ALBPM Basic Tutorial

Version: 6.0

Contents

Overview

This tutorial presents a step-by-step introduction to AquaLogic BPM Studio. After completing this tutorial, you will have a basic understanding and skill set to be able to model and deploy simple business processes.

Basic Requirements

To complete this tutorial, you will need to have AquaLogic BPM Studio 6.0.2 installed in your computer. The latest version of Studio is available in the *BEA Downloads AquaLogic BPM* page.

You must use Studio in the Developer profile. You will not be able to complete the tutorial in the other profiles.

Using the Tutorial

- If you are not familiar with AquaLogic BPM or BPM concepts in general, you should begin by reading *Basic Concepts* on page 5.
- Once you have a basic understanding of AquaLogic BPM, you can proceed to read *The Expense Report Process* on page 9. This section presents both the business process to be modeled and the process as designed with ALBPM.
- Five activities follow, each with a number of tasks, where we show you how to build the ExpenseManagement project together with the ExpenseReport process. The activities are the following:

  1. *Activity 1: Initial Steps* on page 13
  2. *Activity 2: Building the Happy Path* on page 19
  3. *Activity 3: Defining the Expense Report Object* on page 24
  4. *Activity 4: Adding Alternative Paths* on page 37
  5. *Activity 5: Finishing Touches* on page 49

Basic Concepts

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

What Is a Business Process?

A business process represents a specific set of business tasks and activities which must be executed to reach a well-defined outcome. When this outcome is reached, the process is complete.

Examples of simple processes include hiring an employee, processing a sales order, or reimbursing a business expense. More complex business processes can also be designed according to the needs of a particular organization.

Sometimes it may not be possible to reach the main goal of the process. For example, the shipping clerk may havre to cancel a sales order because the product is out of stock. Therefore, a business process must allow for different possible end conditions besides the principal objective of the process. Ideally it will allow more than one way to reach this objective. For instance, if the product is out of stock it may be possible to offer an equivalent alternative. In turn, this offer may be accepted or rejected. A range of possibilities can thus be included in the process.

The Happy Path

The most straightforward or "most expected", path through the process is frequently called the *happy path*. This path leads directly to the goal of the process. In the case of a purchase order, the happy path would go from order to delivery with no complications such as an out of stock condition.

When designing a process, it is usually a good idea to start out with the happy path, and gradually add the handling of more complex conditions to it. To make the design intention clear, it is also good practice to lay out the happy path as a straight line from left to right, and to show less common paths as deviations from the happy path.

Activities

Business processes are broken down into logical steps called *activities*, each of which can comprise one or more *tasks*. When activities are executed automatically by the system, they are called *automatic activities*. When human input is required, they are known as *interactive activities*. The activities of the business process are linked by *transitions*, which determine the order in which they are performed and the basic workflow for the process.

Roles and Participants

Each interactive activity belongs to a *role*. In turn, roles are assigned to *participants*, who are the actual individuals who interact with the process. A participant may have one role or many. The participants who can perform an activity are those who have been assigned the role the activity belongs to.

Exceptions

In the real world it is often impossible to predict every possible situation, so a business process usually includes a way to deal with *exceptions*. Exceptions are special situations where it is not possible to reach one of the normal pre-defined outcomes of the process. However, the process can include a way to handle these events and this usually involves calling attention to the situation so it can be resolved by one of the participants.

What Is a Process Instance?

If you think of a business process as a sequence of steps, then you can think of a process instance as a specific item going through those steps.

The following is a somewhat formal definition:

- A business process instance is a specific item proceeding through a business process.

For example, in a business process which handles purchases, each instance will be an individual purchase order. There can be any number of instances traversing a business process, as there can be any number of purchase orders going through a purchase order system.

Every instance has a specific history and properties. A purchase order usually contains a customer name, a list of items, an amount due, dates of delivery and payment, and other required data. An instance will also have various status conditions. In the case of a purchase order, you would want to know if it has been approved, if it has been paid for, or if the requested products have been shipped.

Each instance has a beginning and end, as defined in the business process. As the instance proceeds through the process, it will be worked on by various participants or processed automatically by software.

Note: In order to understand what a business process *instance* is, you must first understand the concept of a *What Is a Business Process?* on page 5.

What Is an Activity?

Activities define a manual or automated task that conforms one step within a process design. Adding a new activity allows you to create a new step and assign it to a role in a process.

A manual activity requires end user intervention whereas an automatic activity can be automatically completed by the Engine. An activity can include one or more tasks.

The following table describes different categories of Activities. For detailed information on each Activity see *Activity Types*:

Category	**Description**	**Activities**
Process Initiation/Termination Activities	Function as a beginning and end point for the process. The activities are automatically generated and define the scope of the process.	Begin, End
Human Interaction Activities	Allows user interaction with process.	Interactive, Grab, Decision
System Interaction Activities	Handle automatic interactions with business systems.	Automatic
Organizational Interaction Activities	Allow communication with other areas and processes of an organization.	Process Creation, Termination Wait, Process Notification, Notification Wait, Dynamic Process Call
Process Control Activities	Control process flow or generate copies of a process instance to allow flow through multiple paths simultaneously.	Split , Split N, Join, Conditional
Global Activities	Handle global requirements that are not associated with a specific process instance.	Global Creation, Global Automatic
Miscellaneous Activities	Provide other functionality with a process	Connectors, Measurement Marks

Activity Naming Conventions

It is recommended that you name your activity with a verb followed by a noun specifying the Activity's role within the Process. Providing descriptive names for your Activities allows your process to be self-documenting. For example, Create Order, Ship Product, Check Credit are all useful Activity names.

Note: After an Activity name has been defined, it cannot be changed. However, you can change the Activity label which is displayed to end users.

What Is a Transition?

A transition is the bridge between two activities.

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

AquaLogic BPM provides many types of transitions. The most common transitions are *unconditional*, *conditional*, and *due*. The *exception* transition is a little bit more advanced, but is also used frequently, while the *business rule* transition is new:

	Transition	Description
	Unconditional	Instances flow through the transition unconditionally.
	Business Rule	Instances flow through the transition if the specified dynamic business rule evaluates to `true`.
	Conditional	Instances flow through the transition if certain conditions are met.
	Due	Instances flow through the transition according to time conditions.
	Exception	Instances flow through the transition if an exception occurs.

The *compensate*, *message-based*, and *precedence* transitions are used less frequently. If you are just beginning to use ALBPM, you do not need to be familiar with these yet:

	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 activity in the event that BP-method failure occurs.
	Message-based	Instances flow through the transition if the activity that manages different argument mappings receives a message.
	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 dotted transition.

Which transition is used?

All activities at least have an outgoing unconditional transition so there is always a way to continue the process. However, in most real-world processes several activities also have outgoing *conditional* transitions. In this case, the conditional transitions are evaluated first, and the unconditional transition is taken only if the conditional transitions all evaluate to `false`. In programming terms, the unconditional transition is like an *else* clause in an if-then-else construct.

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

Due transitions act separately. They "pull" the instance from the activity as soon as a time condition is met. In this case, all other outgoing transitions are ignored.

The Expense Report Process

Process Description

To design a business process, you must understand the business need that the process solves, and the elements (such as people and data) that the process requires.

In this tutorial we describe how to build a business process for managing expenses. Expense management is a common activity, and can be made simple enough to use in a tutorial.

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.

While the process we build in this tutorial is simple, it illustrates how a process is designed and implemented. A simple process can also be developed as feedback from users is received. One of the main advantages of having a process model which is also executable is that the process can be refined quickly through successive design iterations.

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:

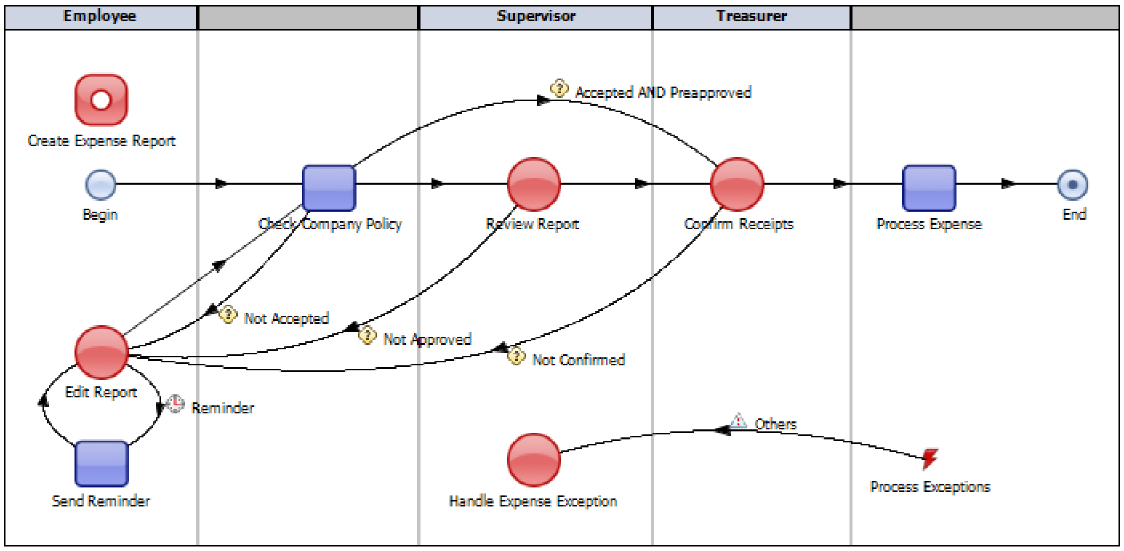
1. An employee purchases a product or service he requires. For instance, a sales person on a trip rents a car.
2. The employee submits an expense report with a list of items, along with the receipts for each item.
3. A supervisor reviews the expense report and approves or rejects the report. Since the 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 so, he accepts the expenses for processing (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.

