
Name:	Requester	
Help		OK Cancel

2. With the Organization editor still open, click the green plus sign button to add a new role.

-Roles	
Name	
Process Owner	
Requester	bbA
	~

Enter the name Reviewer in the Name field of the Role popup window and click OK.

Your Organization editor should now look like this:

-Roles		
Name		4
Process Owner	-	
Requester	-	×
Reviewer	V	~

3. Click Save All and close the Organization editor.

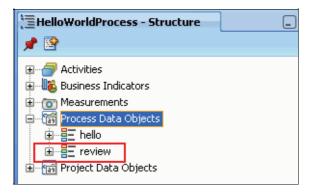
4. Add a new business object in the same manner that you did in the **Creating the Business Object** section above. Follow Steps 2 and 3 of that section, placing the new business object in the **HelloTypes** module and substituting the name and attributes mentioned below.

Name the object **ReviewObject** and add the following attributes to it:

Attribute Name	Туре		
review	String		
reason	String		
HelloWorldProcess	👸 Review O	bject	
Description:			
Documentation: Ed	it		
🗄 Details			
Attributes			
		×	1
		×	1

Click **Save All** and close the object editor. The ReviewObject should now appear in the BPM Navigator beneath the HelloTypes folder.

 Declare a process data object of type ReviewObject in the same manner that you did in the Creating the Business Object section above. Follow the instructions in Step 4, substituting review as the data object name.



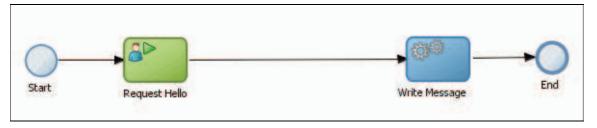
6. Add one more process data object of type String, called reviewOutcome. Follow the same procedure as the previous step, except choose String as the type.

🕹 Cre	eate Data Object	
Name:	reviewOutcome	
Type:	String	▼
💌 Aut	o initialize	
Help	.	OK Cancel

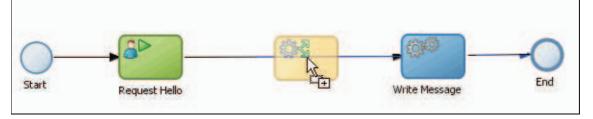
Adding a Business Rule

1. Add a business rule to the design model.

Begin by making more room on the sequence flow between the Request Hello and Write Message activities. Move the End activity and the Write Message activity further to the right.



Expand the Activities accordion panel in the Component palette, then click and drag a Business Rule into the design editor, dropping it onto the sequence flow between Request Hello and Write Message.



The Properties dialog for the business rule appears when you drop the object. Enter **ReviewNeeded** in the **Name** field. Click **OK**.

춸 Properti	es - BusinessRuleTask	×
Basic Imp	lementation	
Name:	ReviewNeeded	۲
		۲
Description:		
. ∃ Sampling	Point	
<	ð	>
Help	OK Cano	el

Click Save All.

2. In the next step, you will define the implementation of the ReviewNeeded business rule using the Composite editor. In order to make the two business objects available to the Composite editor, you must copy the xsd files that were generated for each object when you created them from the businessCatalog\HelloTypes folder to the xsd folder within the directory structure of your project. [This is a workaround to a known bug in this release of the product]

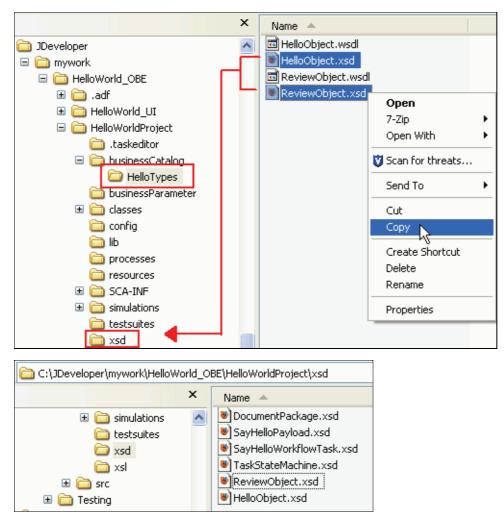
Open Windows File Explorer and navigate to:

```
C:\JDeveloper\mywork\HelloWorld_OBE\HelloWorldProject\businessCatalog\HelloTypes
```

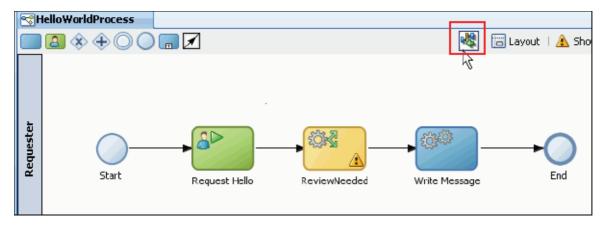
```
In this folder, find and copy the following two xsd files:
    HelloObject.xsd
    ReviewObject.xsd
```

Paste them into the following folder:

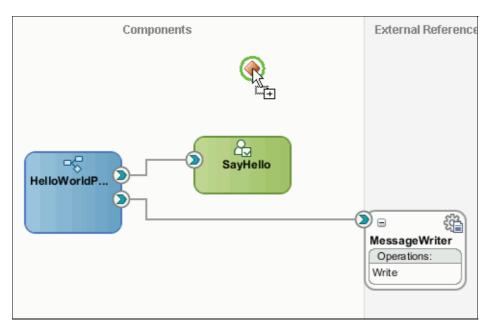
C:\JDeveloper\mywork\HelloWorld_OBE\HelloWorldProject\xsd



3. Open the Composite editor by clicking the Goto Composite Editor button on the Design editor toolbar.



4. Add a **Business Rule** component to the Composite Editor. From the Component palette panel on the right, click and drag a Business Rule from the Service Components section of the palette and drop it in the Components column of the Composite Editor.



When you drop the component, the Properties dialog appears. Enter HelloRules in the Name field. Then click the green plus button above the Inputs/Outputs section and select Input...

🍲 Create I	Business Rules			×	
	ule ss rule defines or const isiness structure or inf		•	is intended to	
General	Advanced	Dictionary			
Specify th	e name and package f		at will be created.		
<u>N</u> ame: <u>P</u> ackage:	HelloRules				
Project:	roject: C:\JDeveloper\mywork\HelloWorld_OBE\HelloWorldProject\HelloWorldProject.jpr				
<u>I</u> nputs/Ou	itputs:			$+ \cdot \times \land \vee$	
Direction	l	Name	Туре	🔿 Input	
				🗢 Output.!N	
Help			(OK Cancel	

The **Type Chooser** popup window appears. Expand the second **HelloObject.xsd** entry, then select the **HelloObject** beneath it and click **OK**.

🕹 Type Chooser	×
	*
Vype Explorer Project Schema Files HelloObject.xsd DocumentPackage.xsd HelloObject.xsd HelloObject.ype ReviewObject.xsd SayHelloPayload.xsd SayHelloWorkflowTask.xsd TaskStateMachine.xsd	
Type: {http://xmlns.oracle.com/bpm/bpmobject/HelloTypes/HelloObject}HelloObject	
Show Detailed Node Information	
Нер ОК Са	ncel

Click the green plus button once more and this time select **Output...** When the Type Choose popup window appears, expand the second **ReviewObject.xsd** entry and select **ReviewObject**. Click **OK**.

💩 Type Chooser	X
	40 40
▼ype Explorer Project Schema Files HelloObject.xsd Project Schema Files HelloObject.xsd Project Schema Files Project Schema Files	5
Type: {http://xmlns.oracle.com/bpm/bpmob	ect/HelloTypes/ReviewObject}ReviewObject
Show Detailed Node Information	
Help	OK Cancel

Back in the Business Rule properties window, click $\ensuremath{\textbf{OK}}$.

Business R	ule	
		one aspect of your business that is intended to the behavior of your business.
General	Advanced	
⊙ <u>C</u> reate	e Dictionary 🔘 <u>I</u> mport Dictio	nary
Specify th	e name and package for the	dictionary that will be created.
<u>N</u> ame:	HelloRules	
<u>P</u> ackage:	helloworldproject	
Project:	C:\JDeveloper\mywork\Hell	oWorld_OBE\HelloWorldProject\HelloWorldProject.jpr
Inputs/Ou	itputs:	🕂 - 💥 🐟 🛇
Direction	Name	Туре
Input	HelloObject	{http://xmlns.oracle.com/bpm/bpmobject/HelloTyp
Output	ReviewObject	{http://xmlns.oracle.com/bpm/bpmobject/HelloTyp
	se as Composite Service	
Help		OK Cancel

Click Save All.

