

# **Oracle® Business Process Management**

Oracle BPM Tutorial

10g Release 3 (10.3.1)

January 2009

Copyright © 2006, 2008, 2009 Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

# Contents

<b>Overview.....</b>	<b>5</b>
<b>Basic Concepts.....</b>	<b>6</b>
Business Process Overview.....	6
Process Instance Overview.....	6
Flow Object Overview.....	7
Flow Object Naming Conventions.....	8
Transitions Overview.....	8
<b>The Expense Report Process.....</b>	<b>10</b>
Process Description.....	10
Process Design.....	10
Interactive Activities.....	11
Automatic Activities.....	12
Transitions.....	12
<b>Activity 1: Initial Steps.....</b>	<b>13</b>
Creating a Project.....	13
Creating a Process.....	14
Creating a Role.....	14
Creating a Role from the Process Editor.....	15
Adding a Global Creation Activity.....	16
Adding Participants.....	17
Running the Process.....	18
Activity 1 Summary.....	19
<b>Activity 2: Building the Happy Path .....</b>	<b>20</b>
Adding the Check Company Policy Activity.....	20
Adding the Review Report Activity.....	21
Adding the Confirm Receipts Activity.....	22
Adding the Process Expense Activity.....	23
Running the Process as an Employee.....	23
Running the Process as Supervisor and Treasurer.....	24
Activity 2 Summary.....	25
<b>Activity 3: Defining the Expense Report Object.....</b>	<b>26</b>
Creating a BPM Object.....	27
Defining BPM Object Attributes.....	28
Defining a BPM Object Group.....	29
Defining a Virtual Attribute.....	30
Defining a Valid Values List.....	31
Creating the Process Variable.....	32
Creating a Presentation.....	33
Refining a Presentation.....	34
Creating a Screenflow.....	35

- Designing a Screenflow.....37
- Running the Process.....38
- Activity 3 Summary.....39
- Activity 4: Adding Alternative Paths.....40**
  - Defining the Check Company Policy Task.....40
  - Adding a Condition Transition.....41
  - Adding the Edit Report Activity.....42
  - Adding Condition Transitions to Edit Report.....43
  - Defining the Review Report Presentation.....44
  - Defining the Review Report Activity.....46
  - Designing the Review Report Screenflow.....47
  - Adding the Not Confirmed Transition.....48
  - Creating the Confirm Receipts Presentation.....50
  - Defining the Confirm Receipts Activity & Screenflow.....52
  - Running the Process.....52
  - Activity 4 Summary.....53
- Activity 5: Finishing Touches.....54**
  - Adding a Due Transition.....54
  - Implementing the sendReminder Method.....55
  - Presetting Default Values.....56

# Overview

---

This tutorial presents a step-by-step introduction to Oracle BPM Studio. It shows you how to model and deploy simple business processes.

## Basic Requirements

To complete this tutorial, you need to have installed Oracle BPM Studio 10.3. The latest version of Studio is available in the [Oracle Downloads](#) page.

BPM Studio provides a separate profile for each of three specific user types: one profile for Business Analysts, one for Business Architects, and another for Developers. Each profile contains a different level of functionality. In this tutorial, you must use the Developer profile. If you use either of the other two profiles, you will not be able to complete the tutorial.

To select the Developer profile, on the Studio toolbar, choose **Help**, then choose **Welcome**, then choose **Developer**.

## Using the Tutorial

- If you are not familiar with Oracle BPM or BPM concepts in general, begin by reading [Basic Concepts](#) on page 6.
- Once you have a basic understanding of Oracle BPM, read [The Expense Report Process](#) on page 10. This section presents both the business process to be modeled and the process as designed with Oracle BPM.
- Five activities follow, each with a number of tasks. They enable you to build the ExpenseManagement project together with the ExpenseReport process. The activities are:
  1. [Activity 1: Initial Steps](#) on page 13
  2. [Activity 2: Building the Happy Path](#) on page 20
  3. [Activity 3: Defining the Expense Report Object](#) on page 26
  4. [Activity 4: Adding Alternative Paths](#) on page 40
  5. [Activity 5: Finishing Touches](#) on page 54

## Basic Concepts

---

If you are not familiar with Oracle BPM Studio or BPM concepts in general, read the topics in this section. If you have used previous versions of Studio, you can skip this section.

### Business Process Overview

A business process is a sequence of business tasks and activities that, when executed, produces a well-defined outcome. Once this outcome is achieved, the process is complete.

A simple business process can involve hiring an employee, processing a sales order, or reimbursing a business expense. A more complex business process can involve many people and activities across an organization.

Sometimes the main goal of a process cannot be achieved. For example, if a product is out of stock, a shipping clerk may need to cancel a sales order. For this reason, a business process must provide for outcomes other than the principal goal. For example, if the product is out of stock it may be possible to offer the client an alternative that the client can then accept or reject. Thus, a process can have a range of possible outcomes.

#### Activities

Business processes include logical steps, called *activities*, each of which can involve performing one or more *tasks*.

There are two types of activities: automatic and interactive. *Automatic activities* are executed automatically by the Process Execution Engine, whereas *interactive activities* require human input.

The activities of a business process are linked by *transitions*, which determine the order in which they are performed and the basic workflow of the process.

#### Roles and Participants

Each interactive activity belongs to a *role*, that is, a title or job function performed by participants in the organization. For example, a role could be *Supervisor* or *Finance Administrator*.

*Participants* are the individuals who interact with the process. To perform an activity, a participant must be assigned the role that the activity belongs to. A participant can have one or more roles.

#### Exceptions

Because it is often impossible to predict every outcome, a business process usually needs a way to deal with *exceptions*. An exception is an event in which a pre-defined outcome of a process cannot be reached.

The way in which a process deals with such an event, known as *exception handling*, can involve such steps as data clean-up or notifying a participant with a supervisory role that the situation needs attention.

### Process Instance Overview

A *business process* is a sequence of steps. A *process instance* is a specific item moving through those steps.

As the instance proceeds through the process, it is acted upon by various participants or processed automatically by software. For example, in a business process that handles purchases, each purchase order would be a process instance acted upon by such participants as shipping clerks, supervisors, and finance administrators.


Any number of instances can traverse a business process. For example, any number of purchase orders can traverse a business process that handles purchases.

Every instance has a specific history and properties. For example, a purchase order usually contains such data as a customer name, a list of items, an amount due, and dates of delivery and payment.

An instance can also have various status conditions. In the case of a purchase order, you want to know if it has been approved, billed, or paid, or if the requested products have been shipped.

Finally, each instance has a beginning and an end as defined in the business process.

 **Note:** In order to understand what a business process *instance* is, you must first understand the concept of a [Business Process Overview](#) on page 6.

 **Note:** In user interfaces designed for end users—for example, Oracle BPM WorkSpace—instances are also called *work items*.

## Flow Object Overview

A flow object models a step in a business process.

The following table describes different categories of flow objects:

Category	Description	Flow Object
Activity	Activities represent the work that companies perform.	<ul style="list-style-type: none"> <li>• Interactive</li> <li>• Decision</li> <li>• Automatic</li> <li>• Group</li> <li>• Subflow</li> <li>• Process Creation</li> <li>• Termination Wait</li> <li>• Grab</li> </ul>
Gateway	Gateways control the divergence and convergence of the process flow. They determine if the paths branch, fork, merge or join to other paths.	<ul style="list-style-type: none"> <li>• Conditional</li> <li>• Split</li> <li>• Or-Split</li> <li>• Multiple</li> </ul>
Event	Events affect the process flow. They happen during the course of a business process. They generally have a cause and an impact.	<ul style="list-style-type: none"> <li>• Message Wait</li> <li>• Send Message</li> <li>• Timer</li> <li>• Compensate</li> </ul>
Global Activity	Global activities handle global requirements that are not associated with a specific process instance.	<ul style="list-style-type: none"> <li>• Global Creation</li> <li>• Global Automatic</li> <li>• Global Interactive</li> </ul>
Artifact	Artifacts provide additional information about the process.	<ul style="list-style-type: none"> <li>• Measurement Mark</li> </ul>

## Flow Object Naming Conventions

Providing descriptive names for your flow object allows your process to be self-documented.

Oracle recommends that you name your activity descriptively with a verb followed by a noun specifying the function of the activity within the process. Some examples of useful activity names are: *Create Order*, *Ship Product*, and *Check Credit*.



**Note:** After you define an activity name, you cannot change it. However, you can change the activity label displayed to end users.

## Transitions Overview

A transition advances the process from one flow object to another. In Business Process Modeling Notation (BPMN), transitions are also known as connecting objects.

Transitions use directional arrows that display the direction of the flow. An instance flows through a process by following the logic that applies to a transition.

### Transition Types

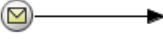
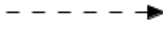
Oracle BPM provides many types of transitions. The most common transitions are: *Unconditional*, *Conditional*, *Due Exception*, *Business Rule*.

Notation	Transition	Description
	Unconditional (Uncontrolled)	Instances flow through the transition without being affected by any conditions. In Oracle BPM, this is known as an uncontrolled flow.
	Unconditional (Default)	Instances flow through this transition when alternative condition transitions are not used. Oracle BPM automatically shows a default unconditional when at least one alternative condition flow is added to the flow object.
	Conditional	Instances flow through the transition if a specified condition is met.
	Business Rule	Instances flow through the transition if the specified dynamic business rule evaluates to true.
	Due (Timer)	Instances flow through the transition when a timer fires.
	Exception (Error)	Instances flow through the transition if an exception occurs.

The transitions in the following table—*compensate*, *message-based*, and *precedence*--are used less frequently. If you are just beginning to use Oracle BPM, you do not need to be familiar with these yet.

Notation	Transition	Description
	Compensate	Instances flow through the transition if compensation processing is required. The actions performed reverse (or undo) any work done in the previous flow object in the event that PBL-Method failure occurs.



Notation	Transition	Description
	Message Based	Instances flow through the transition if a flow object that handles different argument sets receives a message. Available only from Begin or Message Wait events.
	Precedence	Only available in a Split-Join circuit. Copies within a Split-Join circuit can have a synchronization or a precedence. The precedence is represented by a dashed transition line and a solid arrowhead, not to be confused with the BPMN Message Flow, which begins with a circle and has an outline arrowhead.

### Which Transition Is Used?

All flow objects at least have an outgoing unconditional transition so there is always a way to continue the process. However, in most processes, *condition* transitions are also used. When one or more condition transitions originate from a flow object, the remaining unconditional transition is shown as a default flow transition.

In this case, the condition transitions are evaluated first, and the unconditional transition is taken only if the condition transitions all evaluate to `false`. In programming terms, the default unconditional is like the *else* clause in an if-then-else construct.

Business rule transitions are evaluated before condition transitions, so if a business rule transition and a condition transition both evaluate to **true**, the business rule transition prevails.

Due transitions act separately. They "pull" the instance from the flow object as soon as a timer fires. In this case, all other outgoing transitions are ignored.

# The Expense Report Process

---

## Process Description

This tutorial explains how to design and build a business process for a common activity: managing expenses. To design a business process, you must understand the business objective that the process achieves, and the elements (such as people and data) that the process requires.

### Managing Expenses

Expense procedures are simple in principle but involve tradeoffs. A rigid expense reporting system pleases accountants but can slow operations. On the other hand, a company with relaxed expense rules may spend too much, or even lose track of expenditures.

This tutorial shows how a simple process is designed and implemented. Starting simple is convenient not only for the purposes of a tutorial; it is also a useful design approach. You can easily add more capability later on, and further refine the process as you obtain feedback from users. One of the main advantages of using an executable process model is that successive design-test-use iterations can be done quickly.

### Typical Expense Report Sequence

The following sequence of steps roughly describes how an expense report is handled. Pay attention not only to *what* is done but *who* does it, and also note the sequence of events:

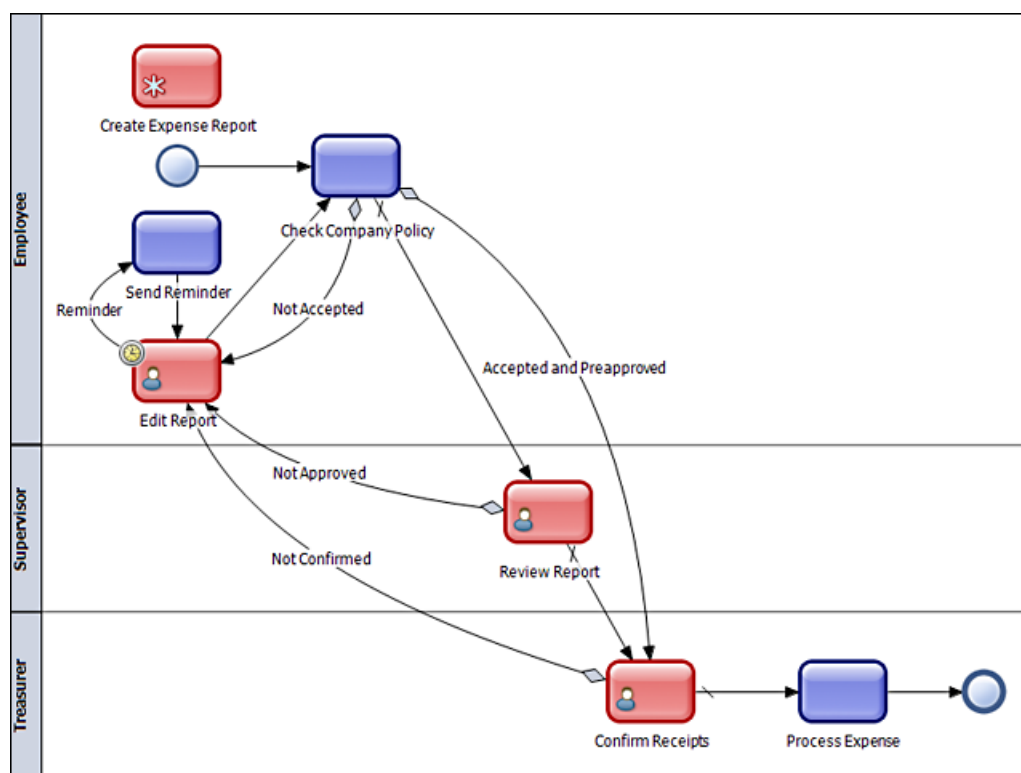
1. An employee purchases a product or service he requires. For instance, a sales person on a trip rents a car.
2. The employee submits an expense report with a list of items, along with the receipts for each item.
3. A supervisor reviews the expense report and approves or rejects the report. Since the company has expense rules, there are circumstances where the supervisor can accept or reject the report upon first inspection. These rules could be automated, to reduce the workload on the supervisor.
4. If the supervisor rejects the report, the employee who submitted it is given a chance to edit it, for example to correct errors or better describe an expense. If the supervisor approves the report, it goes to the treasurer.
5. The treasurer checks that all the receipts have been submitted and match the items on the list. If all is in order, the treasurer accepts the expenses for processing—including, for example, payment or refund, and accounting. If receipts are missing or do not match the report, he sends it back to the employee.
6. If a report returns to the employee for corrections, it must again go to a supervisor, even if the supervisor previously approved the report.
7. If the treasurer accepts the expenses for processing, the report moves to an automatic activity that links to a payment system. The process waits for the payment confirmation.
8. After the payment is confirmed, the process ends.