In the following sections, we describe an AquaLogic 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. In this tutorial we show how this process is developed.

The following image shows the complete Expense Report process diagram as seen in the Studio editor:



To examine the Expense Report process in Studio, you can download the *ExpenseManagement.exp* exported project file. 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 ALBPM 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*, *Edit Report*, and *Handle Expense Exception*. These activities require input from a participant, and are described in detail in *Interactive Activities* on page 10.
- 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 11.
- Transitions, which establish the flow of the process. *Unconditional*, *Conditional*, *Due*, and *Exception* transitions are used. For detailed descriptions of each see *Transitions* on page 11.

Interactive Activities

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

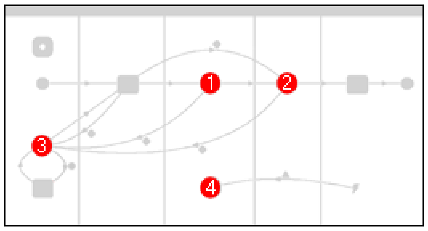


Figure 1: Interactive Activities

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 if corrections are required because the report was rejected, receipts are missing, or company policy is not met (see below).
4. *Handle Expense Exception* - The instance reaches this activity if there is a problem somewhere in the process. The activity belongs to the supervisor, who has a decision-making role and can determine how to proceed.

Automatic Activities

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

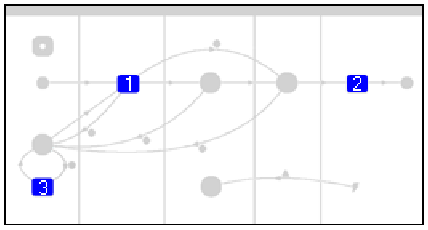


Figure 2: Automatic Activities

1. *Check Company Policy* - This activity is the first performed after the employee submits a report. From here, basic business rules can be checked. For example, if the expense amount is lower than a certain threshold, it does not need to be approved by the supervisor.
2. *Process Expense* - This is where the approved and checked expense report would be processed for payment or accounting. For the purpose of this tutorial, it is a placeholder, since we won't actually process any expenses.
3. *Send Reminder* - If the expense report returns to the employee, he will usually have a limited amount of time to re-submit his report. This activity will e-mail the employee a reminder that his report is pending if a certain amount of time has elapsed.

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.

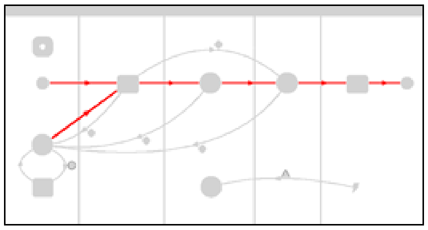


Figure 3: Unconditional Transition

- *Conditional* transition (?) - The instance flows through a conditional transition if an expression defined for the transition evaluates to `true`. If an unconditional transition also flows from the same activity, the *conditional* transition (when `true`) takes priority.

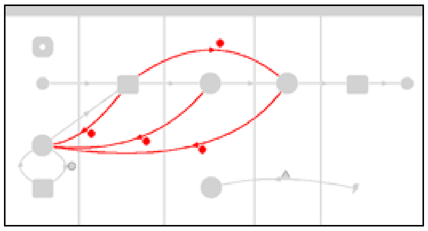


Figure 4: Conditional Transition

- *Due* transition () - The instance will flow through this transition when a specified period of time has elapsed in the origin activity. This is used in the Expense Report process to trigger the Send Reminder activity.

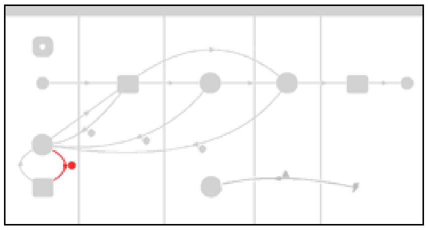


Figure 5: Due Transition

- *Exception* transition () - The instance will flow through this transition when an exception occurs.

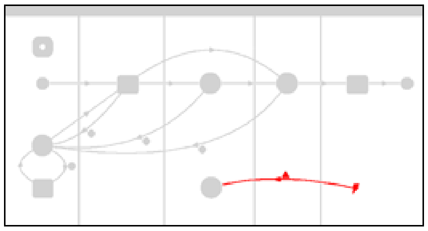


Figure 6: Exception Transition

Activity 1: Initial Steps

In this activity you will create the ExpenseManagement project, the ExpenseReport 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 AquaLogic BPM Studio. Pay special attention to the **Project Navigator** and the **Process Editor**.

To insure 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 AquaLogic BPM Project:

1. If it is not already open, start AquaLogic BPM Studio.
2. In Studio, click **File ➤ New ➤ BPM Project** from the menu, or click on the **New BPM Project** icon ( ) in the toolbar.

   The **New BPM Project** Wizard appears.
3. Enter `ExpenseManagement` in the **Project Name** field.
4. Indicate the storage location of project files in the **Project Root Folder** field, or leave the default location shown.
5. Click **Next**.

   The **New Project Information Summary** page appears.
6. Review the project information and click **Finish**.

The new project is created and added to the **Project Navigator**, as shown below:

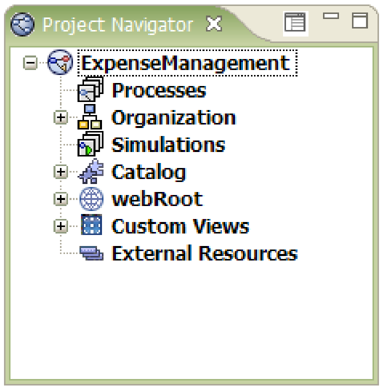


Figure 7: Project Navigator

Creating a Process

Each project can have many processes. In this task, you will create the ExpenseReport process of the ExpenseManagement project.

To create the ExpenseReport process:

1. In the **Project Navigator**, expand the ExpenseManagement project.
2. Right-click **Processes** (), then select **New** ➤ **Process** ().
   The **Process** dialog box appears.
3. Enter `ExpenseReport` in the **Name** field.
4. Select the **Generate Events for all Activities** option. This will later allow you to see every step the process instance (the expense report) has been through.
5. Click **OK**.

   A new window appears with the new process. This is known as an *editor*. The design view is shown, as you can verify 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 are always created automatically, together with the unconditional transition which connects them.

Creating a Role

Roles created in Studio are known as *abstract roles*, because they may be renamed or consolidated when the process is actually deployed to production. In Studio, roles can be added from the **Project Navigator** or from 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. If it is not already expanded, expand the ExpenseManagement project.
2. Expand the Organization section ().
3. Right-click Roles and select **New** ().
   The **Name** dialog box is displayed.
4. Enter `Employee` in the **Name** field, then click **OK**.
   The Employee role is created in the organization, and an editor opens for the 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. Close the role editor.
7. In the **Design** view of the ExpenseReport process editor, right-click somewhere to the left of the Begin activity, and click **Add Role**.
   The **Role** dialog box appears.
8. From the **Name** drop-down list, select *Employee* and click **OK**.
   The Employee role swimlane is added on the left-hand side of the process design diagram.

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

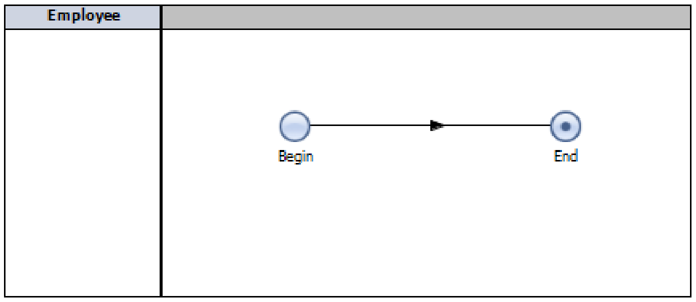


Figure 8: Employee role in ExpenseReport process

Creating a Role from the Process Editor

This tutorial 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 will 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 method from that used in the previous task.

To create the Supervisor role from the Process Editor:

1. In the **Design** view of the ExpenseReport process editor, right-click somewhere between the Begin and End activities, and click **Add Role**.
   The **Role** dialog box appears.
2. Click **New**.
   The **Role Properties** dialog box appears.
3. Enter `Supervisor` in the **Name** field. Note that the **Label** field is set to the same value.
4. Click **OK**.
5. Back in the **Role** dialog box, verify that the **Name** field says *Supervisor*, and then click **OK**.
   The Supervisor role swimlane is added on the left-hand side of the process design diagram.