5. You now need to wire the implementation that you just created (the HelloRules business rule) to the ReviewNeeded business rule activity in the BPM process. Click the HelloWorldProcess tab to bring the focus back to the BPM Design Editor.

Right click the **ReviewNeeded** activity to open its properties window. Click the browse button to the right of the **Business Rule** field.

🔶 Properties - ReviewNeeded	X
Basic Implementation	
Implementation Type: 🧱 Business Rule task	-
Business Rule:	<i>i</i>
Decision Function:	-
Data Associations	
Use Associations Type: Simple V	
Use Transformations	

When the Browse Business Rules window opens, you should now see your **HelloRules** business rule implementation. Select it and click **OK**.

🚔 Browse Business Rules	
Search:	
Search:	
	incel

6. In the Properties window, click the checkbox next to Use Associations. Keep the default setting for Type, which is Simple associations and click the Edit Data Associations button (the pencil icon).

🔶 Properties - ReviewNeeded				
Basic Implementatio	n			
Implementation Type:	Business Rule task	-		
Business Rule:	HelloRules 🗣 🔍	@		
Decision Function:	HelloRules_DecisionService_1	-		
Data Associations —				
Use Associations	s Type: Simple 🕶 🧷			
Use Transformal	tions Edit Data Associations			

The Data Associations window opens. In this window, you map process data objects (hello and review) to the Inputs and Outputs that you defined for the **HelloRules** business rule implementation. From the right column, click and drag the review data object, dropping it into the **Outputs** field as shown below.

a 24			Al 📲 🔚 📼 🚥
Drag variables from the right panel into field	s on the left.		HelowarldProcess
]nputs	ReviewNeeded Inputs Imputs Curputs ReviewObjectType ReviewObjectType	Cutputs	

Now click and drag the hello data object from the right column, dropping it on the Inputs field. Click **OK** to save this mapping and close the Data Associations window.

🖕 Data Associations		
× 🗈		
Drag variables from the right panel into field	s on the left.	
Inputs hello	ReviewNeeded	Outputs
		\searrow

Click OK in the Properties window. Click Save All.

Defining a Decision Table for the Business Rule

In this section you define a set of rules that will be applied when the user enters a hello message and greeting. The following table illustrates what those rules are and how they will be applied.

IF	Length is			
greeting:	Short	Medium	Medium	Long
message:	(anything)	Short	Medium or Long	(anything)
ReviewObject:		→	→	Ļ
reason:	"Greeting is too short"	"Length is too short"	null	null
review:	"true"	"true"	"false"	"false"

The lengths of the HelloObject.greeting and HelloObject.message strings will be evaluated. You will define four rules that determine the combination of greeting and message lengths and then set the appropriate values to the ReviewObject properties: reason and review. For example, if the greeting length is Medium and message length is Short, the ReviewObject's reason property would be set to "Length is too short" and its review property would be set to "true". Consequently, the process would flow to a ReviewNeeded activity and could be accepted or rejected by the reviewer.

A decision table can be designed in many different ways. In our scenario, the decision table consists of four elements as described here:

A set of **Conditions** - In this case, each condition is the length of either the *greeting* or the *message* string variable

A **Range of Values**, called a **bucketset** - For example a variable might be < 5 characters in length, or **between 5 and** 50 characters in length, and so on. These are ranges within a bucketset.

An Action - This is the act that should occur when a rule is evaluated. For example, the *review* property might be set to "true".

A **Rule** - This is a specific mapping of **condition** > **range** > **action**.

The Decision table is created in five high level steps:

- 1. Define the conditions (Step 2 below)
- 2. Define a bucketset (Step 3)
- 3. Assign that bucketset to each condition (Step 4)
- 4. Define the action (Step 5)
- 5. Define the rules (Steps 6 9)
- 1. Open the Rules Editor .

In the BPM Navigator, expand **Business Catalog > Rules > Helloworldproject**. Double click on **HelloRules.rules**.

Application	BPM Project	- CA		
🖃 😽 HelloWorld	Project			
🗄 🕣 🔂 Proces	sses			
😐 🕀 Activit	:y Guide			
🗄 📲 Organ	ization			
🖨 📣 Busine	ess Catalog			
🗄 🛶 🖓 Er	rors			
🗄 😪 Ev	🖶 😪 Events			
🗄 ··· < 🖓 H	elloTypes			
🗄 🖓 🕀	umanTasks			
	eferences			
	ules			
	a Helloworldproject			
😟 🕀 🔄 🗄 🗄 🗄	itions			

The Rules Editor opens and, by default, the Ruleset_1 node in the left panel of the editor is selected.

2. Create a decision table for the rule.

Click the green plus sign button on the toolbar and select Create Decision Table from the drop-down menu.



The Ruleset_1 editor changes to provide a table for you to define the decision table. Change the default name of the decision table from **DecisionTable_1** to **ReviewDecisionTable**. Click on the field containing the default name. A text field appears below it. Enter the new name and hit Enter to accept the value.

E Rule	eset_1 View: 🔯 DecisionTable_1	• • * %
* 3	DecisionTable_1 <enter description=""></enter>	🔜 - 60 🕱 🖬 - 🔞
	Press the Enter key to save changes	

Add two conditions to be checked for the table:

HelloObject.greeting.length() HelloObject.message.length()

Add the first condition by clicking on the **<insert_condition>** field in the table as shown below. This will automatically add a condition row, with a link that says **<edit_ condition>**.

	- ♣ • ≫ । ⊗ ∞ । 苗 - 苗 - 監 । 🚱 🖽 🔜 🖽 •	62
<insert condition=""></insert>		

Right click on the **<edit_condition>** field and select **Edit Condition**. A drop-down list appears, displaying the defined data object types within the project. Expand **HelloObjectType** and select **greeting**.

• <u>C</u> onditions	
C1 <edit condition=""></edit>	
	F _x
Value Options ReviewObjectType HelloObjectType date greeting messag	<

Right click on HelloObjectType.greeting and select Edit Condition again.

Ruleset 1 View: ReviewDecisionTable
ReviewDecisionTable <enter description=""></enter>
C1 R1: 👻 🗐 Local List of Values 🔍 🥒 🕂 🐇 🛛
 <u>C</u>onditions
C1 HelloObjectType.greeties
<u>Delete</u>
🞥 Insert <u>B</u> efore
👒 Insert <u>A</u> fter
Conflict Resolutio 🌮 Edit Condition Bucketset
I Merge Condition

This time, the drop-down list that appears presents more options. Expand **HelloObjectType > greeting** and select the **length()** function.

• <u>C</u> onditions	
C1 HelloObjectType.greeting	
HelloObjectType.greeting	f _x
🔍 Value Options	~
🛱 🧧 HelloObjectType	
i∎…ª date	
🖨 🚥 greeting	
ength()	
toLower base()	
toUpperCase()	
trim()	
🗄 🗝 message	

Add the second condition by clicking the green plus sign button directly above the Conditions table and selecting **Condition** from the drop-down menu.



Right click on the newly added condition as you did for the first condition and select **Edit Condition**. This time, you can go directly to adding the length() function as part of the condition. Expand **HelloObjectType > message** and select **length()**.

• <u>C</u> onditions		
C1 HelloObjectType.greeting.len	ath()	
C2 <edit condition=""></edit>		
Value Options	R.	
→ HelloObjectType	- Ĥ	
i date		
💼 🔤 greeting		
🚊 🚥 message		
e length()		
toL(hprCase()		
toUpperCase()		
trim()		

Click Save All.

3. The rule will compare each condition against a range of values, called a "bucketset". Click the **Bucketset** node in the left panel of the Rules editor to create the bucketset.



Click the green plus sign button and select List of Ranges from the drop-down menu.

ଟ୍ରୋ	+·/×
Description	List of <u>V</u> alues
	🖬 List of <u>R</u> anges 🔪

A new bucketset row is inserted. With this row selected, click the pencil icon to edit the list of ranges.