In the following sections, we describe an Oracle BPM process that implements the above steps and adds some additional features.

## Process Design

The Expense Report process is implemented in the ExpenseManagement sample project, which is included in the Studio installation. This tutorial shows how this Expense Report process is developed.

Once it is complete, the Expense Report process diagram will look like this in the Studio editor:



**Figure 1: Expense Report process diagram**

To examine the Expense Report process in Studio, you may import the `ExpenseManagement.exp` project file included in your Studio installation. By default, it is installed in the following directory: `OraBPMStudioHome\samples\basic\ExpenseManagement.exp`. However, to actually perform the tutorial we recommend that you start from scratch.

## Process Elements

The Expense Report process contains the elements described in this section. You do not need to memorize this list. Before moving to the next section, review this list briefly to get a general notion of the process:

- Three roles, *Employee*, *Supervisor*, and *Treasurer*. Each role has one swimlane in the process diagram, and there are also two unlabeled swimlanes for automatic activities.
- The *Begin* and *End* activities, present in every Oracle BPM process.
- A *global creation* activity, named *Create Expense Report*. This is the activity that the employee uses to create a new report. In business process terms, this is the activity that creates a new instance.
- Four *interactive* activities: *Review Report*, *Confirm Receipts*, and *Edit Report*. These activities require input from a participant, and are described in detail in [Interactive Activities](#) on page 11.
- Three *automatic* activities: *Check Company Policy*, *Process Expense*, and *Send Reminder*. These activities are performed automatically by the system, with no user interaction. Each automatic activity in this process is described in [Automatic Activities](#) on page 12.
- Transitions, which establish the flow of the process. These transitions are used: *Unconditional*, *Conditional*, and *Due*. For detailed descriptions of each, see [Transitions](#) on page 12.

## Interactive Activities

The following interactive activities are defined in the Expense report process:

1. *Review Report* - This activity is performed by the supervisor, who may accept or reject the report.

2. *Confirm Receipts* - This activity is performed by the treasurer, who verifies that each item in the report has a matching receipt.
3. *Edit Report* - This activity is performed by the same employee who originally submitted the report. It is performed if corrections are required because the report was rejected, receipts are missing, or company policy is not met (see below).

## Automatic Activities

The following automatic activities are defined in the Expense report process:

1. *Check Company Policy* - This activity, the first performed after the employee submits a report, can check basic business rules. For example, it may find that the expense amount does not need a supervisor's approval because it is below the threshold specified by the business rules.
2. *Process Expense* - Once the expense report is approved and checked, this activity processes it for payment or accounting. For the purpose of this tutorial, this activity is a placeholder; we won't actually process any expenses.
3. *Send Reminder* - If the expense report returns to the employee, the employee usually has a limited amount of time to re-submit the report. After a specified amount of time, this activity e-mails the employee a reminder that his report is pending.

## Transitions

The following types of transitions are used in the Expense report process:

- *Unconditional* transition - Most of the transitions in the Expense Report process are of this type. The instance flows in the direction of the arrow after an activity has finished.



**Figure 2: Unconditional Transition**

- *Condition* transition - The instance flows through a condition transition if an expression defined for the transition evaluates to `true`. If a default transition also flows from the same activity, the *condition* transition (when `true`) is evaluated first, and the default is taken only if the condition evaluates to `false`.



**Figure 3: Condition Transition**

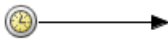
- *Default* transition - The instance flows through a default transition if all other condition transitions evaluate to `false`.

You do not insert default transitions. Instead, an unconditional transition from an activity becomes the default automatically when you add one or more condition transitions to that activity.



**Figure 4: Default Transition**

- *Due* transition - The instance flows through this transition after a specified period of time in the origin activity. In the Expense Report process, this transition triggers the Send Reminder activity.



**Figure 5: Due Transition**

## Activity 1: Initial Steps

In this activity you create the ExpenseManagement project, the Expense Report process, two roles, and a global creation activity.

This activity shows how to begin work on a project and a process. It also helps familiarize you with Oracle BPM Studio. Pay special attention to the **Project Navigator** and the **Process Editor**.

To ensure you do not lose any work, save the project after completing each task. To save all project components in one step, click **File ► Save All** (📁).

### Creating a Project

After evaluating the business need, the first step in modeling a business process is to create a new project. A project contains all of the resources necessary to model and publish your business process.

To create a new Oracle BPM Project:

1. If it is not already open, start Oracle BPM Studio.
2. From the File menu, choose **New ► BPM Project**, or click on the **New BPM Project** icon (🌐) in the toolbar.

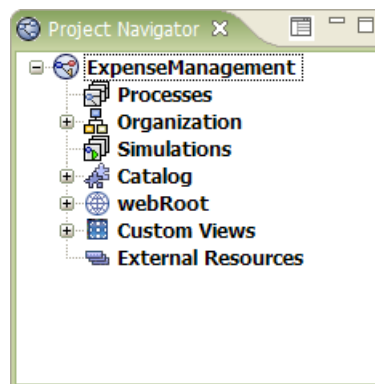
The **New BPM Project** wizard appears.

3. in the **Project Name** field, enter ExpenseManagement.
4. In the **Project Root Folder** field, indicate the location to store project files, or leave the default location shown.
5. Choose **Next**.

The **New Project Information Summary** page appears.

6. Review the project information and choose **Finish**.

The new project is created and added to the **Project Navigator**. If you expand the project, it appears as shown below:





**Figure 6: Project Navigator**

## Creating a Process

A project can have one or more processes. In this task, you will create the Expense Report process of the ExpenseManagement project.

To create the Expense Report process:

1. In the **Project Navigator**, expand the ExpenseManagement project.
2. Right-click **Processes** , then select **New** ➤ **Process** . The **Process** dialog box appears.
3. In the **Name** field, enter Expense Report.
4. Verify that the **Lane Layout** option is set as **Horizontal**.
5. Select the **Advanced** tab.
6. In the **Advanced** tab page, select the **Generate Events for all Activities** option. Selecting this enables you, later on, to see every step the process instance (the expense report) traverses.
7. Choose **OK**.

A new window appears for the new process. This is known as an *editor*. The **Design** view is shown, as you can see by looking at the tab at the bottom of the editor.

The new process contains a *Begin* and an *End* Activity, joined by an unconditional transition. The Begin and End Activities define the entry and exit points for a process.





When you create a new process, these activities, and the unconditional transition that connects them, are created automatically.

## Creating a Role

Roles you create in Studio are known as *abstract roles*, because they may be renamed or consolidated when the process is actually deployed in a production environment. In Studio, roles can be added to the project in the **Project Navigator** or directly in the process editor design window.

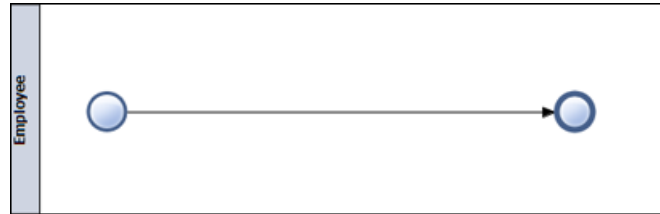
In this task, we create a role from the **Project Navigator**. In the next task, we will create a role from the process design diagram.

To create a role:

1. In the **Project Navigator**, expand the ExpenseManagement project if it is not already expanded.
2. Expand the *Organization* section .
3. Right-click *Roles*  and select **New** . The **Role Properties** dialog appears.
4. Enter Employee in the **Name** field, then click **OK**.  
The Employee role is created. It is listed in the **Project Navigator** under *Roles*. In the right pane, an editor opens for the Employee role.
5. In the editor, you can enter a label for this role in the **Label** field. Leave the default label value, which is Employee. You can also enter a description for this role in the **Description** text box. This is optional, so leave it blank.
6. Leave the **Parametric** option unchecked, and close the role editor.
7. In the **Design** view of the Expense Report process editor, right-click somewhere above the Begin activity, and click **Add Role** .  
The **Role Properties** dialog appears.

8. From the **Name** drop-down list, select *Employee* and click **OK**.  
The Employee role swimlane is added at the top of the process design diagram.
9. By clicking and dragging, move the Begin and End activities to the Employee swimlane, placing the End activity farther towards the right.
10. Right-click on the header of the automatic swimlane (the header with no label), and click Delete (X).  
The automatic swimlane is deleted.

After completing this task, the process design diagram should look like this:



**Figure 7: Employee role in Expense Report process**

Save your changes before proceeding to the next task.

## Creating a Role from the Process Editor

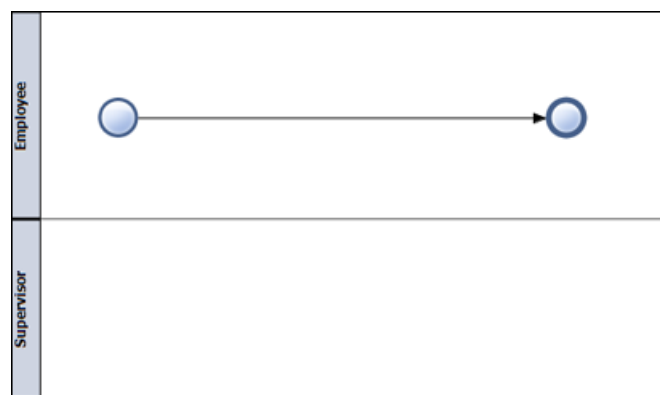
This task shows you how to create a role in the project's organization, from the **Design** view of the Process Editor.

In this task we create the Supervisor role. The supervisor gives initial approval to the expense report submitted by the employee. We create this role directly from the process diagram, to show an alternative to the method used in the previous task.

To create the Supervisor role from the Process Editor:

1. In the **Design** view of the Expense Report process editor, right-click somewhere in the lower half of the Employee swimlane, and click **Add Role**.  
The **Role Properties** dialog appears.
2. Click **New**.  
A second **Role Properties** dialog appears.
3. In the **Name** field, enter *Supervisor*. Note that the **Label** field automatically sets to the same value.
4. Leave everything else as is, and choose **OK**.
5. The first **Role Properties** dial appears.
6. Verify that the **Name** field says *Supervisor*, and then choose **OK**.  
The swimlane for the Supervisor role appears below the one for the Employee role.

After completing this task, your process design diagram should look like this:



**Figure 8: Supervisor role in Expense Report process**

Save your changes before proceeding to the next task.

## Adding a Global Creation Activity

To use a process, there has to be a way of creating a process instance that flows through it. One way to do this is with a *global creation activity*.

A global creation activity creates a process instance that then flows from the Begin activity. It is called a *global* activity because it does not run from within an instance. Rather, it is executed outside of, and independent from, any existing instances.

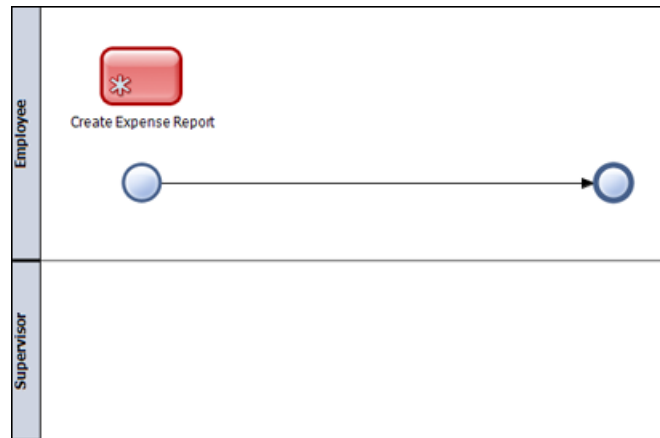
In the Expense Report process, we use just one global creation activity, which we name *Create Expense Report*.

To add the Create Expense Report global creation activity:

1. To the right of the process design editor, in the toolbox, expand **Global Activities** and select **Global Creation** (🔧). When you do this, do *not* hold the mouse button.  
The mouse cursor goes into insertion mode in the process design editor.
2. Insert the global creation activity in the Employee swimlane, above the Begin activity (see the image below), by clicking on the insertion point.  
The **Activity** dialog appears.
3. In the **Name** field, enter `Create Expense Report`. Note that the Activity ID, above the name field, is automatically completed to `CreateExpenseReport`. This is the name that will be used in code by developers.
4. Click **OK**.  
The *Create Expense Report* activity is added to the process.
5. Move the Begin and End activities down, as needed, if spacing is a bit tight in the diagram.

After completing this task, your process design diagram should look like this:





**Figure 9: Create Expense Report Global Creation Activity**

Save your changes before proceeding to the next task.

## Adding Participants

Interactive activities require *participants* to execute them. Participants are the people who log into the system and perform activities.

Participants can perform activities defined in the roles assigned to them. In other words, a participant who has been assigned the Employee role can perform activities present in the Employee role swimlane.

A participant with no roles can perform no activities. You will usually assign a role when you create a participant, so we also assign a role to the participant in this task.

Our first participant is called Peter Jones. Peter is in Sales and reports travel expenses. For the purposes of this process, we consider him an employee and assign him that role.

We call our second participant Paul Smith. We assign the Supervisor role to Paul.

You usually create participants in Studio for testing purposes. In an actual deployment, participants come from the company directory service or are imported from some other personnel data source. So, Peter Jones would most likely not be an actual employee even if you were building this process for real-world use.