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

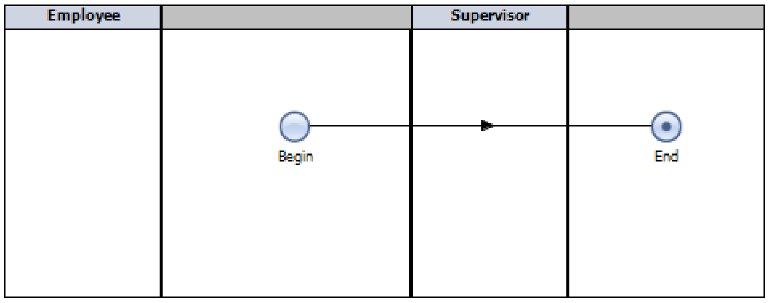


Figure 9: Supervisor role in ExpenseReport process

Adding a Global Creation Activity

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

A global creation activity creates a process instance which will then start flowing 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. In the process design editor, click on the **Global Creation** icon (). When you do this, do *not* hold the mouse button. The mouse cursor will go into insertion mode in the process design editor.
2. Insert the global creation activity in the Employee swimlane, towards the top (see the image below), by clicking on the insertion point.
   The **Activity** dialog box appears.
3. Enter `Create Expense Report` in the **Name** field. 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 created.

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

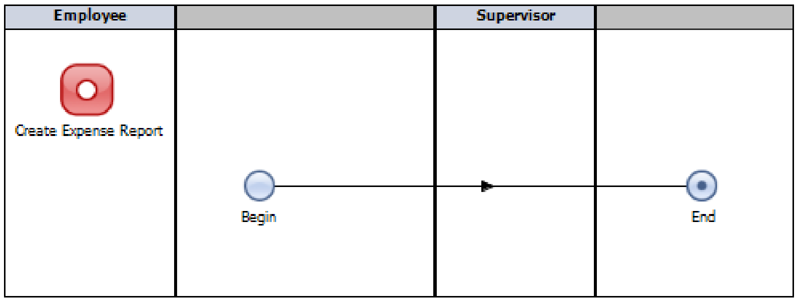


Figure 10: Create Expense Report Global Creation Activity

Creating Participants

To perform interactive activities in the process, you need *participants*. Participants are the people who log into the system and perform activities.

The activities participants can perform depend on the roles assigned to them. Hence, in this task we also assign a role to the participant.

Our first participant is called Peter Jones. Peter is in sales and reports travel expenses. For the purposes of this process, he is assigned the role of employee.

Our second participant is called Paul Smith, who will be assigned the role of Supervisor.

Keep in mind that in most cases, you 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 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 box appears.
3. Enter `Peter Jones` in the **Name** field, and click **OK**.
   A participant editor window opens for Peter Jones.
4. In the editor, go to the **Roles** section at the bottom (you may need to scroll down), and click **Add**.

The **Roles** dialog box appears.

5. Select *Employee* from the **Roles** list, and click **OK**.
   The Employee role is added to the roles list.
6. Save the changes to Peter Jones by clicking on the **Save** icon () or click **File ➤ Save** from the menu.
7. Close the editor.
8. 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.

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.

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 box 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 will close and the **Stop** button () replaces the **Run** button.
4. Click the **Launch WorkSpace** () button.
   The AquaLogic BPM WorkSpace login page will appear in your default Web browser.
5. Log in as `Peter Jones`.
   The WorkSpace main page will appear for user Peter Jones. Note that in the **Applications** panel there is one entry called Create Expense Report. This appears here because of the global creation activity we added to the Employee role previously, and because Peter Jones has the Employee role. If you click on this entry, you will create an ExpenseReport instance.
6. Click on *Create Expense Report* in the WorkSpace **Applications** panel.
   An ExpenseReport 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, click the **Open Search** button () in the **WorkList** panel.
   The **Search** pane of the **WorkList** panel opens.
8. Expand the **Show Instances** section by clicking on the Show icon ().
   The **Show Instances** section expands.
9. Check the **Completed** option in the **status** line, and click **Search**.
   The **WorkList** panel will show 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 WorkList panel.
11. We are done as Peter Jones for now, so log out of WorkSpace by clicking Logout in the upper right of the page.
    The logout page will appear.
12. We now log in as Paul Smith, the supervisor, to see his view of things. Click Re-Login and then in the login page, log in as Paul Smith.
    The WorkSpace session for Paul Smith appears. Note that here, in the **Applications** panel, Create Expense Report is absent. 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 Studio's process execution engine by clicking the **Stop** button (■).
The Run button will replace the Stop button.

After completing this task, you can begin to see how the process design maps 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 ExpenseReport 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 will add both automatic and interactive activities in the most expected path (also called the *happy path*) of the ExpenseManagement project.

Adding the Check Company Policy Activity

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

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

To add the Check Company Policy automatic activity:

1. We will insert the activity in the first automatic activity swimlane, which is to the right of the Employee swimlane. The Begin activity is now in this swimlane. To make some room, click and drag the Begin activity towards the left, to the Employee swimlane, right below the Create Expense Report activity.
   The Begin activity is now in the Employee swimlane. This has no effect on the behavior of the process, but provides a visual cue that the Employee role initiates the process.
2. Click on the automatic activity icon (🟦) in the editor toolbar, and insert the automatic activity in the first unlabeled (automatic) swimlane. 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.
   The **Activity** dialog box appears.
3. Enter `Check Company Policy` in the **Name** field, and click **OK**.
   The Check Company Policy automatic activity is added to the process.

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

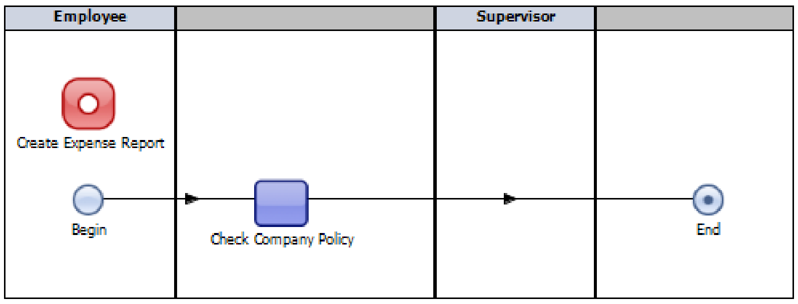


Figure 11: Process with Check Company Policy activity.

Adding the Review Report Activity

In this task you will add the Review Report activity, an interactive activity that will be used by 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, between the Check Company Policy and End activities.
   The **Activity** dialog box appears.
2. Enter `Review Report` in the **Name** field, and click **OK**.

The Review Report interactive activity is added to the process.

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

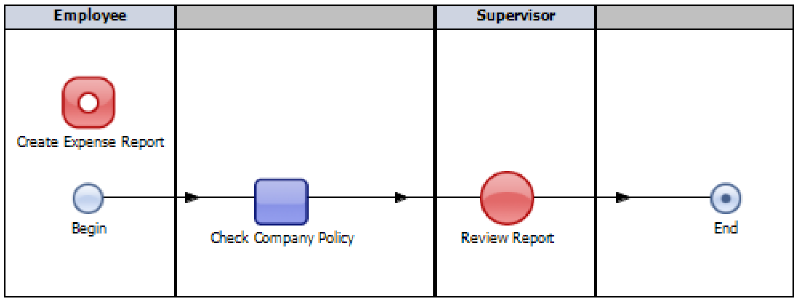


Figure 12: Process with Check Company Policy activity.

Adding the Confirm Receipts Activity

In this task you will add the Confirm Receipts activity, an interactive activity which will be used 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 on the interactive activity icon (●) in the editor toolbar, and insert the interactive activity right before the End activity, in the *automatic* swimlane.
   You cannot insert an interactive activity in an automatic swimlane, so Studio needs to ask you for a role. Hence, the **Role** dialog box is displayed.
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**.
   The **Role Properties** dialog box 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. Click OK.
   The **Activity** dialog box appears.
5. Enter `Confirm Receipts` in the **Name** field, and click **OK**.
   The Treasurer swimlane is added to the process diagram, and the Confirm Receipts interactive activity is inserted into the Treasurer role.

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

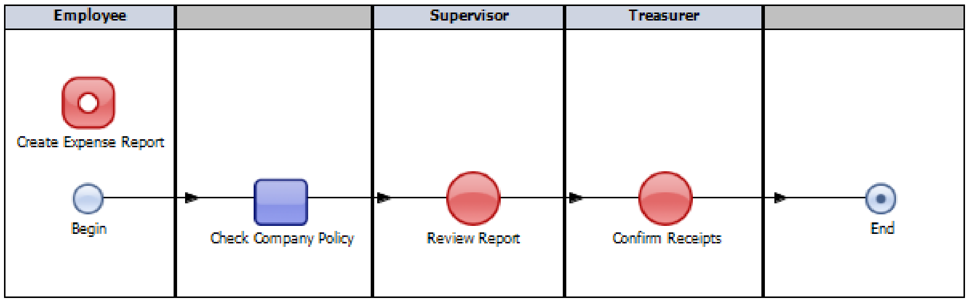


Figure 13: Expense Report process with Confirm Receipts activity.

Adding the Process Expense Activity

The last activity in the happy path is the Process Expense activity. This activity is where the process would perform whatever actions were required to fund or reinburse the expense reported.

Now, we add the Process Expense activity. Being a tutorial project, it will 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 the automatic activity in the automatic swimlane where the End activity is, immediately before it.
   The **Activity** dialog box appears.
2. Enter `Process Expense` in the **Name** field, and click **OK**.
   The Process Expense automatic activity is added to the process diagram.
3. Save your changes.