Ø	Bucketsets			
B <u>u</u> ck	etsets:			🔞 i 🕂 - 🏹 🗙
	Name	Datatype	Form	Description
13	Bucketset_1	int	Range	

In the Edit Bucketset window that appears, change the value of the **Name** field to **Lengths**. Leave the Data Type field set to int.

The Range Bucket Values table has an initial default row in which the **Endpoint** is set to **-Infinity**. Click the green plus sign button twice more to add two more rows.

త	Edit	Bucketset - Buc	cketset_1				
	<u>N</u> ame: <u>D</u> ata T	Type: int					
	Range	Include D Bucket <u>V</u> alues:	isallowed Buckets ir	n Tests			+ ×
E		Endpoint	Included Endpoin!	Allowed in Actions	Range	Alias	Description
		50	Image: A start and a start	✓	>=50	>=50	
		0	~	~	[050)	[050)	
		-Infinity	✓	✓	<0	<0	

Change the 0 (zero) **Endpoint** value to 5 by clicking on the value and typing the new value. Notice that this changes the Range for this row to [5..50] and also changes the Range of the row whose endpoint is **'Infinity** to **<5**.

Change the Alias for each row to Short, Medium, and Long as shown below. Double clicking on the default Alias value makes it editable.

Range	Bucket <u>V</u> alues:					+ 🗙
	Endpoint	Included Endpoint	Allowed in Action	Range	Alias	Description
	50	Image: A start of the start	Image: A start and a start	>=50	Long	
	5	 Image: A start of the start of	 Image: A start of the start of	[550)	Medium	
	-Infinity	 Image: A start of the start of	~	<5	Short	

Click **OK** to close the Edit Bucketset window.

Click Save.

4. Assign the Lengths range of values bucketset to each of the conditions in your conditions table. Click the **Ruleset_1** node in the Ruleset editor to bring back the decision table editor.

Click the C1 condition to select it and then click the Local List of Ranges drop-down list above the Conditions table and select Lengths.

I Facts	Ruleset 1 View: ReviewDecisionTable
∫ _∞ Functions (x) Globals	ReviewDecisionTable <enter description=""></enter>
🤣 Bucketsets	C1 R1: ?
Links	C1 HelloObjectTyp Local List of Ranges
👔 Decision Functions Rulesets 🕂 💥	C2 HelloObjectTyp
& Ruleset_1	+ Create List of Ranges
	Conflict Resolution

Do the same thing for the C2 condition.

* 0	ReviewDecisionTable <enter description=""></enter>
C2 R	1: ? 💌 📰 Lengths 🔍 🥒
-	<u>C</u> onditions
C1	HelloObjectType.areeting.length()
C2	HelloObjectType.message.length()

5. You are almost ready to start defining the rules to be applied to each combination of range of values and condition. First you must define an Action to be performed when a rule is implemented.

Create a new Action. In the Actions panel, click the <insert_action> field and select Assert New.

• Actions			
<insert ac<="" th=""><th>tion></th></insert>	tion>		
	Assert New		
	Modify りく		
	Retract		
	Call		

Once the assert new () line appears, right click it and select Edit.

The Action Editor window opens. Select **ReviewObjectType** from the Facts panel. The properties of this object type now appear in the Properties panel in the lower portion of the window. Check the **Parmeterized** and **Constant** checkboxes for both the **reason** and the **review** properties.

Form: Assert f		T	ieuro)	
L	New ReviewObject	tType (reason:?, rev	lew:r)	
<u>F</u> acts: f _{ac} ReviewObj	iectType			
$f_{\mathbf{x}}$ HelloObjec				
Properties:				
Properties: Property	Туре	Value	Parameterized	Constant
	Type String	Value	Parameterized	Constant
Property		Value		
Property reason	String	Value	 Image: A start of the start of	 Image: A start of the start of
Property reason	String	Value	 Image: A start of the start of	 Image: A start of the start of
Property reason	String	Value	 Image: A start of the start of	Image: A state of the state

Notice that the effect of this is that an assert new() statement is being constructed in the Value field. It instantiates a ReviewObject with values for the reason property (a String), and the review property (a String) You can see this assert statement in the screenshot above. Each rule that you define in the next step will provide specific values for these parameters.

Click OK, then Save.

6. Define the first rule. Click on the C1 row beneath the R1 column. A drop-down list appears representing the bucketset you defined for this ruleset. Check **Short** and click **OK**.

-	<u>C</u> onditions	R1
С1	HelloObjectType.greeting.length()	?
C2	HelloObjectType.message.length()	Short Short Medium Long
×	Conflict <u>R</u> esolution	OK Cancel

Click the **R1** column for the **C2** row. This is the length for the **message** property. In this case, it doesn't matter what the length is, so enter a dash (-) and click **OK**.

-	<u>C</u> onditions	R1
C1	HelloObjectType.greeting.length()	Short
C2	HelloObjectType.message.length()	ζ
×	Conflict <u>R</u> esolution	Short Medium Long OK Cancel

In the **Actions** panel, click the checkbox in the column to the right of the **assert new** statement, indicating that you *do* want the ReviewObject asserted for this rule. Then click the row beneath the checkbox, next to the **reason** property. When the text field appears, type: "Greeting is too short". Hit Enter when finished.

-	Acti <u>o</u> ns		
Al	assert.new.ReviewObiectType(✓	
	reason:String		
	review:String)	"Greeting is too short"	
		No SDK Option Matches Found!	

Click the column next to the review property and enter the value "true" into the text field. Hit Enter when finished.

-	<u>C</u> onditions	R1
C1	HelloObjectType.greeting.length()	Short
C2	HelloObjectType.message.length()	-
-		
×	Conflict <u>R</u> esolution	
-	Acti <u>o</u> ns	
Al	assert new ReviewObjectType(✓
	reason:String	"Greeting is too short"
	review:String)	"true"

You have just defined **Rule 1**, indicating that if the *greeting* is "Short" (<5 characters), the message is subject to review (*review* = "true") and the *ReviewObject.reason* should be set to "Greeting is too short".

7. Define the second rule. Click the green plus sign button above the Rules column and select Rule.

+ • ≈ ×	笛 - 餔 - 監 60 🏗 👪 🖷 - 🔞
<u>R</u> ule	R1
<u>C</u> ondition	Short
Action 🕨	-

Notice that when the new rule column is added, it is added with the heading **R1** and the rule you just defined is moved over to the right, under the heading **R2**.

* 9	ReviewDecisionTable <enter description=""></enter>				
A1 R	1:	╋╸ ※ ◇ ◇ 苗・	ti - 🚉 i 🏍 🕱 🔜 🖽 - 🔞		
-	<u>C</u> onditions	R1	R2		
C1	HelloObjectType.greeting.length()	Z	Short		
C2	HelloObjectType.message.length()	2	-		
×	Conflict <u>R</u> esolution				
•	Acti <u>o</u> ns				
Al	assert new ReviewObjectType(✓		
	reason:String		"Greeting is too short"		
	review: String)		"true"		

Beware: When you make the first edit to the new rule, it will switch back to its original location and original heading name. Always be sure that you are editing the rule you intended to edit.

Make edits to this rule, just as you did the previous rule, using the following table to guide you:

Section of Decision Table	Property	Value to Set	
Condition HelloObject.greeting.length()		Medium	
Condition HelloObject.message.length()		Short	
Action		"Length is too short"	
Action	ReviewObject.review	"true"	

-	<u>C</u> onditions	R1	R2
C1	HelloObjectType.greeting.length()	Short	Medium
C2	HelloObjectType.message.length()	-	Short
×	Conflict <u>R</u> esolution		
-	Acti <u>o</u> ns		
A1	assert new ReviewObjectType(✓	✓
	reason:String	"Greeting is too short"	"Length is too short"
	review:String)	"true"	"true"

8. Define the last rule that appears in the table as shown at the beginning of this section. Define it in the same way you defined the rule in the preceding step. The table below provides the values you will use.

Section of Decision Table Property Value to Set	Table Proper	y Value to Set	
---	--------------	----------------	--

Condition HelloObject.greeting.length()		Long
Condition HelloObject.message.length()		-
Action ReviewObject.reason		null
Action	ReviewObject.review	"false"

Note that you must select the null from the drop-down list, rather than type it into the text field.