To create participants Peter Jones and Paul Smith:

1. In the **Project Navigator**, expand the ExpenseManagement project, and then expand **Organization** (🏢).
2. Right-click on **Participants** (👤), and click **New** (🌟) in the context menu.  
The **Participant** dialog appears.
3. In the Participant ID field, enter *Peter Jones*, and click **OK**.  
A participant editor window opens for Peter Jones.
4. In the First Name and Last Name fields, enter Peter and Jones respectively.
5. In the **E-mail address** field, enter your e-mail address. This ensures that you receive any messages the process sends to the participant Peter Jones.
6. In the editor, go to the **Roles** section, and click **Add**.  
The **Roles** dialog box appears. It displays the Employee and Supervisor roles.
7. Select *Employee* from the **Roles** list, and click **OK**.  
The Employee role is added for Peter Jones.
8. Save the changes to Peter Jones by clicking on the **Save** icon (💾) or click **File ► Save** from the menu.
9. Close the editor window.

10. Follow steps 2 through 7 to create participant Paul Smith, assigning him to the Supervisor role.







After completing this task, your project should include the two new participants. Their names should appear in the Project Navigator under Participants.

## Running the Process

Although our process at this point is still limited in what it can do, you can nevertheless run it to get an early sense of the design and simulation cycle.

You interact with a process using Oracle BPM WorkSpace, a Web-based application for user interaction with business processes. Because WorkSpace is an end-user application, some of the wording in the UI differs from the terms used in Studio. The main difference you need to be aware of is that in WorkSpace instances are known as Work Items. We use both words interchangeably here.

To run the ExpenseReport process:

1. Before running the process, it is a good habit to save your project, so click **File ► Save All** (). If **Save All** is not enabled, it means all the files of your project are already saved, and you can skip this step.
2. To run the process, click the **Start Engine** icon () to start Studio's built-in process execution engine. The **Start Engine** dialog appears.
3. Check both the **Delete Process Instances** and **Delete Log Files** options, and click **OK**. The **Progress Information** box will appear during the startup of the process engine, which takes some time to complete. Once the engine has started, the box closes and, on the toolbar, the **Stop** button () replaces the **Run** button.
4. Click the **Launch WorkSpace** () button. The Oracle BPM WorkSpace login page appears in your default Web browser.
5. In the Username field, enter *Peter Jones*. The WorkSpace main page appears for user Peter Jones. Note that in the **Applications** panel there is a single entry called *Create Expense Report*. This executes the global creation activity we previously added to the Employee role. It appears on the list because Peter Jones has this role.
6. In the WorkSpace **Applications** panel, click *Create Expense Report*. An Expense Report instance is created. Since our process contains no activities, the instance goes directly to the End activity and it looks like nothing has happened.
7. To see the instance you have created, in the **Work Items** panel, click **Show Filters**. The **Filter by** section of the **Work Items** panel opens.
8. In the **Processes** pane, select *Expense Report* from the **Available Processes** list, and click the right arrow () to move it to the **Selected Processes** list.
9. In the **Work Items** pane, select the **Completed** option, and click **Apply Filter**. The **Work Items** panel displays the instance you created in step 6. Note it shows the Activity as *End*, and the Status as *Completed* ().
10. You can create more instances by clicking on *Create Expense Report* as in step 6. The instances appear automatically in the **Work Items** panel, because it is now showing completed instances.
11. Log out of WorkSpace by clicking **Logout** in the upper right of the page. The logout page appears.
12. Log in as Paul Smith, the supervisor by clicking **Re-Login** and then in the Username field, entering *Paul Smith*. The WorkSpace session for Paul Smith appears. Note that the *Create Expense Report* link is absent from the **Applications** panel. This is because Paul has the role of Supervisor, and there is no global activity in that swim lane. In fact there is no activity at all in the Supervisor swim lane, so Paul can't do anything at this point.

13. Log out of WorkSpace, close the browser window or tab where WorkSpace is, and go back to Studio.
14. Stop the Studio process execution engine by clicking the **Stop** button (■).  
The process execution engine stops, and the Run button replaces the Stop button.

After completing this task, you should begin to grasp how process design relates to process execution.

## Activity 1 Summary

Congratulations! You have completed the first activity of the tutorial, which goes all the way from creating the project to running a process.

In this activity you have created the following elements:

- The ExpenseManagement project
- The Expense Report process
- Two roles
- A global creation activity
- Two participants

You have also run the process execution engine and used WorkSpace to create instances and check their status.

We now have the main elements that allow us to go from design to execution as we flesh out the "skeleton" we have created.

## Activity 2: Building the Happy Path

In this activity, you add both automatic and interactive activities in the most expected path (also called the *happy path*) of the Expense Report process.


### Adding the Check Company Policy Activity

In this task you will add an automatic activity that will perform a simple check on the expense report using a simple rule.

Here, we add the *Check Company Policy* automatic activity, but we do not implement it yet, because in the happy path we assume that the report complies with company policy. We will add other possible paths later on.

This is an automatic activity, so it can go in any swimlane. We will insert the activity in the employee swimlane for convenience. The Begin activity is now in this swimlane.

To add the Check Company Policy automatic activity:

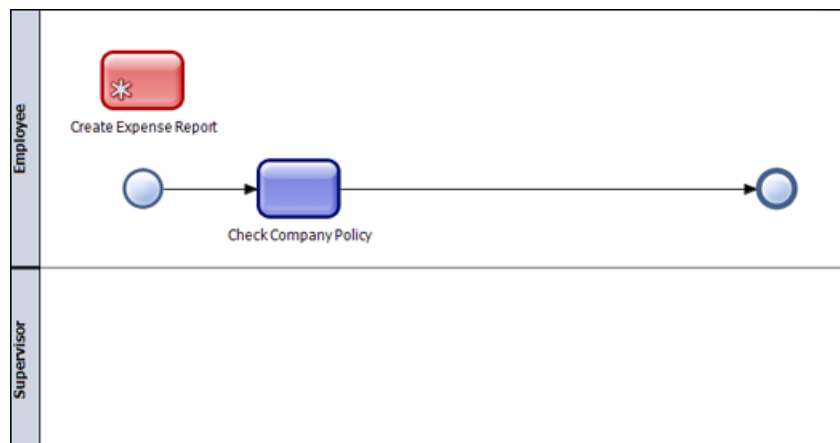
1. In the editor toolbar, click on the automatic activity icon (  ) and insert the automatic activity a bit to the right of the Begin activity. Note that as you move the activity around, the transition between the Begin and End activities is highlighted in purple. Place the activity right on the transition line *while it is highlighted*. The **Activity** dialog box appears.
2. In the **Name** field, enter *Check Company Policy*, and click **OK**.



**Note:** At this point Studio may ask you if you want to layout automatically the design as you add activities. If so, select the **Do not show this message again** option, and click **No**.

The Check Company Policy automatic activity is added to the process.

After completing this task, your process design diagram should look like this:



**Figure 10: Process with Check Company Policy activity.**

Save your changes before proceeding to the next task.

## Adding the Review Report Activity

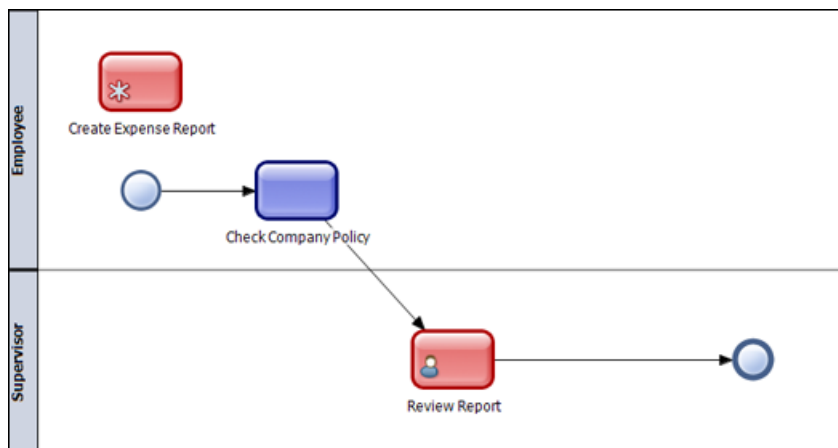
In this task you add the Review Report activity, an interactive activity used by participants in the Supervisor role.

To add the Review Report interactive activity:

1. Click on the interactive activity icon (👤) in the editor toolbar, and insert the interactive activity in the Supervisor swimlane, and to the right of the Check Company Policy activity.  
The **Activity** dialog box appears.
2. In the **Name** field, enter *Review Report*, and click **OK**.  
The Review Report interactive activity is added to the process.
3. Click on the transition between the Check Company Policy activity and the End activity.  
The transition is highlighted with a round grip at each end.
4. Click and drag the right side of the transition (the arrowhead side), so that it goes to the Review Report activity.  
The End activity is now disconnected, so if you save the process at this point some errors will appear in the **Problems** window, which is below the design editor.
5. To reconnect the End activity, click on **Add Transition** (➡), in the **Flow** section of the toolbar, then click on the Review Report activity and then finally on the End activity.  
The End activity is once again connected to the process. If there are errors in the Problems window, save the process. The errors should disappear within a few moments.
6. Move the End activity, by clicking and dragging, down to the Supervisor swimlane. It should be to the right of the Review Report activity (see the diagram below).

💡 **Note:** It is usually good practice to indicate forward progress in the process by flowing from left to right. Avoid backward flows in the happy path.

After completing this task, your process design diagram should look like this:



**Figure 11: Process with Review Report activity.**

Save your changes before proceeding to the next task.

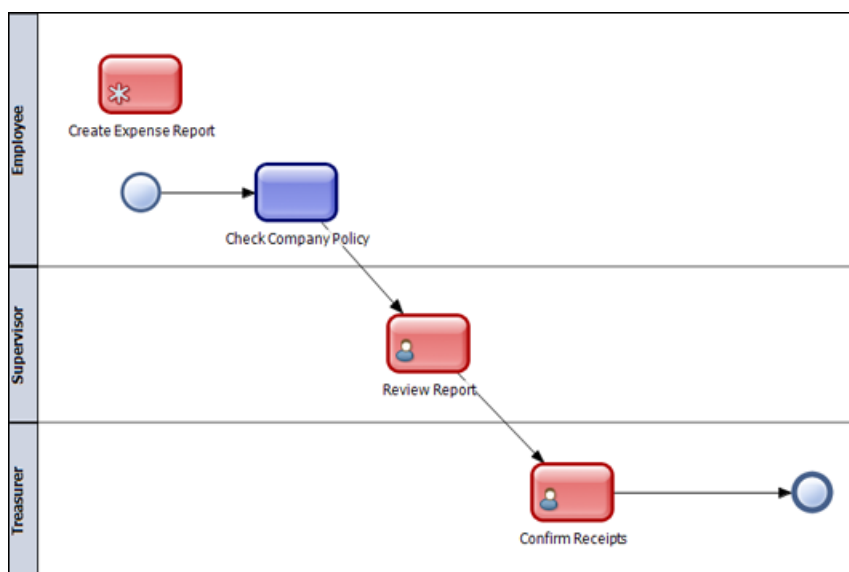
## Adding the Confirm Receipts Activity

In this task you will add the Confirm Receipts activity, an interactive activity that will be performed by the Treasurer role.

Since you have not yet created the Treasurer role, you will add it here "on the fly" as you add the Confirm Receipts activity.

1. Click the interactive activity icon (👤) in the editor toolbar, and insert the interactive activity *below* the Supervisor swimlane, right before the End activity  
The **Role Properties** dialog appears. Because you cannot insert an interactive activity outside a named swimlane, Studio needs to ask you for a role.
2. The **Name** drop-down list contains *Employee* and *Supervisor*, but the Treasurer role does not appear because you have not created it yet. To create the Treasurer role, click **New**.  
A second **Role Properties** dialog appears.
3. In the **Name** field, enter *Treasurer*, and click OK.  
The Treasurer role is added (but not yet on the diagram) and becomes the selected role in the **Role** dialog box.
4. In the Role Properties dialog, click OK.  
The **Activity** dialog appears.
5. In the **Name** field, enter *Confirm Receipts* and click **OK**.  
The Treasurer swimlane is added to the process diagram, and the Confirm Receipts interactive activity is added into it.
6. The Confirm Receipts activity is disconnected from the process, so you must connect it. Move the right (arrowhead) end of the last transition from the End to the Confirm Receipts activity, and add a new transition between the Confirm Receipts and the End activities, as you did in step five of the previous task.
7. Move the End activity to the Treasurer swimlane, keeping it to the right of the Confirm Receipts activity (see the diagram below).

After completing this task, your process design diagram should look like this:



**Figure 12: Expense Report process with Confirm Receipts Activity.**


Save your changes before proceeding to the next task.

## Adding the Process Expense Activity

The last activity in the happy path is the Process Expense activity. In this activity, the process can perform any actions required to fund or reimburse the reported expense.

Note that, since this is a tutorial project, it does not actually process any expenses!

To add the Process Expense automatic activity:

1. Click on the automatic activity icon (  ) in the editor toolbar, and insert it between the Confirm Receipts and End activities.  
The **Activity** dialog box appears.
2. In the **Name** field, enter `Process Expense` and click **OK**.  
The Process Expense automatic activity is added to the process diagram.
3. Save your changes.

After completing this task, the happy path is complete, and your process design diagram should look like this:

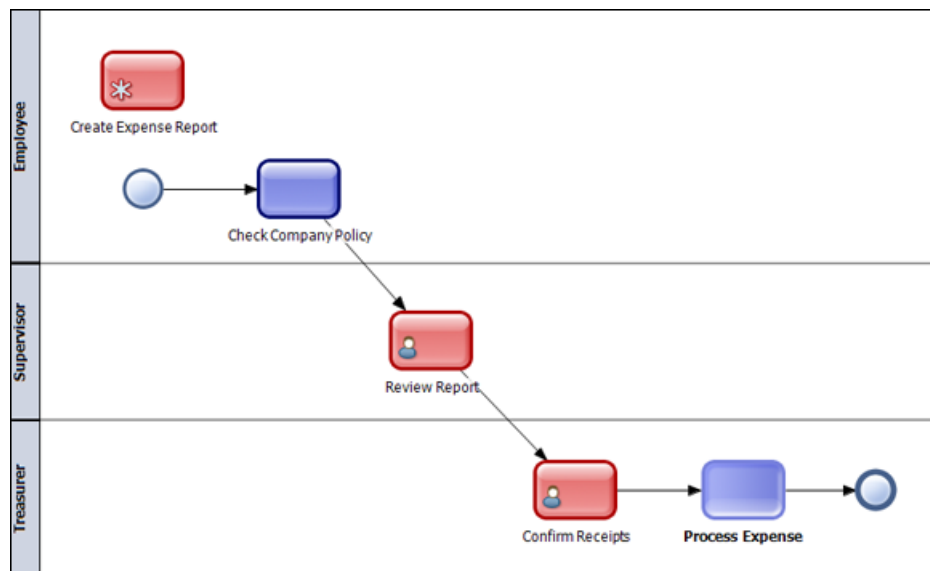




Figure 13: Expense Report process with complete most expected path.