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

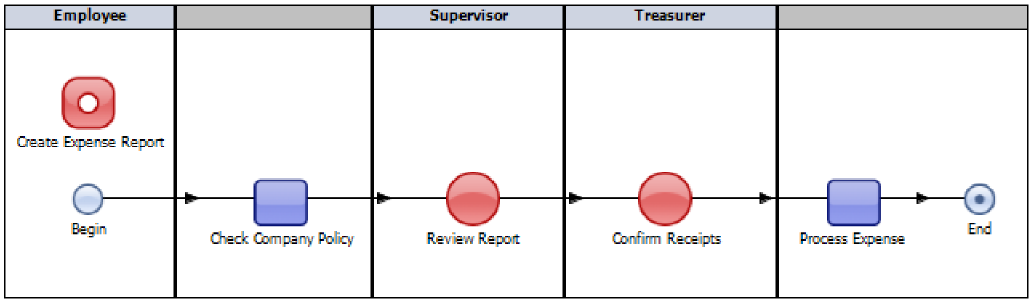


Figure 14: Expense Report process with complete most expected path.

Running the Process as an Employee

You cannot yet input data into the process, but you can now run the process and 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. Check both the **Delete Process Instances** and **Delete Log Files** options, and click **OK**.
3. Click the **Launch WorkSpace** ( ) button.
4. In the WorkSpace login page, log in as `Peter Jones`.
   The WorkSpace main page will appear for user Peter Jones.
5. Click on *Create Expense Report* in the WorkSpace **Applications** panel.
   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 the **Open Search** button ( ) in the **WorkList** panel.
   The **Search** pane of the **WorkList** panel opens.
7. Expand the **Show Instances** section by clicking on the Show icon ( ).
   The **Show Instances** section expands.
8. Set the **assigned to** drop-down list to *All*, and click **Search**.

The **WorkList** panel will show 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 will log in again, so do not exit the browser.

Running the Process as Supervisor and Treasurer

In this task you will continue to see what the process looks like to users. In this case, you will follow the process instance with the Supervisor and then the Treasurer roles.

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 WorkList panel, without having to perform any search. This is because Paul, as supervisor, must now perform the Review Report activity. In effect, you did the equivalent of sending the "Expense Report" to his inbox. This is why the **WorkList** panel shows the Inbox view, which is the defaul view. You can choose other views from the **Views** panel.
2. As Paul, you approve the imaginary expense report (we will build the actual report later), and click on the Send icon ( ) to send it to the next activity in the process.
   The process instance will be sent to the next activity, and will disappear from the WorkList of user Paul Smith.
3. The next activity is Confirm Receipts, 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. You need to go back to Studio to do that, so switch to Studio but do not close the browser window.
5. We don't need to stop and restart Studio's process execution engine every time we make a change. Often, we can simply reload a running project. So *do not* stop the process execution engine at this time. From 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 *Creating Participants* on page 16.
7. Click the **Reload** button ( ).
   Studio will display an "Operation in progress" message, and the project is reloaded.
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, because Mary has the Treasurer role.
9. To see the steps the instance has been through so far, in the WorkList panel, click anywhere on the line of the Expense Report instance, but not in the checkbox or **Send** icon.
   The Expense Report instance will be highlighted in yellow, and information about it will appear in the Instance Detail panel.
10. In the Instance Detail Panel, click the Process Image icon ( ).
    A process diagram will appear. The path the process instance has followed so far will be highlighted in red.
11. Once you have examined the diagram, close the window it's in.
12. In the Instance Detail panel, click the rightmost tab, **Audit Trail** ( ).
    The same activities highlighted in the diagram appear listed in a table.
13. Mary "confirms" the imaginary receipts, so click the **Send** icon ( ) on the **WorkList** panel.
    The process instance disappears from Mary's inbox and goes 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 most expected path, as well as all of the roles you will define.

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 also from role to role.

You now have the basic process design for the expected path an Expense Report will take, but you don't yet have the report itself. You will create the report in Activity 3.

Activity 3: Defining the Expense Report Object

Just about any process works on information of some kind. Here we will define a BPM object which will contain information about the expense report itself, as well as the state of the expense report in terms of the process.

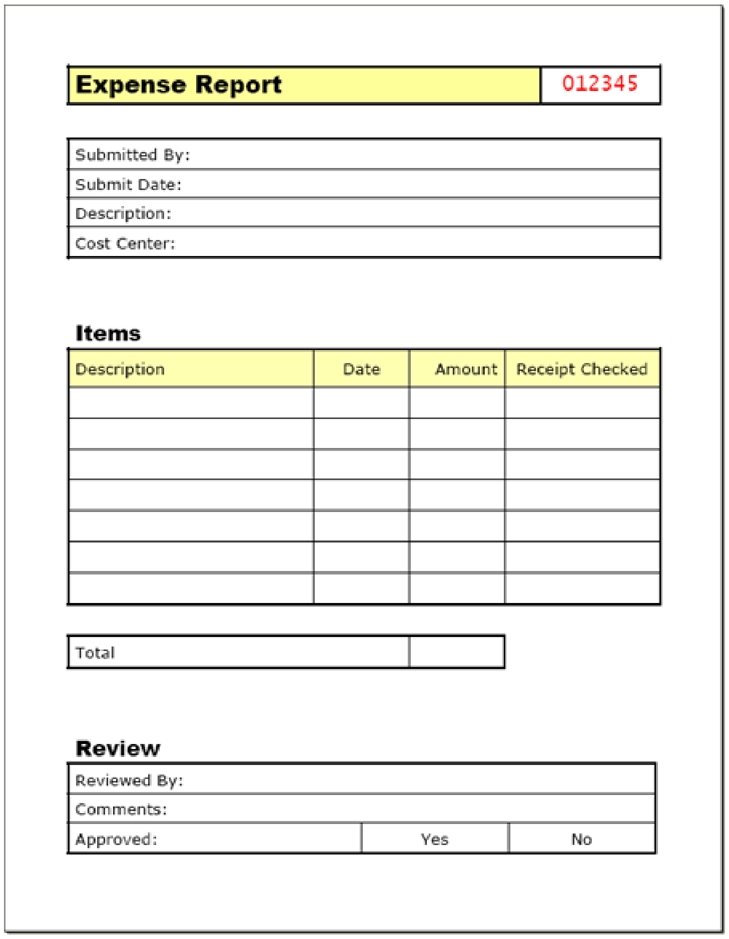
In a real world process you may not necesarily handle the expense report data with a BPM object. Instead, you might use a pointer, such as an ID, to access a record in a corporate database.

That said, you will always handle a at least some information within the BPM process. Here we show you how to 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 only want to learn how to design the process diagram, you can skip this step and go on to Activity 4.

Expense Report Data

To begin, first 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 15: 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.

You can think of this form as containing three sections: a header, with basic information about the submittal, the items section, with the "contents" of the form, and 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 complicated to implement, we will begin with the fixed fields. Our first step is to list these fields and the type of data they contain:

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

To keep track of what you are doing, you can print the expense report form shown. Use either the provided *PDF version*, or the Microsoft *Word version*.

Creating a BPM Object

In the Expense Management project, you will store all the information about expense reports in a BPM object, which is a type of object you can define in ALBPM Studio to store data during the life of the instance.

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. Enter `ExpenseComponents` in the **Module Name** field, and click OK.
   The ExpenseComponents module is added to the catalog.
4. Right-click on the ExpenseComponents module icon and click **New ➤ BPM Object** ( ).
   The **BPM Object** dialog box appears.
5. Enter `ExpenseReport` in the **Name** field, and click OK.
   The ExpenseReport BPM object is added to the ExpenseComponents module.
6. Expand the ExpenseComponents module and then 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:

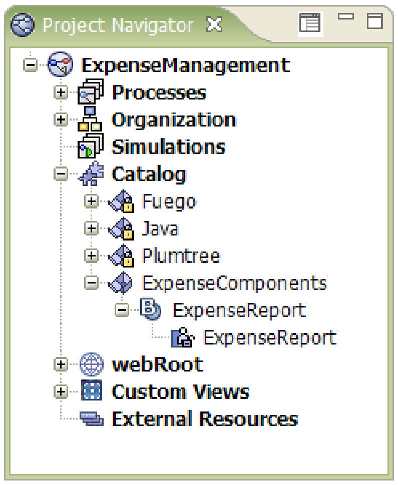


Figure 16: ExpenseReport BPM Object in ExpenseComponents module

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

Defining BPM Object Attributes

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

You 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 24. We leave the implementation of this number to you, so you can test your skills once you have completed the tutorial.