•	<u>C</u> onditions	R1	R2	R3
C1	HelloObjectType.greeting.length()	Short	Medium	Long
C2	HelloObjectType.message.length()	-	Short	-
×	Conflict <u>R</u> esolution			
•	Acti <u>o</u> ns			
Al	assert new ReviewObjectType(✓	✓	✓
	reason:String	"Greeting is too short"	"Length is too short"	null
	review:String)	"true"	"true"	"false"

9. Run the **Gap Analysis** tool to automatically create the final rule. It will determine which conditions have not been covered by the existing rules. Click the Gap Analysis button on the Decision Table editor toolbar.

	□ ◎ ◇ □ 苗 ▪ 插 ▪ ■	🗛 । 🚱 🕱 🔛 🖬 - 🚷
R1	R2	K R3
Short	Medium	Gap Analysis
-	Short	-

The Gap Analysis window opens showing the rule that it determined was missing. Click the checkbox above the rule to allow the rule to be included in the decision table. Click **OK**.

🍝 Gap /	Analysis	\mathbf{X}				
	There is 1 missing rule in the decision table.					
	You can add the missing ru the table header column.	You can add the missing rule to the decision table by selecting the checkbox in the table header column.				
	Conditions 🗹					
HelloOb	jectType.greeting.length()	Medium				
HelloOb	jectType.message.length()	Long, Medium				
💽 <u>F</u> it Co	olumns To Width					
Help		OK Cancel				

The rule now appears in the decision table as Rule 3 and the rule that you defined in the last step moves to the

Rule 4 position. Assert the action for Rule 3 (Column R3) and add the following action values:

```
reason = null
review = "false"
```

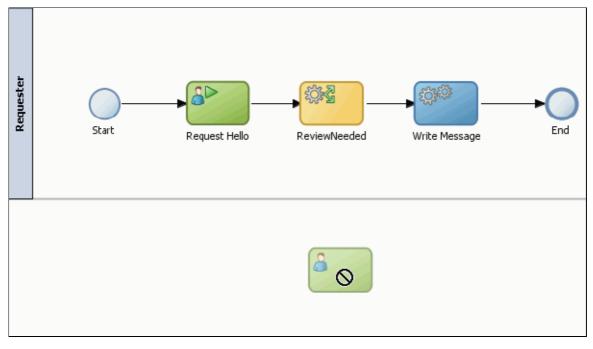
R1	R2	R3	R4
Short	Med	Long	
-	Short	Medium,Long	-
~	 Image: A start of the start of	 Image: A start of the start of	 Image: A start of the start of
"Greeting is too short"		null	null
"true"	"true"	"false"	"false"

Click Save and close the HelloRules.Rules tab.

Adding the Review Message Human Interaction

1. Add a new human interaction activity to the HelloWorldProcess. If necessary, click the HelloWorldProcess tab in the Design Editor to bring it to the front.

Expand the Activities accordion panel in the Component Palette and, from the Interactive section, click and drag a User activity, dropping it *outside* of the Requester lane and directly below the ReviewNeeded activity.



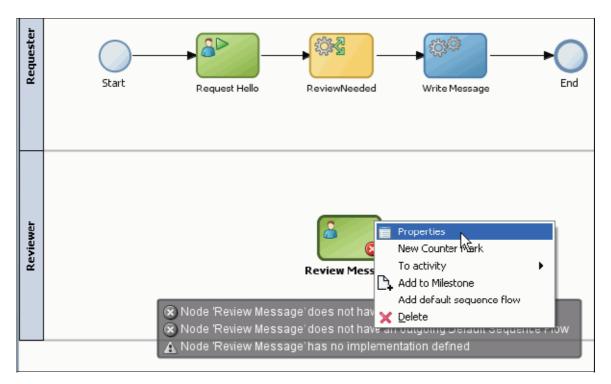
When the Properties window opens, name the activity Review Message and click OK.

📥 Propert	ies - UserTask				×
Basic Im	plementation				
Name:	Review Message				۲
					۲
Description:					
🗄 Sampling	Point				
<		8	à		>
Help				ОК	Cancel

You will then be prompted to assign a role to this activity. Select the **Reviewer** role from the drop-down list. Click **OK**.

े Role properties	
Name	
Reviewer	▼ New
Use variable value as parameter content	
review(ytcome	-
**	
Help	OK Cancel

2. Define the implementation for the Review Message activity. In the Design Editor, right click **Review Message** and select **Properties**.



In the Properties window, click the green plus sign button next to the **Human Task** field to define a new human task implementation.

🐟 Properties - Review Message	×
Basic Implementation	
Implementation Type: 👩 User task	•
Human Task:] 🕂 🔍 🥒
Human Task Attributes	Add

In the Create Human Task window, define the following properties. Accept the other default values.

Name: Review	vMes	sage
Title: Review	the	Message

🍲 Create H	luman Task 🛛 🔀						
General							
Name:	ReviewM	lessage			P	riority: 3 (norm	al) 🔻
Pattern:	📵 Simp	ole					-
Title:	Review t	he Messaç	je				
Outcomes:	APPROV	E,REJECT					Q
Parameters:							₽×
Parameter		Name		Туре		Editable	
Outcome targ	get:						₽ 🏈
Help						ОК Са	ancel

Click the green plus sign button next to the Parameters panel as shown above to add parameters.

The **Data Object** popup window appears next to the Create Human Task window. Drag the hello and review process data objects into the **Parameters** panel. Select the **Editable** checkbox for the hello data object only. Also drag the reviewOutcome process data object into the **Outcome Target** field.

🅹 Create H	luman Task		×		
General				. 4	
Name: R	eviewMessage	Prio	rity: 3 (normal) 🔻		
Pattern:	a) Simple		•	🔷 Data Object	<u>k</u> 🛛
	eview the Message PPROVE,REJECT		् ् द	Drag Data Objects Target field.] ==
Parameter	Name	Туре	Editable	hello	
HelloObject	hello	HelloTypes.Hello	 ✓ 	review	
ReviewObjec		HelloTypes.Revi		HelloWorldP	
Outcome targ	et reviewOutcome 🗲		OK Cancel	Help	Close

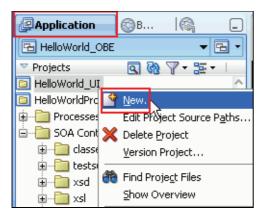
Click **Close** in the Data Object popup window and click **OK** in the Create Human Task window.

Click **OK** in the Properties window.

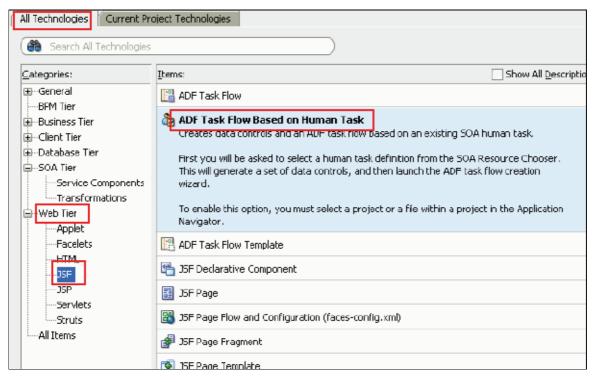
Click Save All.

3. Create a new task flow, based on the ReviewMessage human task that you created in the previous step.

Click on the Application Navigator tab and then right click on the HelloWorld_UI project node and select New...



The New Gallery wizard opens. Click the **All Technologies** tab. Expand the **Web Tier** category and select **JSF**. From the Items panel, select **ADF Task Flow Based on Human Task**. By using this approach to creating the taskflow, you can store the taskflow in the same project that is storing your earlier human taskflow, rather than have separate projects for each taskflow.



Click OK.

You are now prompted to identify the XML file that was generated when you defined the ReviewMessage human task. The **SOA Resource Browser** window opens.

It is currently looking at the <code>HelloWorld_UI</code> folder. Navigate up one level to the <code>HelloWorld_OBE</code> folder and then open the <code>HelloWorldProject</code> folder. Select the file, <code>ReviewMessage.task</code>.

💩 SOA Reso	urce Brow	ser				
📸 File Syste	۲n					-
Location:	🗀 HelloWor	ldProject		•	۵ 🗈	
🔭 da sina		📁 SCA-I				
🗀 .design		🧰 SCA-I				
	esigner dites	_				
🗀 .tasked		🗀 testsu	lites			
	ssCatalog	🚞 xsd				
	ssParameter	🗀 xsl				
🗀 classes			wMessage.task			
🗀 config		📄 SayHe	ello.task			
iib						
🗀 oracle						
proces:						
🗀 resourd	tes					
File <u>N</u> ame:	ReviewMess	age.task				
File <u>T</u> ype:	Task Definit	ion Files (*	.task)			-
Help					ОК	Cancel

Click OK.