## Running the Process as an Employee

You cannot yet input data into the process, but you can now run it to see how the Expense Report instance flows through the happy path.

To run the ExpenseReport process:

1. Click the **Start Engine** icon (  ) to start Studio's built-in process execution engine.  
The **Start Engine** dialog box appears.
2. Verify that both the **Delete Process Instances** and **Delete Log Files** options are selected, and click **OK**.
3. Once the engine has started, and the **Start Engine** icon has changed to a **Stop Engine** icon, click the **Launch Workspace** (  ) button.
4. In the Workspace login page, log in as `Peter Jones`.  
The Workspace main page appears for user Peter Jones.

5. In the WorkSpace **Applications** panel, click *Create Expense Report*.  
An ExpenseReport instance is created. The instance will go to the Check Company Policy automatic activity, and then on to the Review Report interactive activity, where it will wait for the supervisor to perform the activity.
6. To see the instance you have created, click **Show Filters** in the **Work Items** panel.  
The **Filter by** pane of the **Work Items** panel opens.
7. In the **Processes** pane, select *Expense Report* from the **Available Processes** list, and click the right arrow (➡) to move it to the **Selected Processes** list.
8. Set the **Assigned to** drop-down list to *Anyone, at any Role*, and click **Apply Filter**.  
The **Work Items** panel shows the instance you created in step 6. Note that it shows the Activity as *Review Report*, while the Status is blank. In other words, the Expense Report instance is now waiting for the Supervisor to perform the Review Report activity.
9. Log out of WorkSpace by clicking **Logout** in the upper right of the page.  
In the next task, you log in again, so do not exit the browser.

## Running the Process as Supervisor and Treasurer

In this case, you follow the process instance from the perspectives of the Supervisor and then the Treasurer. To run the process as Supervisor and Treasurer:

1. Let's go to the Supervisor's view of things. Log in to the WorkSpace as Paul Smith.  
The WorkSpace session for Paul Smith appears. Note that the Expense Report process instance you created as Peter Jones is listed in the **Work Items** panel, though you set no filter. This is because Paul, as supervisor, must now perform the Review Report activity. In effect, the Peter Jones participant did the equivalent of sending the "Expense Report" to Paul's inbox, and the inbox is the default view in the **Work Items** panel.
2. As Paul, approve the imaginary expense report by clicking **Send**.  
The process instance proceeds to the next activity, and disappears from Paul Smith's **Work Items** panel.
3. The next activity is Confirm Receipts. This is in the Treasurer role, so log out as Paul Smith.  
The re-login page appears.
4. You have defined the role of Treasurer, but have not yet created a participant with that role who can actually log in. You need to go back to Studio to do that, so switch to Studio but do not close the browser window.
5. You do not need to stop and restart the Studio process execution engine every time you make a change. Instead, you can simply reload a running project. So *do not* stop the process execution engine at this time. Instead, in the **Project Navigator**, expand **Organization**.
6. Add a participant called Mary White, and assign Mary the role of Treasurer. If you don't recall how to do this, follow steps 2 through 7 in [Adding Participants](#) on page 17.
7. Click the **Reload** button (🔄).  
Studio displays an "Operation in progress" message as the project is reloaded. Once it finishes reloading, Studio displays Mary White's name under **Participants** in the **Project Navigator**.
8. Go back to the browser, and log in to WorkSpace as Mary White  
The WorkSpace session for Mary appears. The Expense Report process instance you created as Peter Jones is now in Mary's inbox view, because Mary has the Treasurer role.
9. To see the steps the instance has been through so far, in the **Work Items** panel, click anywhere on the line of the Expense Report instance, except the checkbox or Actions column.  
The Expense Report instance will be highlighted in yellow, and information about it appears in the **Work Item Detail** panel.
10. In the **Work Item Detail** panel, click **Process Map**.  
A process diagram appears. The path the process instance has followed so far is highlighted in red.



11. Once you have finished looking at the diagram, close the window.
12. In the **Work Item Detail** panel, click the rightmost tab, **Audit Trail**.  
The same activities highlighted in the diagram appear listed in a table.
13. Acting as Mary, "confirm" the imaginary receipts by clicking **Send** at the work item line in the **Work Items** panel.  
The work item (instance) disappears from Mary's inbox and moves to the Process Expense activity.
14. Our treasurer Mary has completed her task, so log out, close the browser window or tab where WorkSpace is and, in Studio, stop the process engine.

## Activity 2 Summary

You have started to build the process, which now has all the activities in the happy path, and all the necessary roles.

In this activity you have created the following elements:

- The Check Company Policy automatic activity
- The Review Report interactive activity
- The Confirm Receipts interactive activity, along with the Treasurer role
- The Process Expense activity


You have again run the process execution engine and used WorkSpace to create instances, which you have followed as they flowed from activity to activity and from role to role.

We now have the basic process design for the happy path an Expense Report takes, but we have not yet created the report itself. We create the report in Activity 3 that follows.

# Activity 3: Defining the Expense Report Object

Just about any process works on information of some kind. In this activity, we define a BPM object to contain information about the expense report itself, including its status in the overall process.

Ordinarily, you may not handle the expense report data with a BPM object. You might, instead, use a pointer, perhaps an ID to access a record in a corporate database. Nevertheless, you will always handle at least some information within the BPM process. In this activity we do this by defining a BPM object.

 **Note:** To perform the tasks in this activity, Studio must be set to the Developer profile. If you want to learn only how to design the process diagram, you can skip this step and go on to Activity 4.

## Expense Report Data

To begin, examine the expense report form that follows:

Expense Report

012345

Submitted By:

Submit Date:

Description:

Cost Center:

Items

Description	Date	Amount	Receipt Checked

Total

Review

Reviewed By:

Comments:

Approved:

Yes

No

Figure 14: A paper expense report form

This is a simple form for submitting an expense report. In a paper process, the employee would submit this form along with the receipts for the items listed in it.

This form contains three sections:

- A header, with basic information about the submission
- The items section, with the "contents" of the form
- The footer (review section), which contains processing information not originally submitted




Both header and footer contain simple fixed information fields--such as names, dates, a yes/no option, and so forth--while the items section contains a variable number of entries. Since the items section is a bit more difficult to implement, we begin with the fixed fields. As a preparatory step, we list these fields along and the type of data each contains. Gathering this information now will help us later when we actually define the expense report object in Studio.

Field Name	Data Type
Submitted By	String
Submit Date	Date
Description	String
Cost Center	Integer
Total	Money
Reviewed By	String
Comments	String
Approved	Boolean

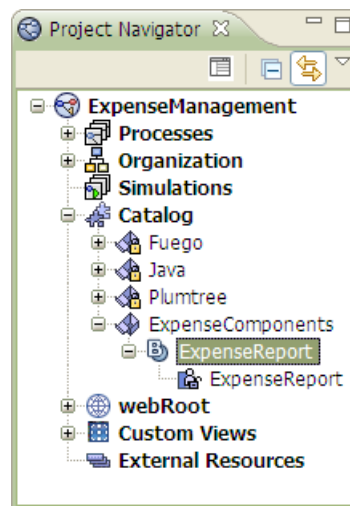
## Creating a BPM Object

In the Expense Management project, you store all the information about expense reports in a BPM object. You define this object in Oracle BPM Studio.

To create a BPM object:

1. In the **Project Navigator**, within the ExpenseManagement project, expand **Catalog** .  
You will see a list of catalog modules. If you are working on a fresh Studio installation, these will be *Fuego*, *Java*, and *Plumtree*.
2. You will add your own module, called ExpenseComponents. Right-click on Catalog and click **New ► Module** .  
The **Module** dialog box appears.
3. In the **Module** field, enter `ExpenseComponents`, and click **OK**.  
The ExpenseComponents module appears in the **Project Navigator** under **Catalog**.
4. Right-click on the ExpenseComponents module icon and click **New ► BPM Object** .  
The **BPM Object** dialog appears.
5. In the **Name** field, enter `ExpenseReport` and click **OK**.  
The ExpenseReport BPM object is added to the ExpenseComponents module.
6. Expand the ExpenseReport BPM object.  
You will see the contents of the ExpenseReport object. It contains one *method*, also called ExpenseReport. This is the *constructor* method, which means that it will execute whenever an Expense Report object is created (or "constructed"). If you need to include code that will initialize something in the BPM object, you can add it to this method.

Your project now contains a catalog module and a BPM object within it. In the **Project Navigator**, you should see the following:



**Figure 15: ExpenseReport BPM Object in ExpenseComponents module**

In the next task, you will add *attributes* to this object. These will contain the expense report data.

## Defining BPM Object Attributes

Once you have the BPM object, you need to specify the type of data to be stored in it. You do this by defining the attributes of the object. Each attribute holds one particular piece of information.

We begin with data fields that are used only once per expense report (once per instance), such as the name of the employee.

Note that the expense report form is numbered, but there is no "report number" in the table in [Activity 3: Defining the Expense Report Object](#) on page 26. We leave the implementation of this number to you, so you can test your skills once you have completed the tutorial.

We skip the *Total* attribute for now because it contains a calculated value. You will add it in a later task.

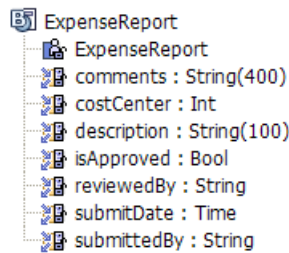
To define the attributes:

1. In the Project Navigator, right-click on the ExpenseReport BPM Object (B), and click **New ► Attribute** (E).  
The **Attribute** dialog box appears.
2. In the **Name** field, enter `submittedBy`.
3. Select *String* from the **Type** drop-down list, and click **OK**.  
The attribute editor for the `submittedBy` attribute opens.
4. In the **Storage Constraints** section of the editor, select **Not Null**. This tells Oracle BPM that the field is mandatory.
5. Save (F) the changes and close the editor window by clicking on the **X** in the *bottom* tab. If you close from the top tab, you close the BPM Object rather than just this attribute.
6. You now define the Submit Date attribute. Again, in the Project Navigator right-click on the ExpenseReport BPM Object (B), and click **New ► Attribute** (E).  
The **Attribute** dialog box appears.
7. Enter `submitDate` in the **Name** field.  
The attribute editor for the `submitDate` attribute opens.
8. From the **Type** drop-down list, select *Time*.  
The **Time Precision** list appears

9. From the **Time Precision** list, select *Date Only*.
10. As before, in the **Storage Constraints** section, check **Not Null**.
11. Save (💾) the changes and close the editor window by clicking on the **X** in the bottom tab.
12. By now you should be able to add attributes on your own. Add the remaining attributes according to the following table:

Name	Type	Additional Properties
description	String	Storage Constraints = Not Null Maximum Length = 100
costCenter	Int	<i>none</i>
reviewedBy	String	Storage Constraints = Not Null
comments	String	Storage Constraints = Not Null Maximum Length = 400
isApproved	Bool	Default Value = False

You have now defined seven attributes in the ExpenseReport BPM object. In the **Project Navigator**, you should see the following:



**Figure 16: ExpenseReport BPM Object Attributes**

In the next task, you add a *group* to this object. Groups are used to store arrays or collections of attribute sets.

## Defining a BPM Object Group

In a BPM object, you implement arrays or collections of attributes by using *groups*. A group can contain one or more attributes.

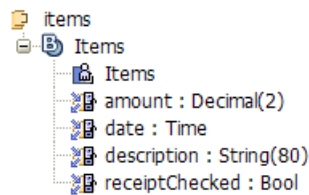
The ExpenseReport object requires only one group, for the item list. Each item has four attributes, listed in the following table:

Name	Type	Special Properties
description	String	Storage Constraints = Not Null Maximum Length = 80
date	Time	Time Precision = Date Only
amount	Decimal	Decimal Digits = 2
receiptChecked	Bool	Default Value = False

To define the group:

1. In the **Project Navigator**, right-click on the ExpenseReport BPM Object (B), and click **New ► Group** (G).  
The **Group** dialog box appears.
2. In the **Group Name** field, enter the word `items`, and click **OK**.  
The `items` group is created. It contains an object called `Items`, that in turn contains a method by the same name.
3. To create an attribute, right-click on the `Items` object (B), and click **New ► Attribute**.  
The **Attribute** dialog box appears.
4. Enter the word `description` in the **Name** field, and set the **Type** list to `String`.  
The attribute editor for the `description` attribute opens.
5. Set the **Maximum Length** to 80 and under **Storage Constraints** check **Not Null**.
6. Save and close the editor.
7. Following steps 3 to 5, define the remaining three attributes setting their properties as indicated by the table above.

You have now defined the `items` group. In the **Project Navigator**, you should see the following:



**Figure 17: ExpenseReport BPM Object Items Group**

In the next task, you add a *virtual* attribute. This returns the total value of the expense report.

## Defining a Virtual Attribute

A virtual attribute is not a stored data value like a regular attribute. Instead, when you access the value of a virtual attribute, code calculates the value to be returned.

You use a virtual attribute to implement the total amount of the expense report. In programming terms, you can think of a virtual attribute as a method or a pair of methods to either set or get a value.

To define a virtual attribute:

1. In the **Project Navigator**, right-click on the ExpenseReport BPM Object (B), and click **New ► Attribute** (B).  
The **Attribute** dialog box appears.
2. Enter the word `total` in the **Name** field.
3. Select `Decimal` from the **Type** drop-down list, and click **OK**.  
The `total` attribute is created, and the attribute editor for it opens.
4. Set the **Decimal Digits** to 2.
5. In the **Storage Constraints** section, set the **Virtual** option.
6. Click **Save** (S) and close the editor.  
The `total` attribute is now virtual.
7. In the **Project Navigator**, expand the `total` attribute.  
Two methods are shown: **Read Access** (B) and **Write Access** (B).
8. Double-click **Read Access**.

A PBL (Process Business Language) method editor window opens. The method is named `read_access_code_total`.