We will 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 (), and click **New ➤ Attribute** ().
   The **Attribute** dialog box appears.
2. Enter `submittedBy` in the **Name** field.
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, check **Not Null**. This tells ALBPM that the field is mandatory.
5. Save () the changes and close the editor window by clicking on the **X** in the *bottom* tab. If you close from the top tab, you will close the BPM Object.
6. You will now define the Submit Date attribute. Again, in the Project Navigator right-click on the ExpenseReport BPM Object (), and click **New ➤ Attribute** ().
   The **Attribute** dialog box appears.
7. Enter `submitDate` in the **Name** field.
8. Select *Time* from the **Type** drop-down list, and click **OK**.
   The attribute editor for the submitDate attribute opens.
9. Select *Date Only* from the **Time Precision** drop-down list.
10. As before, check **Not Null** in the **Storage Constraints** section.
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	*none*
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:

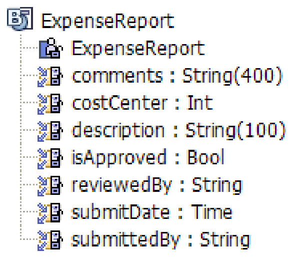

```
ExpenseReport
    ExpenseReport
    comments : String(400)
    costCenter : Int
    description : String(100)
    isApproved : Bool
    reviewedBy : String
    submitDate : Time
    submittedBy : String
```

Figure 17: ExpenseReport BPM Object Attributes

In the next task, you will 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, so you can have a list of items. 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 (), and click **New ➤ Group** ( ).
   The **Group** dialog box appears.
2. Enter `items` in the **Name** field, and click **OK**.
   The `items` group is created. It contains an object called Items, which in turn contains a method by the same name.

3. To create an attribute, right-click on the Items object (), and click **New ➤ Attribute**.
   The **Attribute** dialog box appears.
4. Enter `description` in the **Name** field, and set the **Type** drop-down 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 the above steps, 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 be able to see the following:

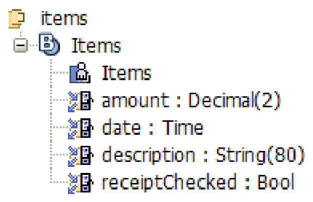


Figure 18: ExpenseReport BPM Object Items Group

In the next task, you will add a *virtual* attribute, for 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 is that calculates the value to be returned. You will use a virtual attribute to implement the total amount of the expense report.

To define a virtual attribute:

1. In the **Project Navigator**, right-click on the ExpenseReport BPM Object (), and click **New ➤ Attribute** ().
   The **Atribute** dialog box appears.
2. Enter `total` in the **Name** field.
3. Select *Decimal* from the **Type** drop-down list, and click **OK**.
   The `total` attribute is created, and the atrribute editor for it opens.
4. Set the **Decimal Digits** to 2.
5. In the **Storage Constraints** section, set the **Virtual** option.
6. Click **Save** () and close the editor.
   The `total` attribute becomes virtual.
7. In the **Project Navigator**, expand the `total` attribute.
   Two methods are shown: Read Access () and Write Access ().
8. Double-click on the Read Access icon, or right-click and click **Open** ().
   A PBL (Process Business Language) method editor window opens. The method is named `read_access_code_total`.
9. 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, 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
```

10. 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, there are 676 two-letter combinations, but only 50 two-letter state codes in the US. To assure data integrity it is best if the user is allowed to chose one of the 50 states instead of entering any possible two-letter code. It is good practice to limit input to the set of valid values whenever this is possible.

BPM object attributes can be configured with a set of predefined valid values. When you do this, the attribute will be presented to the user as a drop-down list, rather than a field. Thus, the user will only be able to select from the choices provided.

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

Valid Values Setting	Description	Presented 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 fixed 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. For example, you can pull the information from a database. The static list is easy to configure without writing any code, so here we use a static list. To add a static list of valid values to an attribute:

1. In the Project Navigator, expand the ExpenseReport BPM Object (B).
2. Double-click on the `costCenter` attribute.
   The editor for the `costCenter` attribute will open.
3. In the **Valid Values** section, select **Static List**, and check the **Edit Value Descriptions** option.
   A Value / Description table will appear.
4. To add an entry to the table, click on the plus sign and 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

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

With this step, the BPM object's data model is complete. In the following task you will define an *instance variable* to hold the BPM object in each instance.

Creating the Instance Variable

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

Instance variables can be of any type, such as a string or a decimal, and there is no requirement that an instance variable be a BPM object. We use a BPM object here to keep things simple, as we only need to keep track of one variable and handle all of our information in one place. In a real-world process, you may or may not find this to be the best strategy, depending on your particular situation.

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

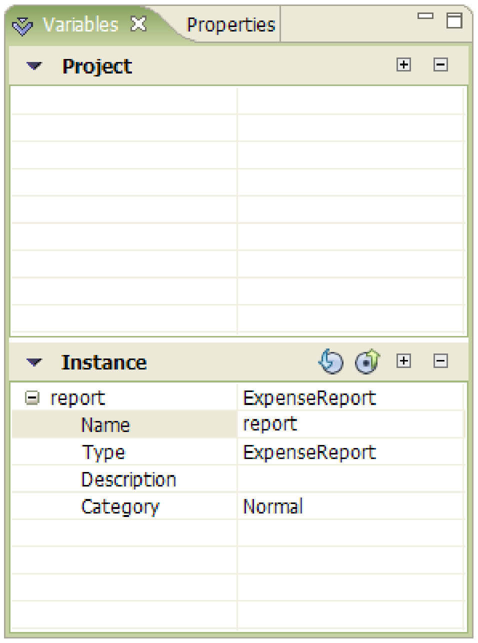
To create the instance variable:

1. Make sure the **Variables** window is visible. It is usually on the 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 instance variables.
2. In the **Instance** section of the **Variables** window, click on the Add icon ( ).
   A variable entry is added to the instance variable list. This variable will have a default name, such as `instanceVar1`, and properties.
3. Edit the *Name* property to `report`.
4. Click on the *Type* property, and click on the **Browse** button (...).
   The **Type** dialog box appears.
5. In the **Type** drop-down list, select *<Component>*.
   The **Component** section of the **Type** dialog box appears.
6. Expand *ExpenseManagement*, then, *Catalog*, and then *ExpenseComponents*.
7. Select *ExpenseReport*, and click **OK**.
   The variable type is changed to ExpenseReport.
8. Still in the **Variables** window (see image below), right-click on the report instance variable, and click **Map as process incoming argument** ().

   This allows the process to receive an expense report object (as created by the Create Expense Report global creation activity) into the `report` instance variable.
9. Save your changes.

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



report	ExpenseReport
Name	report
Type	ExpenseReport
Description	
Category	Normal

Figure 19: Variables window with report instance 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

You have the basic process, and you have a place to put the data. Now you must provide an interface where the participants of the process can input or review this data when it reaches their activity. In ALBPM, the two main ways of doing this are to create a *presentation*, or to use JSP pages.

Both methods are used to create a web page or form which the participant will use to interact with the data. 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.

You will start with the presentation for the user who submits the expense report. To define the SubmitReport presentation:

1. In the Project Navigator, right-click on the ExpenseReport BPM Object (), and click **New ➤ Presentation** ().
   The **Presentation Wizard** appears.
2. Enter `SubmitReport` in the **Presentation** field.
3. The SubmitReport presentation will be 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 which will be shown 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

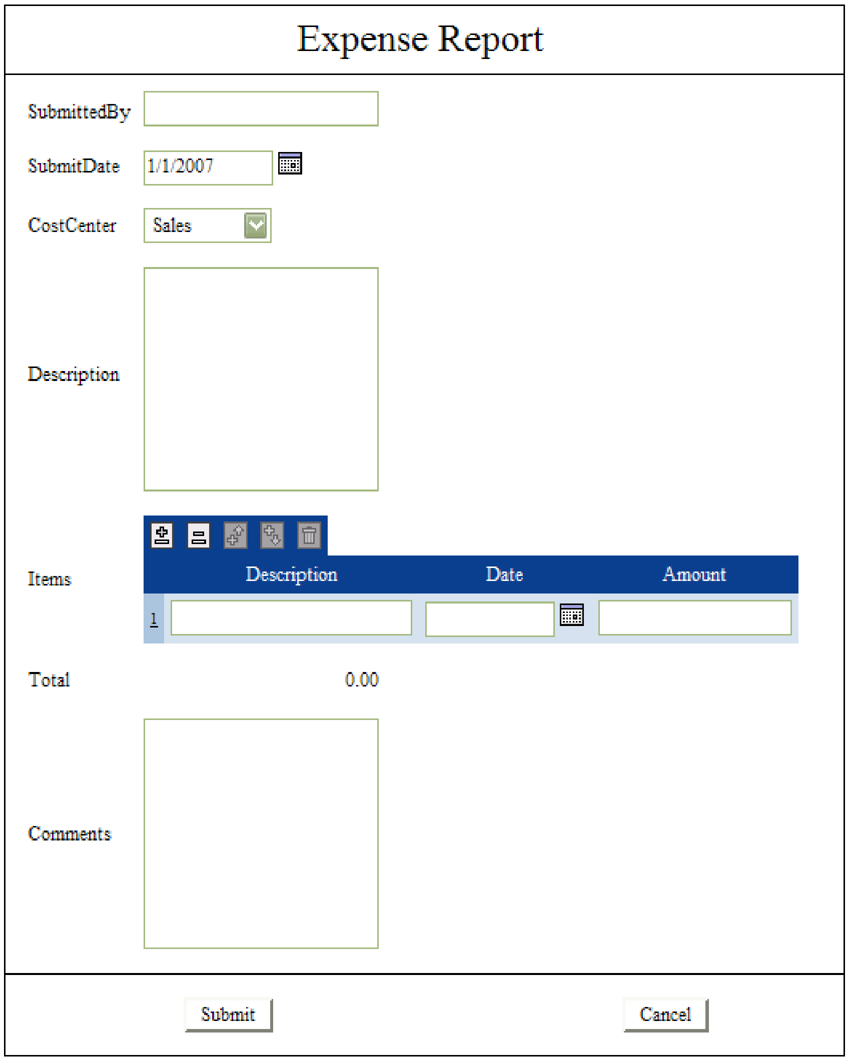
total

comments

The fields will be placed in the presentation as they are ordered in the attribute list. 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.

6. Click **Finish**.
   The `SubmitReport` presentation is created, and a presentation editor is opened with it.
7. Make sure the **Properties** window is open in Studio. It should be on the right side. If not, open it by clicking **Window ➤ Show View ➤ Properties** from the menu.
8. In the presentation, 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_text2`.
9. Set the *On Change Invoke* property to `Refresh()`.
   This will insure that the Total value is recalculated every time an amount is entered or changed.
10. Since the user should not enter a value for the Total, select the text field for the total value and set the *Editable* property to *No* simply by clicking on the property.
11. Save your changes.
12. You can preview what the presentation will look like. To do so, click on the HTML Preview () icon in the presentation editor.

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


Expense Report

SubmittedBy	
SubmitDate	1/1/2007
CostCenter	Sales
Description	
Items	

	Description	Date	Amount
1			

Total	0.00
Comments	

13. 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 drop-down list. Finally, the Total, which you set to not-editable (in effect, read-only), is shown as a label rather than a text field.

Creating a Screenflow

Presentations can be used directly from the process, but normally they are used by *screenflows*, which are a specialized kind of process for user interaction sequences. Screenflows are in turn called from the main process.

To create a screenflow:

1. 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.

2. In the **Implementation Type** section, select *Screenflow* from the drop-down list.

3. In the **Related Screenflow** section, click **New**.
The **New Screenflow** wizard appears.

4. In the **Name** field, enter `Submit Report`, and click **Next**.
The **Select Instance Variables** page of the wizard appears.

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

In	Out
No	No

   All others should be set to *No*.

6. Click **Next**.
   The **Screenflow out - Process in Argument Mapping** page appears.
7. Verify that this page has no entries, and click **Next**.
   The wizard finishes, and you should see the "Screenflow created successfully" message.
8. Click **Finish**.
   The wizard closes.
9. Now specify the location of where the screenflow will be placed in the **Project Navigator**. Click **OK** to accept the default location shown, under *Processes*.
10. 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 included, as when a new process is created.
11. In Instance section of the **Variables** window, add an instance variable named `reportSf`, of type ExpenseReport.

    This instance variable will hold an ExpenseReport object during the life of the screenflow. We added the `Sf` suffix to make it easier to identify as a screenflow instance variable, as opposed to the process instance variable `report`.

    Note that this is not an ALBPM convention and is not required.

12. Right-click on the `reportSf` instance variable and click **Map as process incoming argument** (), and then right-click again and click **Map as process outgoing argument** ().
    The `reportSf` instance variable will receive the value of the incoming argument `reportSfArg`, while the outgoing argument `reportSfArg` will take the value of the `reportSf` instance variable. You can check this by right-clicking on the Begin or End activities and clicking on **Argument Mapping**.
13. 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 still need to design the screenflow, and will do this in the next task.

Designing a Screenflow

You design a screenflow in a manner similar to a process, by adding activities and defining their properties. There are no roles in a screenflow.

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 box appears.
2. In the **Name** field, enter `Input Report`, and click **OK**.
   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 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 *reportSf*.
6. Select the **Use BPM Object Presentation** option, and select *SubmitReport* 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 box appears.

9. From the **Argument Set Names** list, select *InputReportOut*.
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

Both `action` and `result` are predefined instance variables. The action argument is an IF which will cancel the activity if the user clicks any button except the submit button to exit the submission form.

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

Your screenflow should now look like the following:

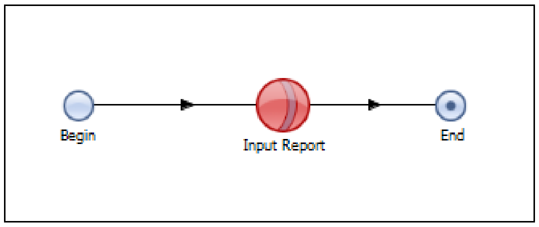


Figure 20: Input Report Screenflow