In the $\ensuremath{\textit{Create Task Flow}}$ window, accept all the default values and click $\ensuremath{\textit{OK}}.$

Create Task Flow		<u>-</u>
Create a task flow source fi unbounded task flow.	ile whose contents define either a bounded task flow or part of the web applica	ation's
	fer specifically to JSP pages or page fragments, but not both. You can also k flow to be a train at this time.	
<u>F</u> ile Name:		
ReviewMessage_TaskFlow.	xml	
Directory:		
C:\JDeveloper\mywork\Hel	lloWorld_OBE\HelloWorld_UI\public_html\WEB-INF	Browse
Create as Bounded T	Fask Flow	
Task Flow <u>I</u> D:	ReviewMessage_TaskFlow	
Create with Page Fr	ragments	
Cre <u>a</u> te Train		
Base on Template:		-
✓ Update the Task	Flow when the Template Changes	
Нер	ок	Cancel

4. The bounded task flow that you just created, using the wizard, now appears in the Task Flow editor (Diagram tab). It does not yet have a web page associated with the taskDetails2_jspx. This is the reason that you see an error indication. Recall that in the basic Hello World process, you allowed Studio to automatically generate a form for the Request Hello task flow. For the Review Message task flow, you create your own.

Double click the **taskDetails2_jspx** icon in the Task Flow editor. The Create JSF Page window opens. Accept all defaults and click **OK**.

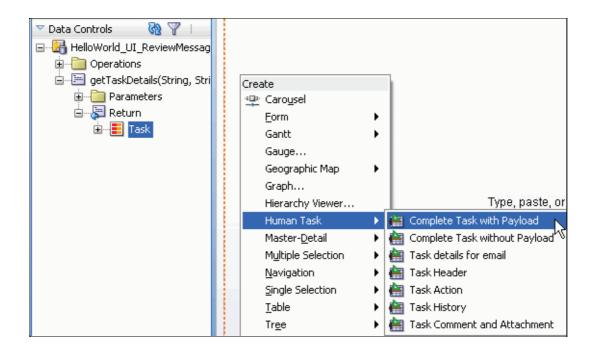
🕹 Create JSF Page 🛛 🔀
Enter the name, directory, and choose a type for the JSF Page. Optionally reference a <u>Page Template</u> to include its content in this page, or apply a <u>Quick Start Layout</u> to add and confidure an initial set of layout components.
File Name: taskDetails2.jspx
Directory: C:\JDeveloper\mywork\HelloWorld_OBE\HelloWorld_UI\public_html Browse
✓ Create as XML Document (*.jspx)
Render in Mobile Device
Initial Page Layout and Content
Ilank Page
O Page Template Oracle Three Column Layout -
O Quick Start Layout
One Column (Stretched) Browse
Page Implementation (UI components are not exposed in managed bean)
Help OK Cancel

It will take several seconds for the JSF page designer to become initialized.

5. You should resize some of the accordion panels in the left side of the Studio window now in order to make the **Data Controls** accordion panel larger, since you will be selecting components from it in this step.

🗢 Data Controls 🛛 🖓 🖓 🛛
🖃 📲 HelloWorld_UI_ReviewMessage
🗄 🖓 🧰 Operations
🛓 🔚 getTaskDetails(String, String, String)

Expand HelloWorld_UI_ReviewMessage > getTaskDetails > Return. Click Task and drag it into the Taskflow Design editor. When you drop it, a menu appears. Select Human Task > Complete Task with Payload.



6. The Edit Action Binding window opens next. If you are unable to expand the HelloWorld_UI_ReviewMessage node, click OK in this window.

A second Edit Action Binding window will appear. Expand HelloWorld_UI_ ReviewMessage > getTaskDetails(...) > Return. Select the Task object. Click OK.

🌢 Edit Action I	Binding		
	ection and the actio ts of the selected o		tiate. The control initiates the action
iaiæ get⊺a iaiæ F		-	
E[Task		k}
Select an <u>I</u> terator	: [taskIterato	r	▼ <u>N</u> ew
Operation:	UpdateAction(SI	ring) 🔻	
	Apply to all	iterators in page defintion	
Parameters : —			
Name	Туре	Value	Option
action	java.lang.Stri	ng	
Help			OK Cancel

After a few seconds, the JSF page appears in the design editor (in design mode). Notice that it contains two boxes, the upper one to accommodate the contents of the message entered by the user, and the lower one to accommodate the review comments by the reviewer.

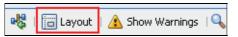
Data	#{date.inputValue}	7	
		\leftarrow	
- Message	#{message.inputValue}		
Review Ob	ject		
Review	#{review.inputValue}		
Reason	#{reason.inputValue}		

You are now finished with the taskflow and JSF page. Click **Save All**. You can close all tabs except for the HelloWorldProcess tab.

Adding Conditional Branching

1. Add an Exclusive Gateway to the process model.

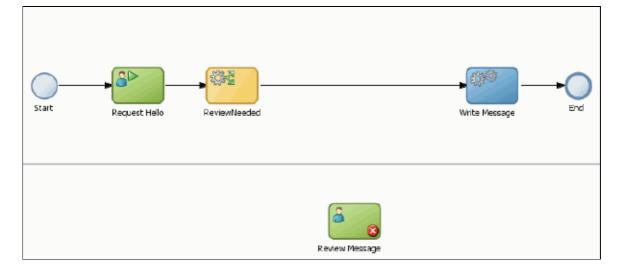
You will be moving design elements around quite a bit in this section and, by default, the Design Editor has Automatic Layout turned ON. You may wish to turn it off so that it does not undo your moves. Click the **Layout** button on the toolbar.



On the Automatic Layout menu that appears, click the **ON** button, which toggles it to an **OFF** button. Note that you can come back to this menu and get a one time auto-layout at any time or choose to turn automatic layout back on.



Make room for changes to the process model by moving the End and Write Message activities further to the right.



From the Gateway accordion panel of the Component Palette, click and drag an Exclusive Gateway, dropping it on the sequence flow between ReviewNeeded and Write Message.

Important: The line must appear **blue** when you drop the gateway in order for the transition line to connect to it properly. You will probably have to drop it near the center of the space between activities in order to pick up the blue line. You can move the gateway further to the left after you have finished placing it.

	8	
ReviewNeeded		Write Message

When you drop it, the Properties window opens. Change the name of the gateway to g1 and click OK.

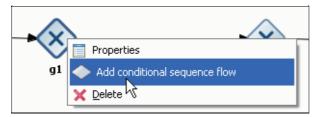
🖕 Properties - ExclusiveGateway	X
Basic Outflows Order	
Name: g1	6
Description:	9
Sampling Point	
<	>
Help OK Cance	

Move the g1 gateway over to the left to make room for another exclusive gateway that you will add in the next step.

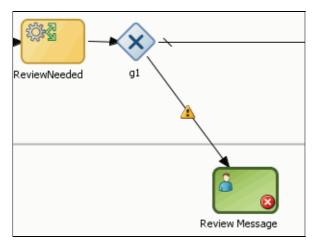
2. Add a second exclusive gateway to the transition line between g1 and Write Message. Follow the same procedure you did in the last step. Name this gateway g2.

Start	Request Hello	Reviewveeded	• X -		e a	Write Message	End
				Review Message			

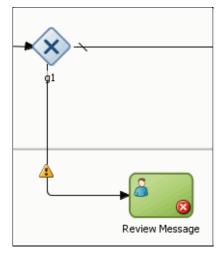
3. Add a conditional sequence flow from g1 to Review Message. Right click on g1 and select Add conditional sequence flow.



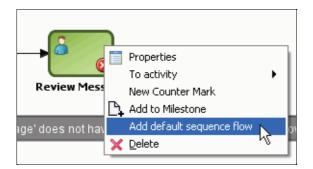
Connect the transition line by clicking on **Review Message**. Do not worry about the warning message regarding lack of default sequence flow. You will fix that in the next step.