- The total attribute is the sum of every item's amount value. To have the `total` attribute return this value, it must be calculated by adding the amounts of all the items. Remove any existing code (a line with `return 0` should be all that you find), and enter the following PBL code into the editor exactly as shown, including capitalization:

```
amount as Decimal(2)

amount = 0

for each item in items do
    amount = amount + item.amount
end

return amount
```

- Save your changes and close the editor. You do not need to enter code for the Write method for this attribute.

You have added a virtual attribute to the ExpenseReport BPM object. In the following task you will define *valid values* for the `costCenter` attribute.

## Defining a Valid Values List

Often, a given piece of data may have a set of valid values much smaller than that allowed by the data type. For these cases, you can define a set of values that the user can select from.

For example, suppose you want a user to specify a state code for the US. There is a total of 676 two-letter combinations, but there are only 50 two-letter state codes. To assure data integrity, it is better if the user is allowed to choose one of only 50 states instead of entering any possible two-letter code.

When you configure a BPM object attribute with a set of valid values, the attribute is presented to the user as a list. The user must select one of the choices provided.

There are three possible **Valid Values** settings for a BPM object attribute:

Valid Values Setting	Description	Presented to User As
All	Any value within the bounds of the data type is accepted. This is the default setting.	Text field
Static List	One of a list of values is accepted. The list is set at design time.	Drop-down list
Dynamic Method	One of a list of values is accepted. The list is dynamically built by a method in run-time.	Drop-down list

The dynamic method is more flexible because you can pull the information from a database. However, this tutorial uses a static list because it is easy to configure without writing any code.

To add a static list of valid values to an attribute:

- In the **Project Navigator**, expand the ExpenseReport BPM Object (B).
- Double-click on the `costCenter` attribute.  
The editor for the `costCenter` attribute opens.
- In the **Valid Values** section, select **Static List**, and check the **Edit Value Descriptions** option.

A Value / Description table appears.

- To add each entry to the table, click **Add** (+). Then enter a numeric value in the *Value* column, and a text string in the *Description* column.

Add the values shown in the table below:

Value	Description
100	Sales
200	Marketing
500	Support

- Set the **Default Value** field to 100.
- Save your changes and close the editor for this attribute.

With this step, the data model for the BPM object is complete. In the following task you define a *process variable* to hold the BPM object as the instance travels through the process.

## Creating the Process Variable

You can store data about the instance in process variables. We will define a process variable of type ExpenseReport to store the expense report data in each instance.

Process variables can be of any type, such as a string or a decimal. A process variable does not have to be a BPM object. We use a BPM object here to keep things simple, so we only need to keep track of one variable and thus handle all of our information in one place. Depending on your particular situation, you may or may not find this to be the best strategy.

Note that while BPM object definitions are a part of the *project*, process variables are defined for each *process*. If you had another process where you wanted to use the ExpenseReport object, you would need to define a process variable in it.

To create the process variable:

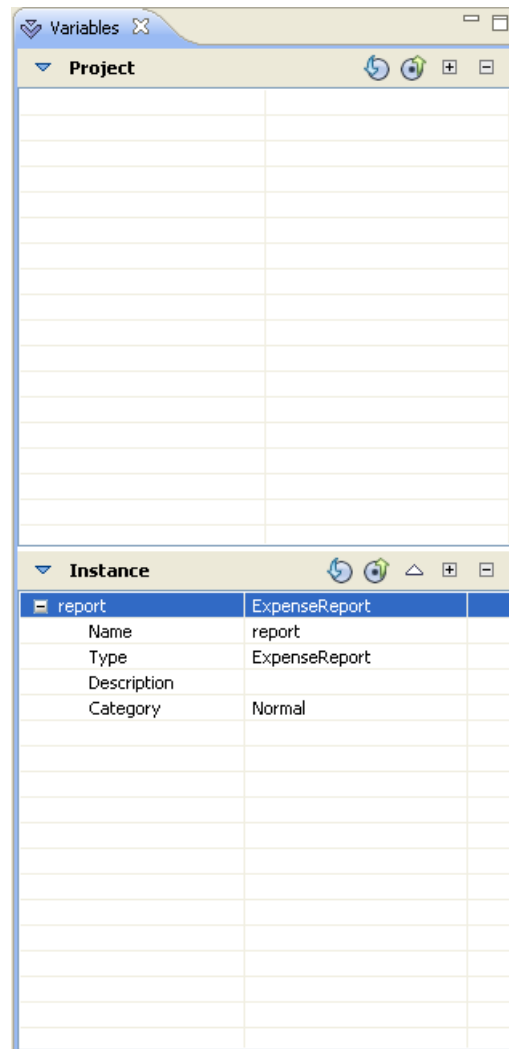
- In the **Project Navigator**, expand **Processes**, then select **Expense Report**.
  - Make sure the **Variables** window is visible. It is usually on the far right side of the screen. If it is not visible, click **Window ► Show View ► Variables** (🔍).
- The **Variables** window has two sections: **Project**, for project variables, and **Instance**, for process variables.
- In the **Instance** section of the **Variables** window, click the **Add** (+) icon .  
A variable entry is added to the process variable list. This variable has a default name and properties that you edit in the following steps.
  - Change the *Name* property to report.
  - Click on the *Type* property, and click the **Browse** button (...).  
The **Type** dialog box appears.
  - In the **Type** list, select <Component>.  
The **Component** section of the **Type** dialog appears.
  - Expand *ExpenseManagement*, then *Catalog*, and then *ExpenseComponents*.
  - Select *ExpenseReport*, and click **OK**.  
The variable type is changed to ExpenseReport.
  - Still in the **Variables** window, right-click on the report process variable, and click **Map as process incoming argument** (🔗).



The **Argument Mapping** dialog appears. An argument set called `BeginIn` has been defined with one argument called `reportArg`. The Create Expense Report global creation activity will pass an `ExpenseReport` object through this argument, and thus into the `report` process variable that is mapped to it.

10. You can map arguments manually in the **Argument Mapping** dialog. However, for the purposes of this tutorial, the only required mapping happened automatically by the **Map as process incoming argument** command you executed in step 8.
11. In the **Attribute Mapping** dialog, click **OK**, and save your changes.

After completing this task, the **Variables** window should look like this:



**Figure 18: Variables window with report process variable**



In the following task you will define a *presentation* to expose the BPM object to the users who will input the expense report.

## Creating a Presentation

In this tutorial, we have developed a simple process and a place to store the data traversing that process. When this data arrives in a participant's inbox, the participant must interact with it--updating it, perhaps, or simply reviewing it. To interact with the data, the participant needs an interface. In Oracle BPM, we develop such an interface either by creating a *presentation* or by using JSP pages.

Presentations can be built quickly by Studio, while JSP pages can have a look and feel precisely defined by developers. Even if the process you are developing will use JSP when deployed, presentations are useful during process development.

In this step, we create a presentation for the user who submits the expense report. This presentation will be based on the ExpenseReport BPM object. To create this presentation, follow these steps:

1. In the **Project Navigator**, right-click the ExpenseReport BPM object () and click **New ► Presentation** (). The **Presentation Wizard** appears.
2. Enter `SubmitReport` in the **Presentation** field.
3. Select the **From Template** checkbox. Do not select the **BPM Preferences** checkbox.
4. Click **Next**. The **Presentation Referenced Attributes** dialog appears. It enables you to select the attributes that will be shown as fields in the presentation.
5. Select the attributes shown in the table below, in the order shown.



<code>submittedBy</code>
<code>submitDate</code>
<code>description</code>
<code>costCenter</code>
<code>items[].description</code>
<code>items[].date</code>
<code>items[].amount</code>
<code>total</code>
<code>comments</code>

You control this order either by adding each attribute in the desired sequence, or by selecting an attribute in the list and using the **Up** and **Down** buttons to place it in the proper position.

The fields are placed in the presentation following the order of the attribute list.

6. Click **Finish**.

The `SubmitReport` presentation is created, and a presentation editor is opened with it.

The presentation editor includes the **Design Elements** toolbox. To dock the toolbox on the right side, click the **Dock Right** icon (). To dock it on the left side, click the **Dock Left** icon (.

Now that we have created the presentation, we must now specify how it should look and behave. The next step shows how to do this.

## Refining a Presentation

In the previous step in this tutorial, we created a presentation for our Expense Report. In this step, we refine that presentation, determining how it will look and behave.

Make sure the **Properties** window is open. It should be on the right side. If not, open it by clicking **Window Show View ► Properties** from the menu.

To refine the presentation of the Expense Report, follow these steps:

1. In the presentation editor, select the text field in the *Amount* column.

The properties for the text field appear in the **Properties** window. The *Name* property of the text field should be `items_text3`. If it is not, set it to `items_text3`.

2. Set the *On Change Invoke* property to `Refresh()`.  
This ensures that the *Total* value is recalculated every time an amount is entered or changed.
3. Since the user should not enter a value for the total, select the *Total* field, and, in the **Properties** window, set the *Editable* property to *No* simply by clicking on the property.
4. Save your changes.
5. You can preview what the presentation will look like. To do so, click on the **Preview in Browser** icon in upper right corner of the presentation editor.

An HTML preview of the presentation is displayed in your browser, and should look like this:

**SubmitReport**

SubmittedBy

SubmitDate

Description

CostCenter

Items

	Description	Date	Amount
1	<input type="text"/>	<input type="text"/>	<input type="text"/>

Total 0.00

Comments

6. Close the presentation editor.

Note that Studio created the *Description* and *Comments* fields as multi-line text boxes automatically, due to the string length. The *SubmitDate* and *Date* fields are accompanied by a calendar tool, while the *CostCenter* field is a list. Finally, the *Total*, which you set to not-editable (in other words, read-only), is shown as a label rather than a text field.

## Creating a Screenflow

You can design a process to call presentations directly, but normally you call them from a *screenflow* that is, in turn called from the business process. A screenflow is a special type of process that defines user interaction sequences.



You can create a screenflow in different ways. Here we create the screenflow directly from the activity that calls it.

To create a screenflow:

1. In the **Project Navigator**, expand **Processes**, then click Expense Report.  
The process design editor displays the notation for the Expense Report
2. In the process design editor, right-click on the Create Expense Report activity, and click **Main Task** from the context menu.  
The **Main Task** dialog box appears.
3. From the **Implementation Type** list, select *Screenflow* from the drop-down list.
4. In the **Related Screenflow** section, click **New**.  
The **Screenflow Information** dialog appears.
5. In the **Name** field, enter Submit Report, and click **Next**.  
The **Select Instance Variables** dialog appears.
6. Set the reportArg variable as follows:

In	Out
No	No

Leave all others as *No*.

7. Click **Next**.  
The **Screenflow out - Process in Argument Mapping** dialog appears.
8. Verify that this page has no entries, and click **Next**.  
The message "Screenflow created successfully" appears.
9. Click **Finish**.  
The **Screenflow out - Process in Argument Mapping** dialog closes and you are prompted to specify the location of the screenflow in the **Project Navigator**
10. Click **OK** to accept the default location shown, under **Processes**.
11. Click **OK** in the **Main Task** dialog box.  
The dialog box closes and an editor opens with the Submit Report screenflow diagram. The Begin and End activities are automatically displayed, just as when a new process is created.
12. In the **Project Navigator**, expand **Processes** and click Expense Report. The process notation appears in the design editor. In the **Variables** window, in the **Instance** section, add a process variable named reportSf, of type ExpenseReport. You can do this by following steps 2 to 7 of the [Creating the Process Variable](#) on page 32 activity.  
  
This process variable holds an ExpenseReport object during the life of the screenflow. We added the Sf suffix to make it easier to identify as a screenflow process variable. This suffix is not an Oracle BPM convention and is not required.
13. Right-click on the reportSf process variable and click **Map as process incoming argument** .  
The **Argument Mapping** dialog appears. It shows that, in the *Begin* activity, the reportSf instance variable receives the value of the *incoming* argument reportSfArg (hence this is the *BeginIn* mapping).
14. Click **OK** in the **Argument Mapping** dialog box, and then right-click again on the reportSf process variable and click **Map as process outgoing argument** .  
The **Argument Mapping** dialog box appears again. This time it shows that at the *End* activity, the *outgoing* argument reportSfArg is assigned the value of the reportSf screenflow process variable (hence this is the *EndOut* mapping).
15. Click **OK** to close the **Argument Mapping** dialog, and save your changes.

You have now created the Submit Report screenflow, and set this screenflow as the main task of the Create Expense Report activity. You have defined the process variable for the screenflow (reportSf), and you have

mapped the inbound and outbound arguments of the screenflow to it. You can check this at any time by right-clicking on the Begin or End activities and clicking on **Argument Mapping**.



You still need to design the screenflow, and will do this in the next task.

## Designing a Screenflow

You design a screenflow much as you do a process, by adding activities and defining their properties.


Although screenflows are similar to processes, there are important differences. Fewer types of activities can be used in a screenflow, and some of those activities—for example, the Interactive Component Call activity—are unique to screenflows. Also, there are no roles in a screenflow, since the role using it is that of the calling activity.

To design the Submit Report screenflow:

1. Add an Interactive Component Call () to the screenflow diagram, between the Begin and End activities. The **Activity** dialog appears.
2. In the **Name** field, enter *Input Report*, and click **OK**. Click **No** if the automatic layout design question appears.  
The *Input Report* interactive component call activity is added to the diagram.
3. Right-click on the Input Report activity icon and click **Main Task**.  
The **Main Task** dialog appears.
4. In the **Implementation Type** list, select *BPM Object Interactive Call*.
5. In the **Select BPM Object Variable** list, select *reportSf*.
6. Select the **Use BPM Object Presentation** option, and select *SubmitReport* from the drop-down list.  
This is the presentation you created in the Creating a Presentation task.
7. Select the **Input** option.
8. You need to specify the output arguments, so click **Argument Mappings**.  
The **Argument Mapping** dialog appears.
9. On the left side of the dialog box, you can see that there are two argument sets: *InputReportIn* and *InputReportOut*. We will leave *InputReportIn* empty. From the **Argument Set Names** list, select *InputReportOut*.
10. Enter the following two values to the table. Click the **Add** icon () to add each Process variable:

Instance Variable	Argument
action	selectedButton == "submit" ? OK : CANCEL
result	selectedButton

Both *action* and *result* are predefined process variables. The argument for the action variable is a conditional statement (like an if-then-else) that will cancel the activity if the user clicks any button except the submit button to exit the submission form.

11. Click **OK**, then click **OK** again in the **Main Task** dialog box.  
The interactive component call is added and mapped.
12. Save your changes with **Save All** (.