14. Close the screenflow editor.
15. You need to specify the argument mapping in 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.
16. Click **Argument Mapping**.
The **Argument Mapping** dialog box appears.
17. Add an entry to the *Submit Report In* page exactly as follows:

Submit Report's input arguments		Value
reportSfArg	=	ExpenseReport()

Here, `reportSfArg` is the incoming argument to the screenflow's Begin activity, while `ExpenseReport()` is the Expense Report object's constructor.

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 also use it 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 enter and 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 insures all of your project elements are saved.
2. Click the **Start Engine** icon () to start Studio's 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 ().

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 set to complete the process in Activity 4.

Activity 4: Adding Alternative Paths

You now have a good part of the functionality of the most expected path of the project. It is time to consider 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 desireable 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 where you will code a simple rule that will handle pre-defined approval and rejection criteria.

The intent of implementing company policy in an automatic activity is to lighten the workload on supervisors, who should not need to spend time on trivial cases.

The company rules are the following: Any expense report totaling less than $2,500 is accepted. Any expense report of less than $25 is also automatically approved and goes directly to the Treasurer. If the total is $2,500 or greater, the expense report is rejected.

To define the company policy:

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 box 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. You will note that you never declared the `result` variable. This is a predefined instance variable of type `String`, and is 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 which sets two parameters (a predefined instance variable and a BPM object attribute) according to the company policy for expense reports. The process flow itself is controlled by conditional transitions, as you will see starting with the next task.

Adding a Conditional Transition

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

To add a conditional transition:

1. In the Studio, go to the process design editor.
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 hard to see.
5. We will make this transition a curve. To be able to select it, first drag the Review Report activity down (we will move it back later).
   You should now be able to see the Accepted and Preapproved transition.
6. Click on the arrow of the transition and drag upwards. Note how it becomes an arc.

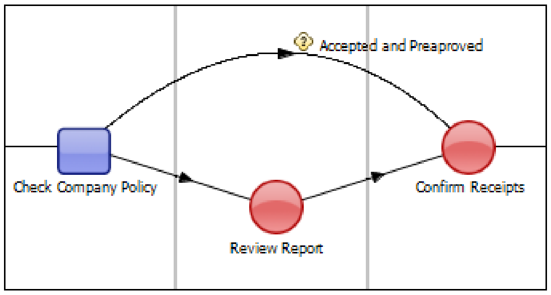


Figure 21: Accepted and Preapproved Transition

   At this point, you should see something like the above in your process diagram.
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, **Conditional** is selected, because this is a conditional transition.
10. In the text box for the conditional 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, which you worked on in the previous task.
11. Click **OK**.
    The expression is set.
12. Save your changes.

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

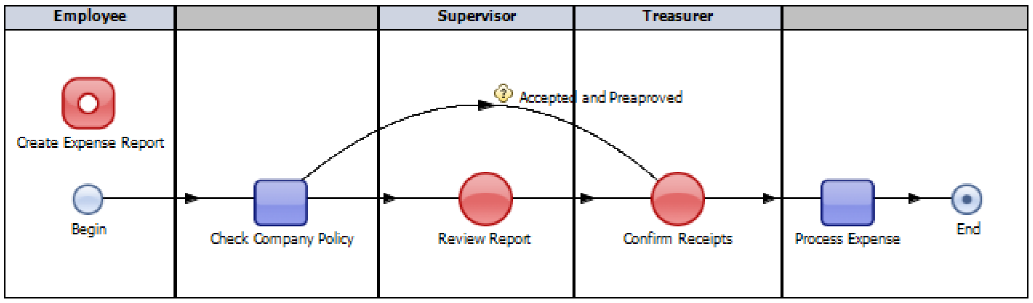


Figure 22: Expense Report process with Accepted and Preapproved transition.

Adding the Edit Report Activity

If an Expense Report is rejected either automatically or by another participant, we want the employee to be able to edit the report to correct mistakes or to add clarifying information. To allow the employee to do this, 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, except that in this case the employee must modify existing data rather than fill out a blank form. Therefore, the screenflow will also need to receive the data, and argument mapping will be required in both directions, as you will see in steps 8 and 9.

To add the Edit Report interactive activity:

1. In the process design editor window, right-click in the Employee role, somewhere below the Begin activity, and click **Add activity ➤ Interactive** (●).
   The **Activity** dialog box appears.
2. In the Name field, enter `Edit Report`, and click **OK**.
   The Edit Report interactive activity is added to the diagram.
3. Right-click on the Edit Report activity and click **Add unconditional transition**.
4. Click on 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.
5. 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 box appears.
6. Set the **Implementation Type** to *Screenflow*.
7. You will use an existing screenflow. In the **Related Screenflow** section, choose *Submit Report* from the **Name** drop-down list.

   You can re-use an existing screenflow, rather than create a new one, because the Employee participant must edit the same information he entered originally. No new fields are required.
8. Click **Argument Mapping**, and set the *Submit Report In* page to the following:

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

   Remember that to add a mapping pair to the list, you must click the Add icon (⊞).
9. Set the *Review Report Out* page to the following:

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

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

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

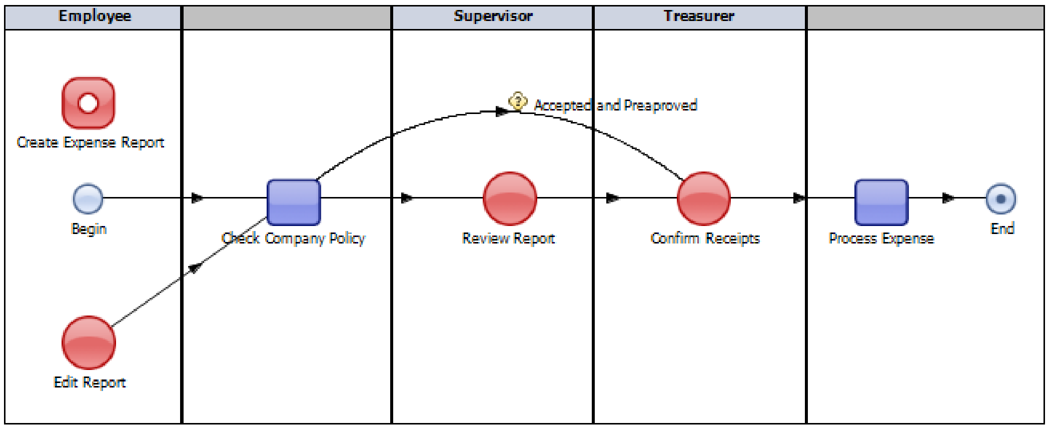


Figure 23: Expense Report process with Edit Report activity added.

Adding Conditional Transitions to Edit Report

You now have the Edit Report activity, but there is no path that will take the instance to it. In this task we add two conditional transitions which are are taken when the expense report is not accepted the Check Company Policy activity, or not approved at the Review Report activity.

To add the two conditional 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 page, enter the following in the conditional expression text box, capitalizing as shown:
   ```
   result != "Accepted"
   ```
5. Click **OK**.
   The Not Accepted conditional transition is added.
6. Drag the Not Accepted conditional transition so it becomes curved, as shown in the figure below. To be able to select the transtion, you may need to drag the uncoditional transition out of the way first.
   To straighten any curved transition, right-click on the transition and uncheck the **Curve** option.
7. Follow steps 1 through 5 to add a conditional 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:

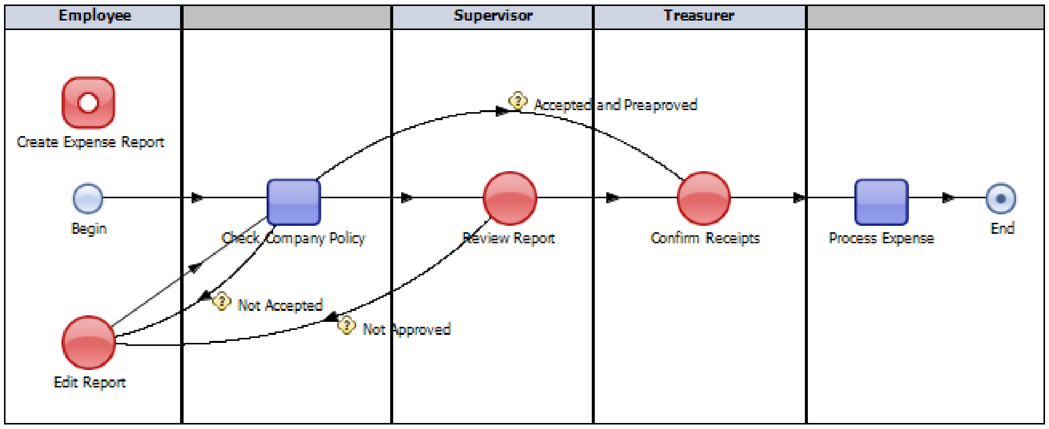


Figure 24: Expense Report process with additional conditional transitions

Defining the Review Report Presentation

We now have a conditional transition to be used in case the supervisor does not approve the expense report. However, we have not defined the Review Report activity. To do so, you will 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. Enter `ReviewReport` in the **Presentation** field.
3. The ReviewReport presentation will be 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 which will be shown as fields in the presentation. Select the attributes shown in the table below, in the order shown.

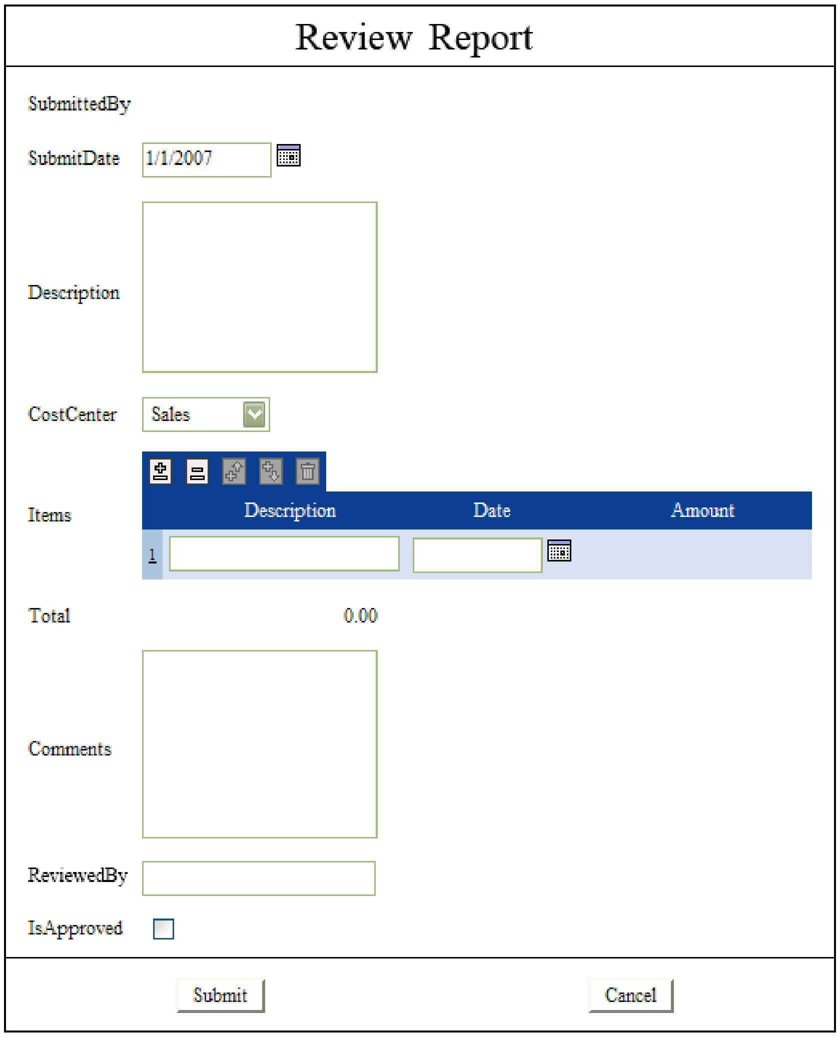
submittedBy
submitDate
description
costCenter
items[].description
items[].date
items[].amount
total
comments
reviewedBy
isApproved

   Note that these are the same attributes as 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 in Studio.
8. In the presentation, select the input text box in the *SubmittedBy* row, and set the *Editable* property to *No*, by clicking on the property. The name of this field should be `text0`.

   We set this so that the users of this presentation cannot change the employee name.
9. In the presentation, 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_text2`.
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 set the *Editable* property to *No*.
12. Save your changes.
13. To verify, click on the HTML Preview () icon in the presentation editor.

    The ReviewReport presentation should look like this:



Review Report

SubmittedBy	
SubmitDate	1/1/2007
Description	
CostCenter	Sales
Items	
Total	0.00
Comments	
ReviewedBy	
IsApproved	☐

Items:

	Description	Date	Amount
1			

Figure 25: 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

We now have a conditional transition to be used in case the supervisor does not approve the expense report. However, we have not defined the Review Report activity. You will define this activity as a screenflow similar to the Submit Report screenflow.

Since the report instance variable is already defined, the process will be simpler

To define the Review Report activity:

1. Right-click on the Review Report interactive 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 **New Screenflow** wizard 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 *No*.
6. Click **Next**.
   The wizard finishes with a "Screenflow created succesfully" message.
7. Click **Finish**, and then click **OK** to accept the default when you are asked to select a location for the new screenflow.
8. Back in the **Main Task** dialog box, click **Argument Mapping**.
   The **Argument Mapping** dialog box 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 now 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 Input Report interactive component call 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 box 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 **OK** again in the **Main Task** dialog box.

    Your screenflow should now look like the following:

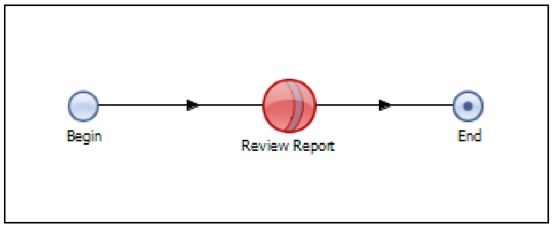


Figure 26: Review Report screenflow

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 has not confirmed one or more receipts.

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

You could place this loop in the conditional transition itself, but it's better if you define a method. In this way, you will be able to use the method elsewhere. If any changes are made to the way this information is stored in the BPM object, you will need to update only one piece of code.

Finally, the conditional expression in the transition itself will be 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 will use 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. Enter `areReceiptsChecked` in the **Method Name** field, 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
   ```