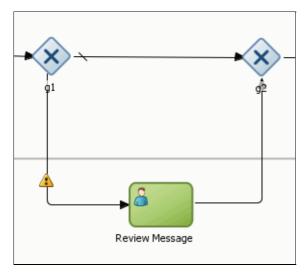
Reshape the transition line as shown below by clicking and dragging it into the desired shape.



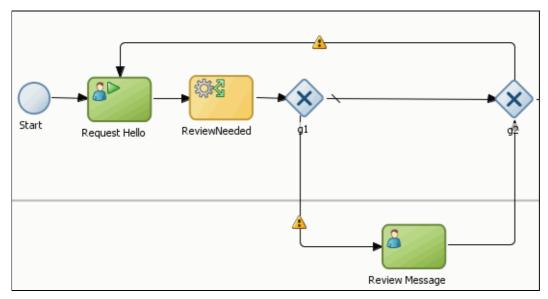
4. Add a default sequence flow from Review Message to g2. Right click on Review Message and select Add default sequence flow.



Connect the other end of the sequence flow by clicking on g2. Reshape the transition line as shown below, as you did in the previous step.



5. Add a conditional sequence flow from g2 back to **Request Hello**. Reshape the transition line as you have the other two transition lines. (It will be obscured by the existing main sequence flow) It should look like this when you've finished.



6. Define the condition for the sequence flow from **g1** to **Review Message**. Double click on the transition line between the two objects. The Transition properties window opens.

On the **Description** tab, name the transition **Review Needed**.

🝲 Transition from Activity: 'g1' to Activity: 'Review Message'	
Description Properties	
Name: Review Needed	

Click the Properties tab. On this tab, you define an expression whose outcome determines whether the process flows down this transition to the Review Message activity. In other words, you define the condition for the conditional sequence flow. Click the Expression Builder button on the right side of the window.

🔷 Transition from Activity: 'g1' to Activity: 'Review Message'	
Description Properties	
Туре	
Condition	-
Expression:	
Simple XPath	
	~

The Expression Builder opens. Build the expression in 3 steps:

- 1. In the Variables panel, expand **review** and select the **review** attribute.
- 2. Click the Insert Into Expression button. This puts review.review into the Expression panel at the top.
- 3. In the Expression panel, add:
 - = "true" to the existing variable reference.

🕌 Expression Builder	X
Expression:	Ø 🕲 🗋
review.review = "true" 3	
2> Insert I	nto Expression
¥ar	iables
A1 14 1	
HelloWorldProcess	
Content Preview:	Description:
review.review	Data Declaration
Help	OK Cancel

Click $\ensuremath{\text{OK}}$ when finished to close the Expression Builder.

Click **OK** to close the Transition properties window.

7. Define the condition for the sequence flow from g2 to **Request Hello**. Double click on the transition line between the two objects. The Transition properties window opens.

On the **Description** tab, name the transition **Message Rejected**.

े Transition from A	ctivity: 'g2' to Activity: 'Request Hello'	
Description 📄 Prope	rties	
Name:		
Message Rejected		
Description:		

Click the Properties tab. Click the **Expression Builder** button to open the Expression Builder and define the following expression, using the same procedure as you did in the last step:

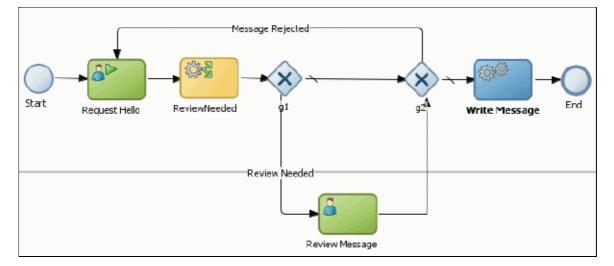
reviewOutcome = "REJECT"

🛓 Expression Builder				X
Expression:	_	ß	ଷ୍ପ	
reviewOutcome = "REJECT"				
	•			
A Insert I	nto Expression			
	iables			
H 🖽 🔚 🖛 🚥				
🖃 😴 HelloWorldProcess				
ÈÈ				
Content Preview:	Description:		\mathbb{R}	
			Ů	
Help	ОК		Cance	
			Cance	21

Click $\ensuremath{\text{OK}}$ to save the expression.

 $\label{eq:click} \mbox{Click } \textbf{OK} \mbox{ in the Properties window.}$

8. Your process should now look like this:

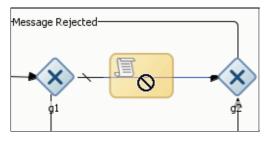


Adding a Script Task

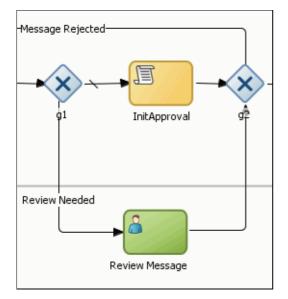
1. When the process flow reaches the second gateway (g2), it checks the value of the *reviewOutcome* variable. There must be a value in the reviewOutcome variable in order for the process to move on to the Write Message activity. If the value is REJECT, the process returns to Request Hello. When a message is flagged for review, the reviewer clicks either REJECT or ACCEPT, thereby populating the *reviewOutcome* variable.

If the process skips the review branch and moves from the g1 gateway directly to the g2 gateway, the *reviewOutcome* variable has no value.

Add a script task to initialize reviewOutcome between g1 and g2. From the Activities accordion panel of the Components Palette, click and drag a Script task, dropping it on the transition line between g1 and g2.



When the Properties window appears, name the activity InitApproval on the Basic tab and click OK.



2. Define the implementation for the InitApproval task. Right click on InitApproval and select Properties.

When the Properties window opens, click the **Implementation** tab. Then select the **Use Associations** checkbox. Click the pencil icon to open the Data Associations window.

🔷 Properties - InitApproval	×
Basic Implementation	
Implementation Type: 🛐 Script task	•
Data Associations	
Use Associations Type: Simple	
Use Transformations	

In the Data Associations window, drag the **reviewOutcome** process data object from the right panel into the gray box in the left panel labeled **"Custom Assignments**". In the text field to the left of the gray box, provide an initial value for the *reviewOutcome* variable of **"ACCEPT"**.

🕹 Data Associations	
Drag variables from the right panel into fields on the left. Tag variables here. Custom Assignments Reccept" Reccept Record assignments Record	HelloWorldProcess
Нер	OK Cancel

Click OK in the Data Associations window.

Click **OK** in the Properties window.

Click Save All. The Hello World process is now complete!

Deploying and Testing the Application

In this section, you deploy the Hello World application to the BPM engine running in the WebLogic server that is part of your SOA installation. This tutorial assumes that your server is running on a remote Linux machine. You need to know:

The server hostname and port

The WebLogic domain in which SOA is running

The username and password for the WebLogic administrative user

The name of the specific WebLogic server to which you will deploy. This tutorial assumes a single server (i.e. not managed instances) configuration and will deploy to the Admin Server.

Before deploying the application, you connect to the internal LDAP realm within the WebLogic server and map the Reviewer and Requester roles to a user in the LDAP.

After deployment, you run the Oracle BPM Workspace web application to test the Hello World application.

Mapping the Studio Role to an LDAP Role Deploying the Process Testing the Process in Workspace

Mapping the Studio Role to an LDAP Role

1. In the BPM Project Navigator, expand the HelloWorldProject and right click on the Organization node, selecting Open.



It may take several seconds to open the Organization editor.

2. Add user members to the Requester role. In the **Organization** editor, select the **Requester** role. Click the green plus sign button to the right of the **Members** panel.

Organization		
Roles		
Name		
Process Owner		T
Requester		· 🖌
Reviewer		~
Role		
Name: Requester		
-Members-		
		Type: User 💌
Name	Туре	
	Туре	
		~~ ~

The **Identity Lookup** window opens. Studio is not yet aware of your remote server, so the only application server that appears is the integrated weblogic server that was installed with JDeveloper.

Create a new profile for your remote server. Click the green plus sign icon next to the **Application Server** field to launch the **Create Application Server Connection** wizard.

🛓 Identity Look	ир 🔀
Application Server:	IntegratedWebLogicServer (Resource Palette Connection)
Realm:	▼
Search <u>P</u> attern:	* User Name 🔹 🧕
Search User	
Select	Hierar Repor Detail << >>

3. In the Name and Type page of the wizard, enter Remote_WLServer as the Connection Name. Make sure that the Connection Type is WebLogic 10.3, and click Next.