Your screenflow should now look like this:

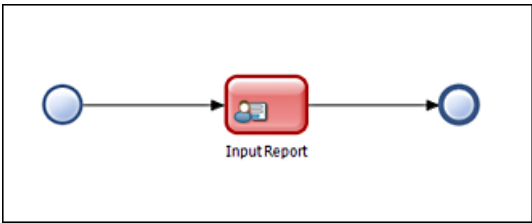


Figure 19: Input Report Screenflow

- 13. Close the screenflow editor.
- 14. Now you need to specify the argument mapping to the screenflow from the Create Expense Report activity. In the process design editor, right-click on this activity and click **Main Task** . The **Main Task** dialog box opens.
- 15. From the **Implementation Type** list, select Screenflow, and from the **Related Screenflow** list, select Submit Report.
- 16. Click **Argument Mapping**. The **Argument Mapping** dialog appears.
- 17. In the left column, select *Submit Report*. In the right pane, in the **Value** column, type `ExpenseReport ( )`. Note that `ExpenseReport()` does not appear in the list. You must type it.

At this point, the Argument Mapping dialog shows `reportSfArg` as the incoming argument to the Begin activity of the screenflow, while `ExpenseReport ( )` is the constructor of the `ExpenseReport` object.

We use a constructor explicitly because the Submit Report screenflow has incoming as well as outgoing arguments. You would not need this if the screenflow had no incoming argument. We designed the screenflow with an incoming argument so we can use it again in the Edit Report activity.

- 18. Add an entry to the *Submit Report Out* page exactly as follows:

Expense Report's input arguments	Submit Report's output arguments
reportArg	= ReportSfArg

Here, `reportArg` is the incoming argument to the Expense Report process Begin activity, while `ReportSfArg` is outgoing argument from the screenflow.

- 19. Click **OK** and then click **OK** again in the **Main Task** dialog box.

With the screenflow, all the components are now in place so that users with the Employee role can submit the expense report. You will test this in the next and final task of this activity.

## Running the Process

You can now input data into the process in the Employee role.

To run the `ExpenseReport` process:

- 1. From the menu, click **Save All** (💾). This ensures that all of your project elements are saved.
- 2. Click the **Start Engine** icon (▶) to start the Studio built-in process engine. The **Start Engine** dialog box appears.
- 3. Check both the **Delete Process Instances** and **Delete Log Files** options, and click **OK**.
- 4. Click the **Launch Workspace** (🌐) button.
- 5. In the Workspace login page, log in as Peter Jones.

6. Click on *Create Expense Report* in the WorkSpace **Applications** panel.  
An ExpenseReport instance is created, and a window opens with the Expense Report form.
7. Fill out the form with sample information. In the SubmittedBy field, enter `Peter Jones` exactly as you entered the login name. To add an item in the Items section, click the icon with the plus sign (+).
8. Log out of WorkSpace by clicking **Logout** in the upper right of the page.
9. In Studio, stop the process engine by clicking on the **Stop Engine** button (■).

You submitted an expense report as Peter Jones. For the other participants, we still need to add screenflows to the Review Report and Check Receipts activities, so this is all we will do in the WorkSpace for now.

## Activity 3 Summary

You have accomplished a great deal in this activity. Your process can now handle expense report data and has an interface to enter it, yet you have written almost no code.

In this activity you have created the following elements:

- The ExpenseReport BPM object
- Attributes for the ExpenseReport object
- A group for the ExpenseReport object
- A presentation, so the Employee role can submit the expense report
- A screenflow, so the process can call the presentation

You have tested the presentation and by extension the screenflow and the underlying BPM object. If everything worked correctly, you are all set to complete the process in Activity 4.

## Activity 4: Adding Alternative Paths

---

You now have created a good part of the functionality of the most expected path of the project. It is time to create alternative paths. These paths are also expected, but they are expected to be less frequent than the happy path.

Alternative paths can be more, equally, or less desirable than the most expected path. In this activity we will adopt the convention of adding more desirable, or "happier", paths *above* the most expected path, and less desirable or "less happy" paths *below* the most expected path.

### Defining the Check Company Policy Task

The first activity after the instance is created is Check Company Policy. This is an automatic activity in which you code a simple rule to handle pre-defined approval and rejection criteria.

The reason to implement company policy in an automatic activity is to lighten the workload on supervisors, so they don't spend time on trivial cases.

The hypothetical company rules are: Any expense report totaling less than \$2,500 is accepted into the approval process. Any expense report of less than \$25 is approved automatically and goes directly to the Treasurer. If the total is \$2,500 or more, the expense report is rejected.

To implement the company expense rules:

1. Right-click on the Check Company Policy automatic activity, and click **Main Task**.  
The **Main Task** dialog box appears.
2. In the **Method** section, click **New**.  
The **New Process Method** dialog appears.
3. You can accept the default suggestion for the **Method Name** field, which is `checkCompanyPolicy`, so click **OK**.
4. Back in the **Main Task** dialog box, click **Edit**.  
A PBL method editor page opens in the ExpenseReport editor.
5. In the editor, enter the following PBL code exactly as shown:

```
if report.total < 2500.0 then
  result = "Accepted"
  if report.total < 25.0 then
    report.isApproved = true
  end
else
  result = "Rejected"
end
```

This code implements the rule described above. Note that you never declared the `result` variable. This is a predefined process variable of type `String`, meant to be used to store a value indicating the result status of a given activity.

6. Save your changes and then close the editor page (from the bottom tab).

You have implemented a PBL method that sets two parameters (the predefined variable `result` and the `ExpenseReport` object attribute `isApproved`). These parameters will be used by conditional transitions to control the process flow itself is controlled by conditional transitions, as you will see starting with the next task.

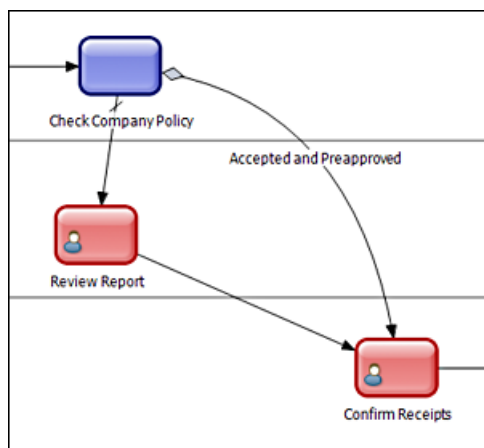


## Adding a Condition Transition

The instance is routed through a *condition* transition when a given condition is met. Here we will add a condition transition so that the report skips the approval activity if the total amount is low.

To add a condition transition:

1. In the **Project Navigator**, expand **Processes** and click Expense Report.  
in the right pane, the process design editor displays the notations for the Expense Report process.
2. Right-click on the Check Company Policy automatic activity, and click **Add conditional transition** (🔗).  
Your mouse cursor will begin dragging a transition line.
3. Click on the Confirm Receipts interactive activity.  
The **Transition from Activity** dialog box appears.
4. Enter *Accepted* and *Preapproved* in the **Name** field, and click **OK**.  
The transition is added to the diagram--however it is hidden behind another transition and therefore hard to see.
5. We will make this transition a curve. To be able to select it, first drag the Review Report activity left (we will move it back later).  
You should now be able to see the *Accepted* and *Preapproved* transition.
6. Click on the middle of the transition and drag towards the upper right. Note how it becomes an arc. At this point, you should see something like the following in your process diagram.



**Figure 20: Accepted and Preapproved Transition**

7. Drag the Review Report activity back to its original location.
8. Right-click on the Accepted and Preapproved transition.  
The **Transition from Activity** dialog box appears.
9. Select the **Properties** tab.  
Note that in the **Type** drop-down list, **Condition** is selected, because this is a condition transition.
10. In the text box for the condition transition expression, enter the following, exactly as shown:  

```
result == "Accepted" and report.isApproved == true
```

This expression checks for the values set in the Check Company Policy activity that you worked on in the previous task.
11. Click **OK**.  
The condition expression is set.
12. Save your changes.

After completing this task, the process design diagram should look like this:

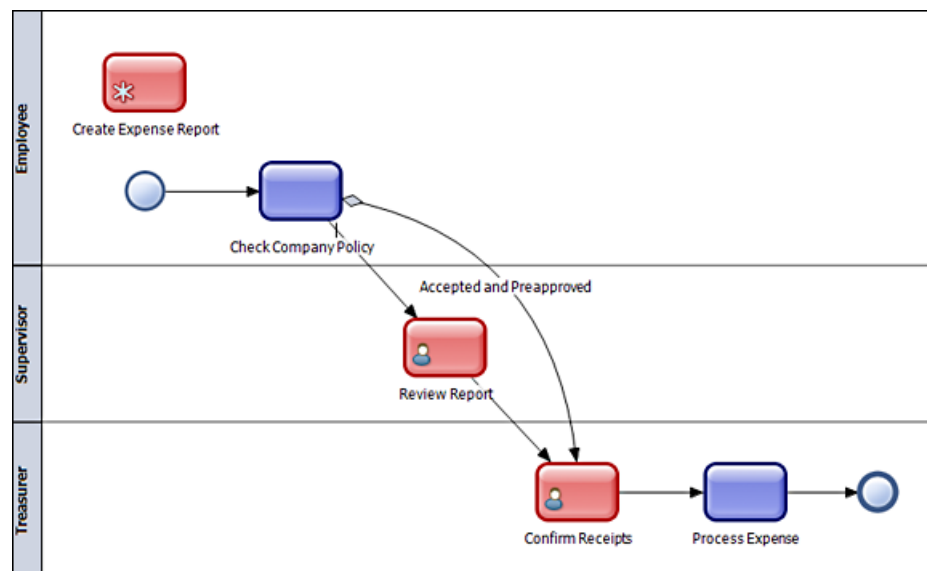


Figure 21: Expense Report process with Accepted and Preapproved transition.

## Adding the Edit Report Activity

If an Expense Report is rejected either automatically or by another participant, the employee must be able to correct mistakes or add clarifying information. To enable the employee to make such edits, we add the Edit Report activity.

The Edit Report Activity is an interactive activity in the Employee role. It presents the same Expense Report form as the Create Expense Report activity, but in this case the employee must modify existing data rather than fill out a blank form. Therefore, the screenflow must also receive the data, and argument mapping is required in both directions, as evident in steps 8 and 9.

To add the Edit Report interactive activity:

1. In the **Project Navigator**, expand **Processes** and click Expense Report.  
in the right pane, the process design editor displays the notations for the Expense Report process.
2. In the process design editor, place the cursor over the horizontal line separating the Employee and Supervisor roles, and drag this line down to make more room in the Employee swimlane.
3. Right-click in the Employee swimlane, somewhere below the Begin activity, and click **Add activity Interactive** (👤).  
The **Activity** dialog appears.
4. In the Name field, enter Edit Report, and click **OK**.  
The Edit Report interactive activity is added to the diagram.
5. Right-click on the Edit Report activity and click **Add unconditional transition**.
6. Click the Check Company Policy automatic activity.  
The unconditional transition is added going from the Edit Report interactive activity to the Check Company Policy automatic activity.
7. You have added the Edit Report activity, and must specify the task it will execute. Right-click on the Edit Report activity and click **Main Task**.  
The **Main Task** dialog appears.
8. Set the **Implementation Type** to *Screenflow*.

9. You will use an existing screenflow. In the **Related Screenflow** section, choose *Submit Report* from the **Name** drop-down list.
- Rather than create a new screenflow, you can re-use an existing one. This is because the Employee participant must edit the same information the participant entered originally. No new fields are required.

10. Click **Argument Mapping**, click the Add icon (+), and set the *Submit Report In* page to the following:

Submit Report's input arguments	ExpenseReport's instance variables
reportSfArg	= report

11. Set the *Submit Report Out* page to the following:

Expense Report's instance variables	Submit Report's output arguments
report	= reportSfArg

12. Click **OK**, and then click **OK** again in the **Main Task** dialog box.
- The Edit Report activity is configured with the Submit Report screenflow.
13. Save your changes.

After completing this task, your process design diagram should look like this:

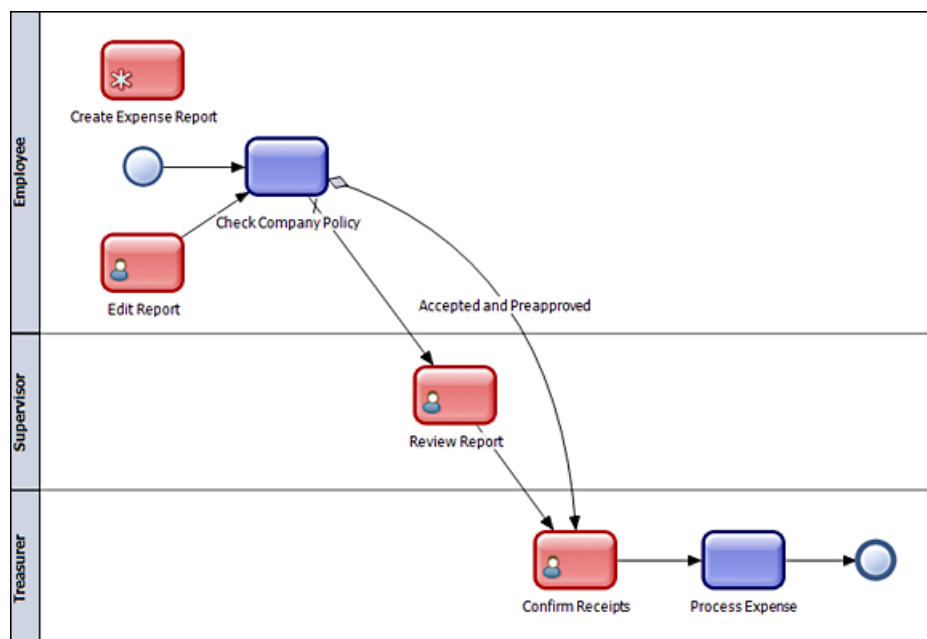


Figure 22: Expense Report process with Edit Report activity added.

## Adding Condition Transitions to Edit Report

You now have the Edit Report activity, but there is no path to take the instance to it. In this task we add two condition transitions: one for when the expense report is not accepted at the Check Company Policy activity, and the other for when it is not approved at the Review Report activity.

To add the two condition transitions:

1. Right-click on the Check Company Policy automatic activity, and click **Add conditional transition** (⚙️).
2. click on the Edit Report activity.

The **Transition from Activity** dialog box appears.