े Create Application S	Server Connection - S	Step 1 of 5			X
Name and Type					
Name and Type Authertication Configuration Test Finish	Specify a unique nam Create connection in: Connection Name: Connection Type: WebLogic 10.3 •	e and type for the connection. IEE Connections	The name must	be a valid Jav	va identifier,
Heb		< Back	<u>N</u> ext >	Einish	Cancel

In the Authentication page of the wizard, enter weblogic as the Username and welcome1 as the Password (or substitute your weblogic password). Click Next.

Create Application S	Server Connection - Step 2 of 5
Authentication	
Authoptication	Specify a username and password to authenticate the connection.
Authentication Configuration Test Finish	Username: weblogi: Password: ••••••••
Help	< <u>B</u> ack <u>N</u> ext > Enish Cance

In the Configuration page of the wizard, enter your **Weblogic Hostname**, accept the default values for Port and SSL Port, and enter domain1 in the **Weblogic Domain** field. Click **Next**.

Create Application S	Gerver Connection - Step 3 of 5
Configuration	
Name and Type Authentication Configuration Test Finish	WebLogic Server connections use a host name and port to establish a connection. The Doman of the target will be verified Weblogic Hostname (Administration Server): host01.example.com Port: SSL Port: 7001 7002
Help	< <u>Back</u> <u>N</u> ext > <u>Finish</u> Cancel

In the Test page of the wizard, click the **Test Connection** button. You should see results similar to those shown below. If not, click the **Back** button, correct any errors and test again.

Test		
Name and Type Authentication <u>Configuration</u> Test	Click Test Connection to determine if the in connection with the application server. <u>Iest Connection</u> <u>Status:</u>	nformation specified successfully establishes a
Einish.	Testing JSR-160 Runtime Testing JSR-160 DomainRuntime Testing JSR-88 Testing JSR-88-LOCAL Testing JMD1 Testing JSR-160 Edit Testing HTTP Testing Server MEasure Model Testing HTTP Authentication 9 of 9 tests successful.	SUCCESS. SUCCESS. SUCCESS. SUCCESS. SUCCESS. SUCCESS. SUCCESS.

Click Finish.

You are now returned to the Identity Lookup window.

4. Select the **Remote_WLServer** profile from the **Application Server** list. After a few seconds, Studio will connect to the LDAP server in the remote server and you see **jazn.*** appearing in the **Realm** field.

Click the browse icon next to the **User Name** field to bring up a list of all users in the Demo Community from the LDAP server. Select **jcooper** from the list and click the **Select** button.

緍 Identity Look	ир 🔀
Application Server:	Remote_WLServer (Resource Palette Connection)
Realm: —>	jazn.com 💌
Search <u>P</u> attern:	* User Name 🔹 🖸
Search User	
eheming fkafka istone	~
jausten jcooper	
	Mier Rep Detail < >>
jcooper 🗲	
	R <u>e</u> move De <u>t</u> ail
Help	

jcooper will appear in the Selected Users panel of this window. Click **OK**. You are returned to the Organization editor.

Roles		
Name		
Process Owner		
Requester		
Reviewer		
Role		
Name: Requester		
Members		
		Type: User 💌
Name	Туре	4
jcooper	User	
		~~

5. Follow the same procedure as in the previous step to add jcooper to the **Reviewer** role as well.

Roles		
Name		4
Process Owner		<u> </u>
Requester		
Reviewer		
Role		
Name: Reviewer		
Members		
		Type: User 💌
Name	Туре	
jcooper	User	+ ×
		6.6

6. Click Save All and close the Organization editor.

Deploying the Process

1. Deploy the HelloWorldProject.

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

C Applie	ation BPM Pro 🚱 🕞	SayHelloWorldProcess
Hello	World_OBE 🛛 👻 🔂 🔹	
Project	s 🖪 🗞 🖓 • ﷺ • 🗌	
	elloWorld_UI elloWorldProject	
	 New Edit Project Source Paths Delete Project Version Project 	
	Find Project Files Show Overview	equest Hello ReviewNeed
	Make HelloWorldProject.jpr	Ctrl-F9 Alt-F9
		HelloWorldProject
Applica	▶ Run ∰ Debug	

The Deploy HelloWorldProject wizard opens.

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

Deployment Action	DIDIO10101010101010101010105555
Deployment Action	Select a deployment action from the list below.
Deploy Configuration Summary	Deploy to Application Server Deploy to SAR
Help	< Back Next > Finish Cancel

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

Deploy Configuration		
Deployment Action Deploy Configuration Task flow deployment Summary	HelloWorldProject Composte Revision ID – Project: HelloWorldProject Current Revision ID: 1.0 New Revision ID: 1.0	SOA Configuration Plan Do not attach Select a configuration plan from the list.
t>		as default. composites with the same revisid ID. onfiguration plan for all composites: Browse Cancel

In the Task flow deployment page, select the checkbox next to **Projects**. This selects all taskflow projects in the HelloWorldProject, so you will see the HelloWorld_UI project selected now also.

Tasl	k flow deployment	t		01010101	01010101010	14949169340	
	Deployment Action <u>Deploy Configuration</u> T ask flow deploymer <u>Select Server</u>	Ear Profile Name:		posite revision to ted profies to app AR			
Ó	Summary	Optional: Select WA Deployable Taskflov	v Projects	eck projects to ex WAR Profiles	clude from	deployment App Context Ro	ot
	R		elloWorldProject jpr [ł	HelloWorld UI	T	/workflow/HelloV	Vorld_U1
Ŀ	telp			< <u>B</u> ack	<u>N</u> ext >	Einish	Cancel

Click Next.

In the Select Server page, select Remote_WLServer and click Finish. Deployment will begin.

Select Server	816161616161616161616161616161616161616	
Deployment Action Deploy Configuration Task flow deployment Select Server SOA Servers Summary	Application Servers: IntegratedWebLogicServer (domain unconfigured) Remote_WLServer VLServer VLServer	
Help	< Back Next > Finish	Cancel

Check the **Deployment** tab in the **Log** panel in the lower central portion of the Studio window to watch the progress and determine when deployment has finished. This first deployment of the SOA Composite (shown as sca_HelloWorldProject_rev1.0.jar in the log viewer screenshot below) will be very quick. The deployment of the WAR file containing the taskflows (HelloWorld_UI.war) will take much longer. Total deployment time will be around 15 minutes, depending upon your environment.

	[11:58:38 AM]	Successfully deployed archive sca_HeiloWorldProject_rev1.0.jar to partition "de
	[11:59:03 AM]	Retrieving existing application information
	[12:07:41 PM]	Deploying Application
	[12:09:41 PM]	[Deployer:149192]Operation 'deploy' on application 'HelloWorld_UI' is in progre
	[12:12:20 DM]	[Deployer:149194]Operation 'deploy' on application 'HelloWorld_UI' has succeede
	[12:12:36 PM]	Application Deployed Successfully.
	[12:12:49 PM]	The following URL context root(s) were defined and can be used as a starting po
	[12:12:49 PM]	HelloWorld_UI.war
	[12:12:49 PM]	/workflow/HelloWorld_UI
_	[1Z:1Z:49 PM]	HelloWorld_UI.war
Log	[12:12:49 PM]	/workflow/HelloWorld_UI
÷	[12:12:50 PM]	Blapsed time for deployment: 15 minutes, 18 seconds
ne	[12:12:50 DM]	Deployment finished
loyi		The second se
Deployment	<	
	Messages E	ktensions Deployment SOA

Testing the Process in Workspace

1. Open a browser (either on the server or from your windows machine) and enter the following URL:

```
http://<your server hostname>:7001/bpm/workspace
```

When the Welcome page of the Workspace appears, enter jcooper in the **Username** field and welcome1 in the **Password** field. Click **Log In**.



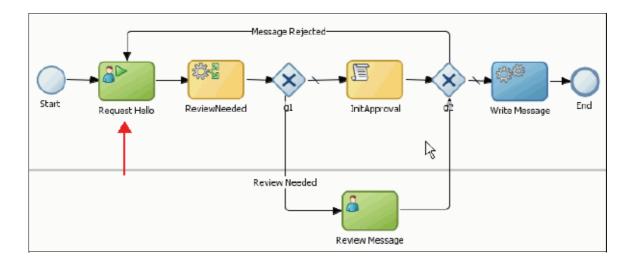
The main Workspace window opens.