- 3. In the **Name** field, enter Not Accepted.
- 4. In the **Properties** tab page, enter the following in the condition expression text box, capitalizing as shown:

result != "Accepted"
- 5. Click **OK**.  
The Not Accepted condition transition is added.
- 6. Drag the Not Accepted condition transition so it becomes curved, as shown in the figure below. To be able to select the transition, you may need to drag the unconditional transition out of the way first.  
To straighten any curved transition, right-click on the transition and uncheck **Curve**.
- 7. Follow steps 1 through 5 to add a condition transition as follows:

From	To	Name	Expression
Review Report	Edit Report	Not Approved	report.isApproved != true

After completing this task, your process design diagram should look like this:

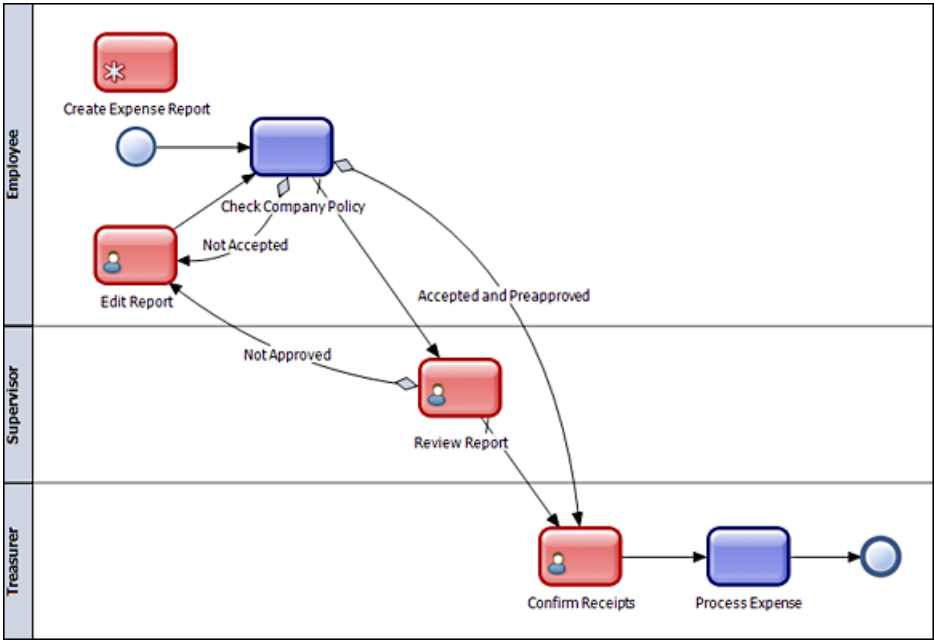




Figure 23: Expense Report process with additional condition transitions

Defining the Review Report Presentation

We now have a condition transition that the process instance flows through if the supervisor does not approve the expense report. We must now define the Review Report activity in which the Supervisor can approve or reject the report.


To implement the Review Report activity, you first need to create the Review Report presentation. The Review Report Presentation is similar to the Submit Report presentation, except that it also allows the supervisor to accept or reject the expense report.

To create the Review Report Presentation:

1. In the Project Navigator, right-click on the ExpenseReport BPM Object () , and click **New Presentation** () .  
The **Presentation Wizard** appears.
2. In the **Presentation** field , enter ReviewReport.
3. The ReviewReport presentation is based on the ExpenseReport BPM object. To select this, check the **From Template** checkbox. Do not check the **BPM Preferences** checkbox.
4. Click **Next**.  
The **Presentation Referenced Attributes** page of the **Presentation Wizard** appears.
5. In this dialog box you select the attributes that display as fields in the presentation. Select the attributes shown in the table below, in the order shown.

submittedBy
submitDate
description
costCenter
items[].description
items[].date
items[].amount
total
comments
reviewedBy
isApproved

Note that these are the same attributes you specified in the Submit Report presentation, plus the reviewedBy and isApproved attributes.

6. Click **Finish**.  
The ReviewReport presentation is created, and a presentation editor is opened with it.
7. Make sure the **Properties** window is open.
8. In the presentation, select the input text box in the *SubmittedBy* row, and, in the **Properties** window, set the *Editable* property to *No* by clicking on the property. The name of this field should be text0. If it is not text0, then change it to text0.  
We set this so that the users of this presentation cannot change the name of the submitting employee.
9. Select the text field in the *Amount* column.  
The *Name* property of the text field should be items\_text3. If it is not, then, in the **Properties** window, click the property and change the value to items\_text3.
10. Set the *Editable* property to *No*.  
We set this so that the users of this presentation cannot change expense amounts.
11. Also select the text field for the Total value and, in the **Properties** window, set the *Editable* property to *No*.
12. Save your changes.
13. To verify, click on the Preview in Browser () icon in the presentation editor.

The ReviewReport presentation should look like this:

# ReviewReport

SubmittedBy

SubmitDate

9/16/2008

Description

CostCenter

Sales

Items	Description	Date	Amount
1	<div></div>	<div></div>	

Total

0.00

Comments

ReviewedBy

IsApproved

☐

Submit

Cancel

Figure 24: Review Report presentation

14. Once you are satisfied that the presentation is correct, close the presentation editor and save if necessary.

## Defining the Review Report Activity

You have a conditional transition to be used in case the supervisor does not approve the expense report, and you have the presentation that the supervisor is to use. Here we implement the Review Report activity as a screenflow like the one we created for the Submit Report activity.

In this case, the report process variable is already defined, so the process is simpler.

To define the Review Report activity:

1. Right-click on the Review Report activity, and click **Main Task** .  
The **Main Task** dialog box appears.
2. In the **Implementation Type** section select *Screenflow* from the drop-down list.  
The **Related Screenflow** section appears.
3. Click **New**.  
The **Screenflow Information** dialog appears.
4. In the **Name** field, enter Review Report, and click **Next**.  
The **Select Instance Variables** page of the wizard appears.

5. The `report` variable should be set as follows:

In	Out
Yes	Yes

All others should be set to *No*.

6. Click **Next**.  
The wizard finishes with a "Screenflow created successfully" message.
7. Click **Finish**, and, when prompted for a location for the new screenflow, click **OK** to accept the default .
8. Back in the **Main Task** dialog, click **Argument Mapping**.  
The **Argument Mapping** dialog appears.
9. Verify that the *Review Report In* page shows the following:

Review Report's input arguments	ExpenseReport's instance variables
reportArg	= report

Here, `reportArg` is the incoming argument to the screenflow's Begin activity, while `report` is the process instance variable that will be passed to it.

10. Verify that the *Review Report Out* page shows the following:

ExpenseReport's instance variables	Review Report's output arguments
report	= reportArg

Again, `report` is the process instance variable, while `reportArg` is now the screenflow's *outgoing* argument to that will be returned to the calling activity.


11. Click **OK**, then **OK** again in the **Main Task** dialog box.  
A design editor opens for the Review Report screenflow.

You have created the Review Report screenflow, and set this screenflow as the main task of the Review Report activity. You will design the screenflow in the next task.

## Designing the Review Report Screenflow

Here you design the Review Report screenflow to complete the functionality of the Review Report activity.

To design the Review Report screenflow:

1. Add an interactive component call () to the screenflow diagram, between the Begin and End activities.  
The **Activity** dialog box appears.
2. In the **Name** field, enter `Review Report`, and click **OK**.  
The activity is added to the diagram.
3. Right-click on the Review Report activity icon and click **Main Task**.  
The **Main Task** dialog box appears.
4. In the **Implementation Type** drop-down list, select *BPM Object Interactive Call*.
5. In the **Select BPM Object Variable** drop-down list, select *report*.
6. Select the **Use BPM Object Presentation** option, and select *ReviewReport* from the drop-down list.
7. Select the **Input** option.
8. You need to specify the output arguments, so Click **Argument Mappings**.  
The **Argument Mapping** dialog appears.

9. From the **Argument Set Names** list, select *ReviewReportOut*.
10. Click the Add icon (+) to add an Instance variable.
11. Enter the following two values to the table:

Instance Variable	Argument
action	selectedButton == "submit" ? OK : CANCEL
result	selectedButton

12. Click **OK**, then, in the **Main Task** dialog, click **OK** again.  
Your screenflow should now look like the following:



**Figure 25: Review Report screenflow**

13. Save and close the screenflow design editor.

With the Review Report screenflow, all the components of the Review Report activity are complete.

## Adding the Not Confirmed Transition

In this task you add the last conditional transition of the Expense Report process. This transition is to be used when the treasurer does not confirm one or more receipts.

Because we don't know the number of items in an expense report beforehand, we need to code a loop to check the status of each receipt.

You could place this loop in the conditional transition itself, but it is better to add a method to the BPM Object. In this way, you can use the method elsewhere, and, if you make any changes to the way this information is stored in the BPM object, you need to update only one piece of code.

Also, the conditional expression in the transition itself is simpler, and thus easier to read and maintain. It is good practice to use simple expressions within conditional transitions.

You will define the `areReceiptsChecked` method and then add the *Not Confirmed* transition, which uses it.

To add the `areReceiptsChecked` method and the Not Confirmed transition:

1. In the **Project Navigator**, right-click on the ExpenseReport BPM Object, and click **New Method**. The **Method** dialog box appears.
2. In the **Method Name** field, enter `areReceiptsChecked` and click **OK**. A method editor opens.
3. In the **Properties Window**, set the *Return Type* property to *Bool*.
4. Enter the following code into the editor, exactly as shown:

```

for each item in items do
  if not item.receiptChecked then
    return false
  
```



```

    end
end
return true


```

```

for each item in items do
if not item.receiptChecked then
return false
end
end
return true

```

This code returns `false` if any item has not been checked. If all items have been checked, it returns `true`.

5. Save and close the editor.
6. In the process design editor, right-click on the Confirm Receipts activity and click **Add conditional transition** .

The mouse cursor begins dragging a transition line originating from the Confirm Receipts activity.

7. Click on the Edit Report activity.  
The **Transition from Activity** dialog appears.
8. Enter `Not Confirmed` in the **Name** field.
9. Go to the Properties page of the dialog box. In the conditional expression text box, enter the following expression:

```
not report.areReceiptsChecked( )
```

This expression will return `true` if `areReceiptsChecked` returns `false`. In other words, the instance flows through the condition transition if the supervisor does not check receipts for one or more items.

10. Click **OK**.  
The Not Confirmed conditional transition is added.
11. Click and drag the Not Confirmed transition into an arc, then save your changes.

After completing this task, your process design diagram looks like this:

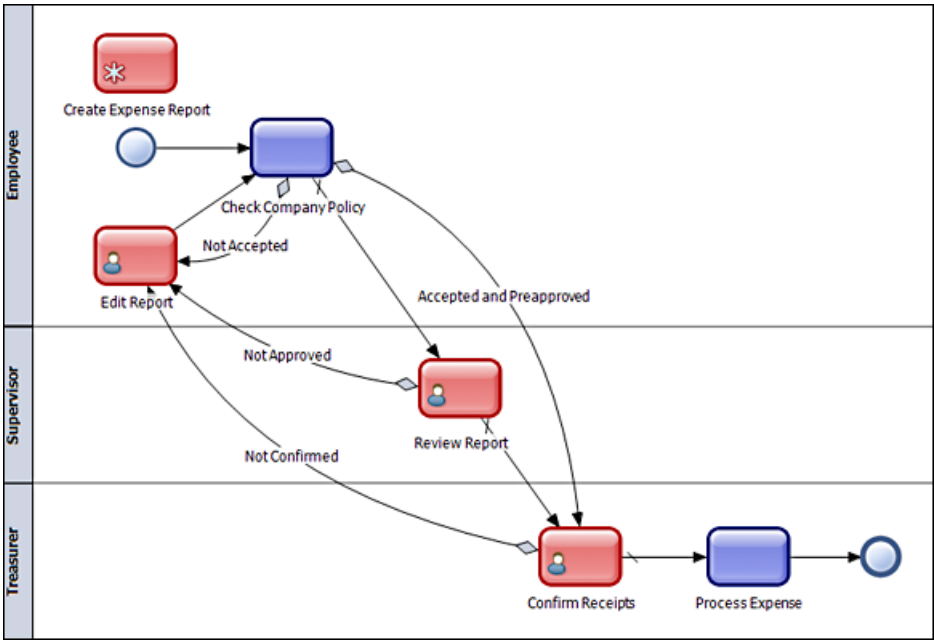


Figure 26: Expense Report process with Not Confirmed conditional transition

### Creating the Confirm Receipts Presentation

You must implement a task for the Confirm Receipts activity. As with the other interactive activities, the Confirm Receipts activity uses a screenflow. As before, you must first create the presentation this screenflow uses.

The Confirm Receipts presentation is similar to the Review Report presentation, but it also enables the treasurer to check off each item that has the required receipts.

To create the Confirm Receipts presentation:

1. In the Project Navigator, right-click on the ExpenseReport BPM Object (B), and click **New Presentation** (P).  
The **Presentation Wizard** appears.
2. Enter `ConfirmReceipts` in the **Presentation** field.
3. Check the **From Template** checkbox. Do not select the **BPM Preferences** checkbox.
4. Click **Next**.  
The **Presentation Referenced Attributes** page of the **Presentation Wizard** appears.
5. In this dialog box you select the attributes which will be shown as fields in the presentation. Select the attributes in the table below, in the order shown.


submittedBy
submitDate
description
costCenter
items[].description
items[].date
items[].amount

```

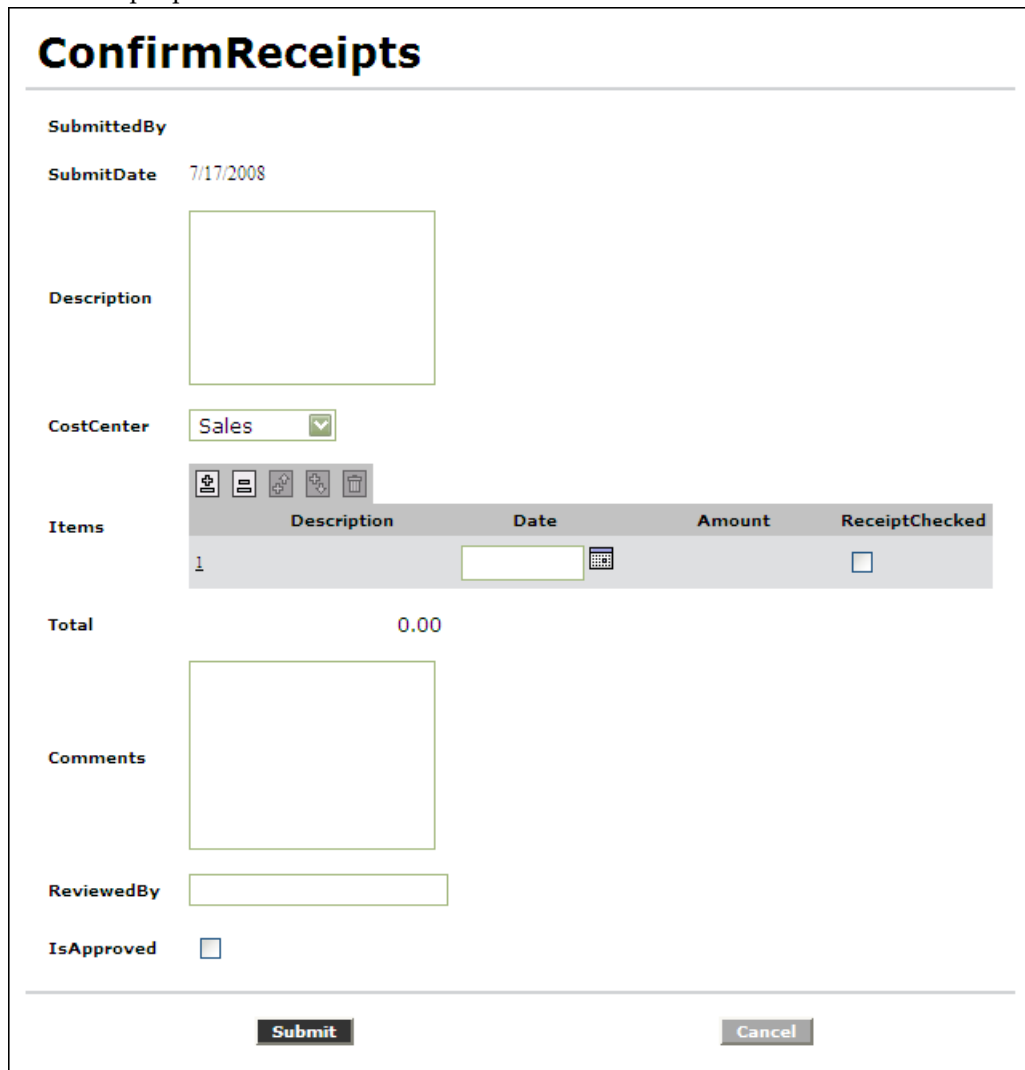
items[].receiptChecked
total
comments
reviewedBy
isApproved

```

In this case, we use all of the attributes of the ExpenseReport object.

6. Click **Finish**.  
The ConfirmReceipts presentation is created, and a presentation editor is opens.
7. Using the **Properties** window, in the fields for the submittedBy, submitDate, items[ ].description, items[ ].amount, and total attributes, set the *Editable* property to *No*.
8. Save your changes.
9. To verify, click on the Preview in Browser () icon in the presentation editor.

The Confirm Receipts presentation should look like this:



**ConfirmReceipts**

SubmittedBy

SubmitDate 7/17/2008

Description

CostCenter Sales

Items

	Description	Date	Amount	ReceiptChecked
1				<input type="checkbox"/>

Total 0.00

Comments

ReviewedBy

IsApproved ☐

Submit Cancel

Figure 27: Confirm Receipts presentation

10. Close the presentation editor.

## Defining the Confirm Receipts Activity & Screenflow

You have a condition transition to be used in case one or more receipts are missing or incorrect. We implement the Confirm Receipts activity as a screenflow.

This procedure is identical to the one you followed to define the Review Report activity.

To define the Confirm Receipts activity:

1. Follow the steps in [Defining the Review Report Activity](#) on page 46, but substitute "Confirm Receipts" wherever it says "Review Report".
2. Follow the steps in [Designing the Review Report Screenflow](#) on page 47, and here also use "Confirm Receipts" instead of "Review Report".

You have created the Confirm Receipts screenflow and set this screenflow as the main task of the Confirm Receipts activity.

## Running the Process

The basic Expense Report process is now complete, though we will add some features to it in Activity 5.

To run the ExpenseReport process:

1. Click the **Start Engine** icon (▶) to start Studio's built-in process engine.  
The **Start Engine** dialog box appears.
2. Check both the **Delete Process Instances** and **Delete Log Files** options, and click **OK**.
3. Once the engine has started, click **Launch Workspace** (🌐).
4. In the Workspace login page, log in as Peter Jones.
5. In the Workspace **Applications** panel, click on *Create Expense Report*.
6. Fill out the form with sample information. In the SubmittedBy field, enter Peter Jones exactly as you entered the login name. Add just one item for \$24.
7. Log out of Workspace and log in again as Paul Smith, the participant with the supervisor role.  
Because you entered an expense report totaling less than \$25, the instance should have gone directly to the Confirm Receipts activity. Therefore, if the process is working correctly, Paul's **Work Items** panel should show no work items (instances).
8. Log out and log in again as Mary White, the treasurer role participant.  
Mary should have the ExpenseReport process instance in her Inbox view in the **Work Items** panel. You should see **Confirm Receipts**, as a link in the *Activity* column.
9. Execute the Confirm Receipts activity by clicking on it.  
The Confirm Receipts form opens in the browser. It should show the item you entered previously when logged in as Peter Jones.
10. In the item line, select the ReceiptChecked checkbox, and click **Submit**.  
The form closes and the instance is removed from Mary White's inbox.

Even though the Expense Report process is simple, several sequences are possible. Here you ran the instance through the "fast-track" path by submitting an expense report for less than \$25. You can try the following scenarios:

- Total of more than \$2,500 - The instance should go back to the employee role.
- Total between \$25 and \$2,500, approved by supervisor and receipts checked - this is the happy path.
- Total between \$25 and \$2,500, rejected by supervisor. - Goes back to employee, who adds an explanatory comment.

- Total between \$25 and \$2,500, receipt not verified. - Goes back to employee, who removes item.

When you are done experimenting with the process, log off, go back to Studio, and stop the process engine.

## Activity 4 Summary

Congratulations! You have completed the main elements of the Expense Report process.

At this point you should understand Oracle BPM at the concept level. In this activity you added condition transitions. You also added a method to the Expense Report BPM object, and you have created presentations that provide different views of the same underlying BPM object.

In Activity 5 you refine the process with some additional features.

## Activity 5: Finishing Touches

---



The basic Expense Report process is now complete. You can refine it in several ways so that it is easier to use and maintain. In this activity you add some improvements that expose you to additional Oracle BPM features.

### Adding a Due Transition

In this task we add the Send Reminder automatic activity, which reminds the employee that a report he has submitted requires editing. Since we want this activity to execute after a given interval of time has elapsed, we connect it with a *due* transition.

The due transition is fired by a timer on the activity it is sourced from. You add a due transition that is executed when a certain amount of time has elapsed after an instance has arrived at the Edit Report activity.

To add the Send Reminder activity and the Reminder due transition:

1. Make the Employee swimlane wider, and shift the Edit Report activity down, so there is room to insert a new activity above it.
2. Right-click in the space between the Edit Report activity and the Begin, and click **Add Activity Automatic** .  
The **Activity** dialog box appears.
3. Enter `Send Reminder` in the **Name** field, and click **OK**.  
The Send Reminder automatic activity is added to the diagram.
4. Right-click on the Edit Report activity and click **Add due transition** .  
The mouse pointer will begin dragging a transition line.
5. Click on the Send Reminder activity.  
The **Transition from Activity** dialog appears.
6. Enter `Reminder` in the **Name** field.
7. In the Properties page of the **Transition from Activity** dialog, in the expression text box, enter `' 2m '` (complete with the single quotes) and click **OK**.  
This is an interval literal. For more information on interval literals, see Time and Interval Overview in the Studio Reference.  
The Reminder due transition is added. We set the interval at two minutes so you will not have to wait long when testing this part of the process. A reasonable reminder interval would of course be much longer, such as one day ('1d').
8. Click and drag on the Reminder due transition to curve it towards the left.
9. Right-click on the Send Reminder activity and click **Add unconditional transition**.
10. Click on the Edit Report activity.  
The unconditional transition is added from Send Reminder to Edit Report.

After completing this task, your process design diagram should look like this:

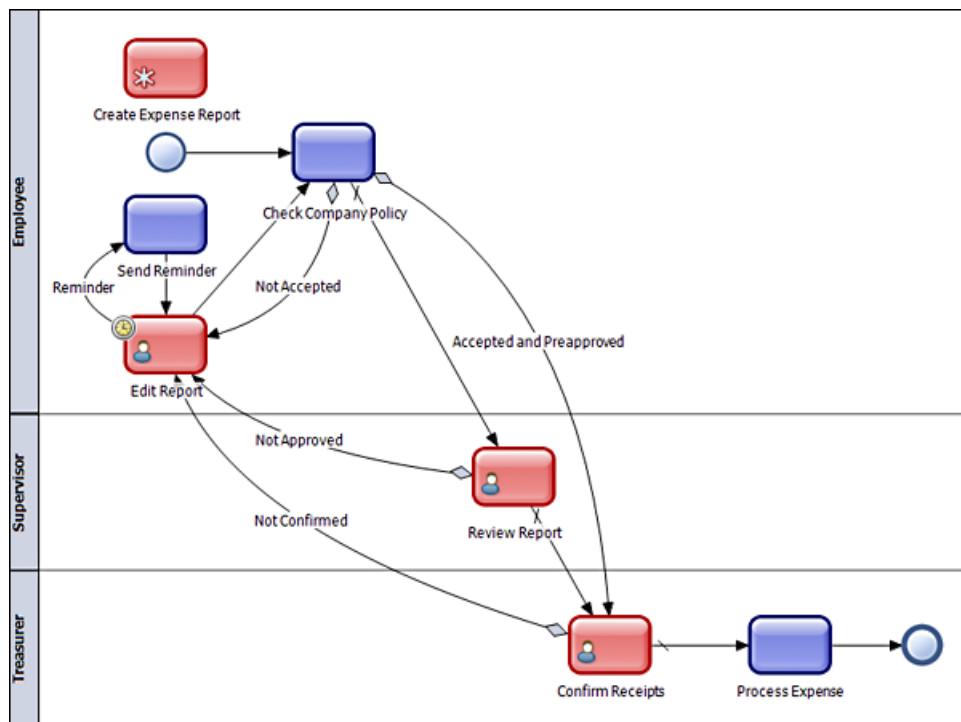


Figure 28: Expense Report process with Reminder Due transition

## Implementing the sendReminder Method

You have the Send Reminder activity, but have not yet implemented it. You will set the implementation type of its Main Task as a method. This method sends a reminder e-mail.


1. Right-click on the Send Reminder activity and click **Main Task** .  
The **Main Task** dialog box appears.
2. Verify that *Method* is selected in the **Implementation Type** drop-down list. If not, set it.
3. In the Method panel, click **New**.  
The **New Process Method** dialog box appears.
4. Enter `sendReminder` in the **Method Name** text box, and click **OK**.
5. Back in the **Main Task** dialog box, click **Edit**.  
A PBL code editor opens for the `sendReminder` method.
6. Enter the following code, exactly as shown, but replace *<your e-mail address>* with an e-mail address you have access to:

```
// Send reminder e-mail

reminderEmail as Mail
reminderEmail = Mail()
reminderEmail.from="<your e-mail address>"
reminderEmail.recipient=Participant(report.submittedBy).email
reminderEmail.subject="Expense Report Message"
reminderEmail.message="This is to remind you that a report you submitted
requires editing."

sender as MailSender
sender = MailSender(reminderEmail)

sender.send()
```

7. Save and close the editor.
8. For this to work, you also need to configure the Studio engine with a working outbound (SMTP) e-mail server. In the **Project Navigator**, right-click on the top of the ExpenseManagent project, and click **Engine Preferences** . The **Engine Preferences** dialog appears.
9. Enter the name of your SMTP mail server in the **SMTP Mail Server Name** field, for example *smtp.xyzcorp.com*.
10. Click **OK**.  
The engine dialog box closes with the new settings.


## Presetting Default Values

When you design a process, one of your goals should be that no participant should need to enter data that the system can obtain automatically. In this task you will modify the Expense Report process to preset user attributes and provide a reasonable default for the date attribute.

The ExpenseReport process requires participants to input their name. However, because the process engine already knows the name of any participant from the log-in process, this step should not be required.

In this task you modify the ExpenseReport process so that the `submittedBy` participant is preset and the `submitDate` attribute has a default value.

You add an automatic activity to the Submit Report screenflow. In this activity, you create a method to set the default values.

1. In the **Project Navigator**, expand Processes and double-click on Submit Report.  
The design editor for the Submit Report screenflow appears.
2. Insert an automatic activity () between the Begin and Input Report activities. Name this activity `Initialize Report`.
3. Right-click on the Initialize Report activity you just added, and click **Main Task**.  
The **Main Task** dialog appears.
4. Set the **Implementation Type** list to *Method*.
5. In the **Method** panel, click **New**.  
The **New Process Method** dialog box appears.
6. Accept the default name offered, `initializeReport`, in the **Method Name** field, and click **OK**.
7. Back in the **Main Task** dialog box, click **Edit**.  
A code editor opens for the `initializeReport` method.
8. In the editor, enter the following code exactly as shown:

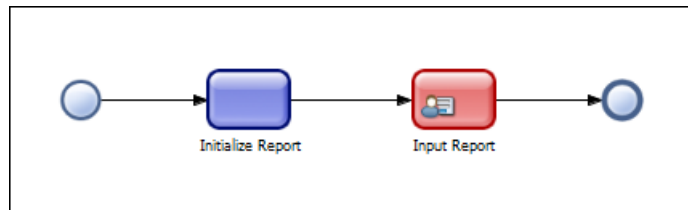
```
reportSf.submittedBy = Participant.id
reportSf.submitDate = 'now'
```

The first line uses the built-in `Participant` object, which is in the `Fuego.Lib` module.

9. Save your changes and close the editor.

Your screenflow should now look like the following:





**Figure 29: Submit Report screenflow with Initialize Report activity**

You can follow these steps to in the Review Report screenflow, to preset the `reviewedBy` attribute.

To prevent the user from editing these preset values, you can also set the `submittedBy` field (which should have been named `text0`) to read-only in the `SubmitReport` presentation. To do this, set its *Editable* property to *No*. You can do the same with the `reviewedBy` text field in the `ReviewReport` presentation.