2. Instantiate the process by clicking the HelloWorldProcess v1.0 link under Applications in the Tasks tab on the left side of the window.

Note: If the HelloWorldProcess v1.0 link does not appear in the Applications panel on the Tasks tab, click the **Process Tracking** tab. It should appear there. You can instantiate the process from that tab.

Tasks Process Tracking	Standard	Dashboards	-	
Applications		My Tasks	Initi	iated Tasks
	ଜ୍ୟ	Actions *	G ₀	Assignee N
[HelloWorldProject] HelloWorldProcess v	.0	Title		
⊻ Worklist Views	/%			
V 👺 My Work Queues				
V Standard Views				
📃 Due Soon				
High Priority				
E Past Day	=	(1)		
🖽 Past Week				

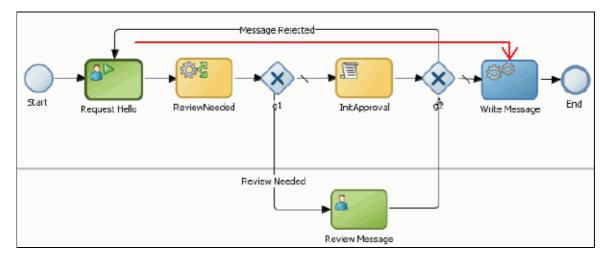
This action executes the first task of the process, Request Hello.



3. First enter a message that will *not* require review. In the **Please Enter a Hello Message** popup window that appears, enter a value in the **Date** field using the format MMM d, yyyy (example: Jun 2, 2010) Also enter a value in both the **Greeting** and **Message** fields (Greeting field length > 5, Message field length > 5). Click **Submit**.

	ra Hello Message	Actions 🔻	SUBMIT	Claim
De 🎽 De	etails (j)			
Contents				
Date	Jun 2, 2010			
Greeting	Hello World			
Message	Have a wonderful day!			
	1			

After submitting the message, the process goes to the ReviewNeeded business rule. Since both the Greeting and the Message are considered "Medium" in length, the message does not require a review. Hence the process goes through the g1 gateway, the *reviewOutcome* variable is initialized in the script task, and the process flows on through the g2 gateway to the **Write Message** activity.



The Write Message activity is a service type activity, not an interactive activity., therefore there will be no indication

of its action within the Tasks panel of the Workspace.

4. Open the file that was created by the Write Message implementation. Recall that you configured the file adaptor to write the file to the file path "." (dot). This is relative to the domain1 directory within the WebLogic file structure.

On the server, using either a terminal window or an Explorer style window as shown here, navigate to:

<path to your mwhome>/mwhome/user_projects/domains/domain1

Find the file Hello_1.xml. Open it in a text editor to view the outcome from your process.

👂 📁 sysman	
Imp	
D 📁 UMSJMSFileSt	ore
🗍 edit.lok	
[] fileRealm.pro	perties
Hello_1.xml	Open with "Firefox Web Browser"
🔊 startWebLogi	Open with "Emacs Text Editor"
Þ 📁 utils	Open with "Text Editor"
V 📁 wiserver_10.3	Open with Other <u>Application</u>
domain-registry.xml	Copy
🚺 ocm.rsp	- C Zoby
T registry dat	Ma <u>k</u> e Link

It should look something like this:

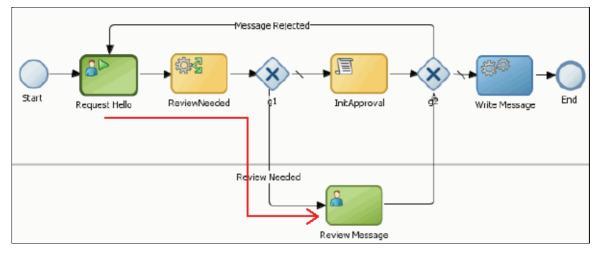
Hello_1.xml ×
xml version="1.0" encoding="UTF-8" ? <helloobject< th=""></helloobject<>
<pre>xmlns="http://xmlns.oracle.com/bpm/bpmobject/HelloTypes/</pre>
HelloObject">
<pre><date xmlns="">2010-06-02T00:00:00Z</date></pre>
<preeting xmlns="">Hello World</preeting>
<message xmlns="">Have a wonderful day!</message>

5. Create another instance of the HelloWorld process by once more clicking the HelloWorldProcess v1.0 link. This time, enter a message that will be routed to the Review Message activity.

When the **Please enter a hello message** window appears, enter a date and some text that has a *length* < 5 in the **Greeting** field. Leave the **Message** field blank. Click **Submit**.

Please enter a Hello message	•
Please enter a Hello message	Actions - SUSMIT Claim
≥ 🦀 Details 🕕	.0
✓Contents	
Date Jul 12, 2010	
Greeting Hi	
Message	

6. The process now flows, as before, to the business rule where the greeting is found to be "Short". As you'll recall, when the greeting is "short", the message requires review, regardless of what is in the Message field. Consequently, the process now flows to the Review the Message activity, which is assigned to the Reviewer role.



The **Review the Message** activity now appears in the **My Tasks** tab (remember that jcooper also has the **Reviewer** role). Click it to see the details in the lower panel of this tab.

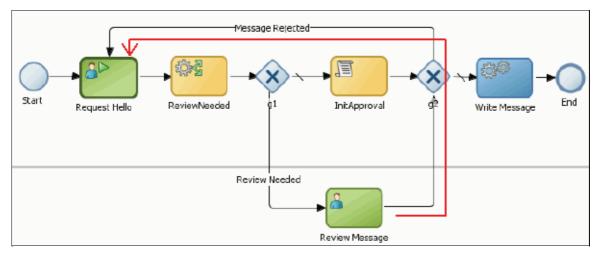
My Tasks Initiated Tasks Administration Tasks						
Actions 🗸 🛛 🙀 🛛 Assignee Me & Group	🐱 Status	Assigned	×	Search		*
Title	Number	Priority	Assignees	State	Created	
🗓 Review the Message	200064	3	HelloWorldPre	ojec Assigned	Jul 12, 2010	6:04 AM
4						
Ť						

In the details panel, you can see the taskflow form that you created for this activity, containing form elements for both the HelloObject and the ReviewObject. Notice that the ReviewObject attributes have been assigned by the Business Rules engine after executing the business rule for this message.

▶ Details i ▶ Contents Date Jul 12, 2010 Greeting Hi Message Message Review Object Review true Reason Greeting is too short ♥ History Image Task Snapshot	Claim
Date Jul 12, 2010 Greeting Hi Message Message Review Object Review true Reason Greeting is too short ✓ History Image: Task Snapshot ✓ Task Snapshot ✓ Future Participants # Participant Action Updated By Action Date	
Greeting Hi Message Image: Comparison of the state o	
Message Review Object Review true Reason Greeting is too short ✓ History I Task Snapshot ✓ Future Participants Full task actions # Participant Action Updated By Action Date	
Review Object Review true Reason Greeting is too short History Image: Task Snapshot I	_
Review true Reason Greeting is too short History Image: Task Snapshot	
Image: Task Snapshot Image: Full task actions Image: Task Snapshot Image: Full task actions Image: Task Snapshot Image: Action Snapshot Image: Action Snapshot Image: Task Snapshot Action Image: Action Snapshot Action Data	
# Participant Action Updated By Action Da	
	te
1	
1.1 Assigned workflowsystem Jul 12, 20	10

Click Reject when you've finished examining the details.

7. The process now flows to the g2 gateway where the value of the reviewOutcome variable is examined. It has been set to REJECT by the reviewer, so the process now flows back to Request Hello.



The **Please enter a hello message** link appears in the My Tasks panel. This is the Title string for the human taskflow associated with the **Request Hello** task. Previously, this window was launched immediately upon instantiation of the process. This time it appears within the My Tasks panel.

My Tasks Initiated Tasks Administration Tasks							
Actions 🗸 🛛 🙀 🛛 Assignee 🛛 Me & Group	 Status 	Assigned	V		*		
Title	Number	Priority	Assignees	State	Created		
🔋 Please enter a Hello message	200065	3	HelloWorldProjec	Assigned	Jul 12, 201		

Click the link to view the input form and enter a more appropriate message that will *not* require review as you did in Step 3. Click **Submit** when finished.

The process will now flow through the two gateways to the Write Message activity and to the End activity. The My