   This code will return `false` if any item has not been checked. If all items have been checked, it will return `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 will begin dragging a transition line originating from the Confirm Receipts activity.
7. Click on the Edit Report activity.
   The **Transition from Activity** dialog box will appear.
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`.
10. Click **OK**.
    The Not Confirmed conditional transition is added.
11. Save your changes.

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

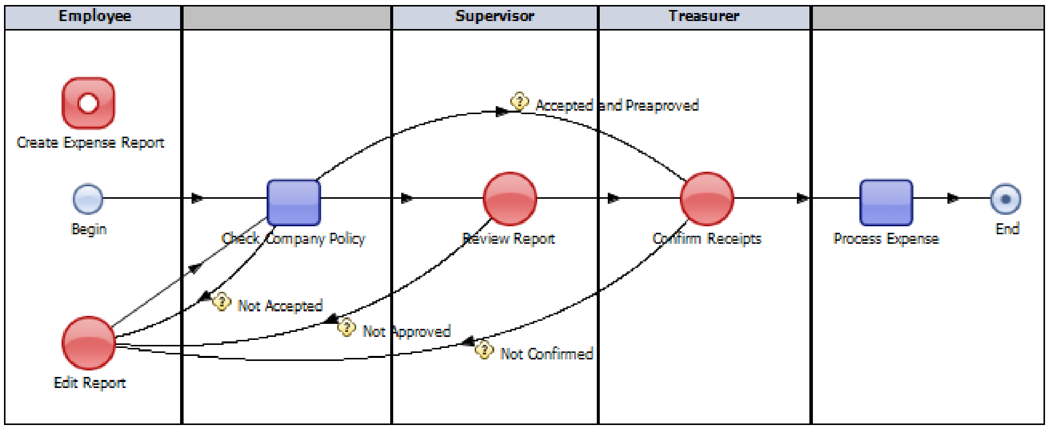


Figure 27: Expense Report process with Not Confirmed conditional transition

Creating the Confirm Receipts Presentation

You have not yet defined a task for the Confirm Receipts activity. As with the other interactive activities, Confirm Receipts will use a screenflow. As before, you must first define the presentation this screenflow will use.

The Confirm Receipts presentation is similar to the Review Report presentation, except that it also allows the treasurer to check off each confirmed receipt.

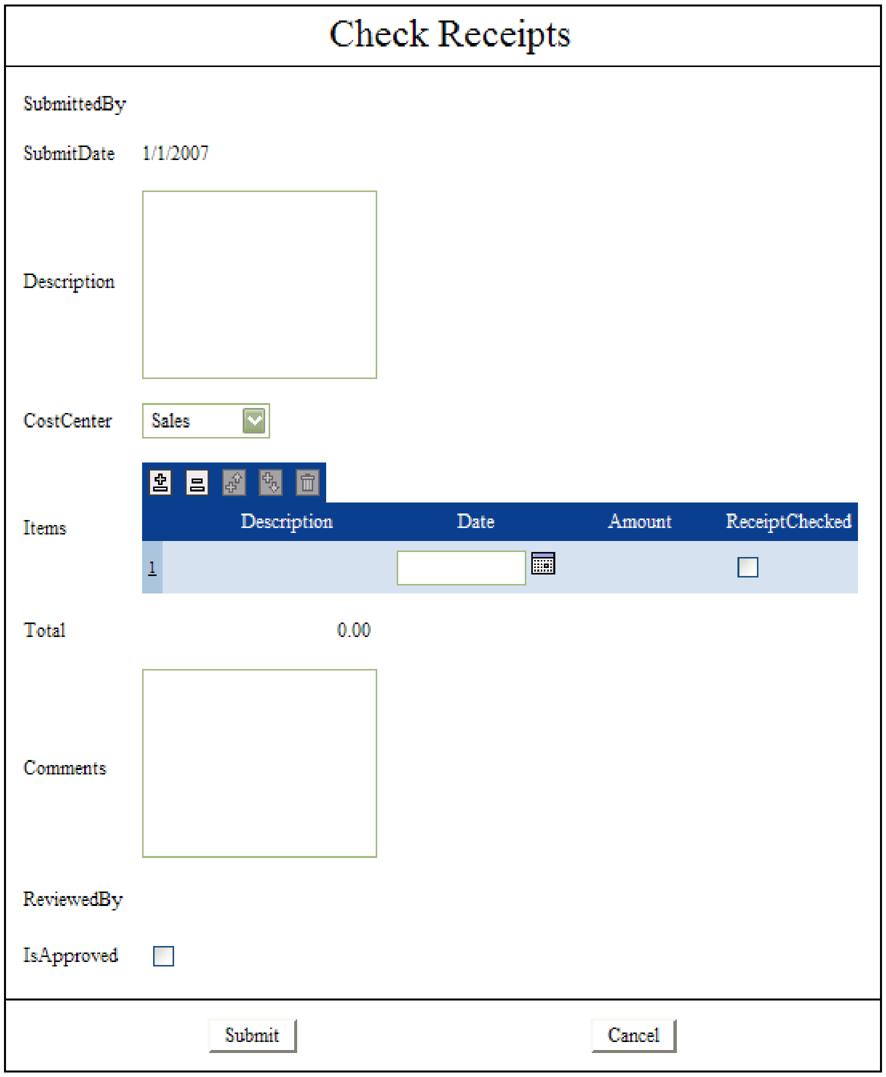
To create the Confirm Receipts presentation:

1. In the Project Navigator, right-click on the ExpenseReport BPM Object (), and click **New ➤ Presentation** ().
   The **Presentation Wizard** appears.
2. Enter `ConfirmReceipts` in the **Presentation** field.
3. 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 which will be shown 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
items[].receiptChecked
total
comments
reviewedBy
isApproved

   Note that these are the same attributes as you specified in the Submit Report presentation, plus the `items[].receiptChecked` attribute.
6. Click **Finish**.
   The `ConfirmReceipts` presentation is created, and a presentation editor is opened with it.
7. Using the **Properties** window, set the *Editable* property in the fields for the `submittedBy`, `submitDate`, `items[].description`, `items[].amount`, and `total` attributes to *No*.
8. Save your changes.
9. To verify, click on the HTML Preview () icon in the presentation editor.

   The Confirm Receipts presentation should look like this:


Check Receipts

SubmittedBy	
SubmitDate	1/1/2007
Description	
CostCenter	Sales
Items	

	Description	Date	Amount	ReceiptChecked
1				☐

Total	0.00
Comments	
ReviewedBy	
IsApproved	☐

Figure 28: Confirm Receipts presentation

10. Close the presentation editor.

Defining the Confirm Receipts Activity & Screenflow

We now have a conditional transition to be used in case one or more receipts are missing or incorrect. However, we have not defined the Confirm Receipts activity. You will define this 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 43, but substituting "Confirm Receipts" wherever it says "Review Report".
2. Follow the steps in *Designing the Review Report Screenflow* on page 44, and here also use "Confirm Receipts" instead of "Review Receipts".

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

Running the Process

The Expense Report process is now complete.

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. Click the **Launch WorkSpace** ( ) button.
4. In the WorkSpace login page, log in as `Peter Jones`.
5. Click on *Create Expense Report* in the WorkSpace **Applications** panel.
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 or less.
7. Log out of WorkSpace and log in again as Paul Smith, the participant with the supervisior 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 List** panel should have no 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 **WorkList** panel. You should see Confirm Receipts, in the *Activity* column.
9. Excecute the Confirm Receipts activity by clicking on the Execute icon ( ).
   The Check Receipts form opens in the browser. It should show the item you entered previously, while logged in as Peter Jones.
10. In the item line, check 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 fairly simple, several sequences are possible. Here you ran the instance through the "happier" path by submitting an expense report for less than $25. You should 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 supervisior 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.

Activity 5: Finishing Touches

The basic Expese 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 will add some improvements that will expose you to additional ALBPM features.

Adding a Due Transition

In this task we add the Send Reminder automatic activity, which will remind 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 will connect to it with a *due* transition.

The due transition fires as a function of the activity it is sourced from. You will add a due transition which will be taken when a certain amount of time has passed since an instance has arrived at the Edit Report activity.

To add the Send Reminder activity and Reminder Due transition:

1. Right-click in the space below the Edit Report activity and click **Add Activity** ➤ **Automatic** ().
   The **Activity** dialog box appears.
2. Enter `Send Reminder` in the **Name** field, and click **OK**.
   The Send Reminder automatic activity is added to the diagram.
3. Right-click on the Edit Report activity and click **Add due transition** ().
   The mouse pointer will begin dragging a transition line.
4. Click on the Send Reminder activity.
   The **Transition from Activity** dialog box appears.
5. Enter `Reminder` in the **Name** field.
6. In the Properties page, enter `'2m'` (complete with the single quotes) in the expression text box, and click **OK**.
   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.
7. Click and drag on the Reminder due transition to curve it towards the right.
8. Right-click on the Send Reminder activity and click **Add unconditional transition**.
9. Click on the Edit Report activity, and then drag the transition towards the left. See the diagram at the bottom.
   The unconditional transition is added.

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

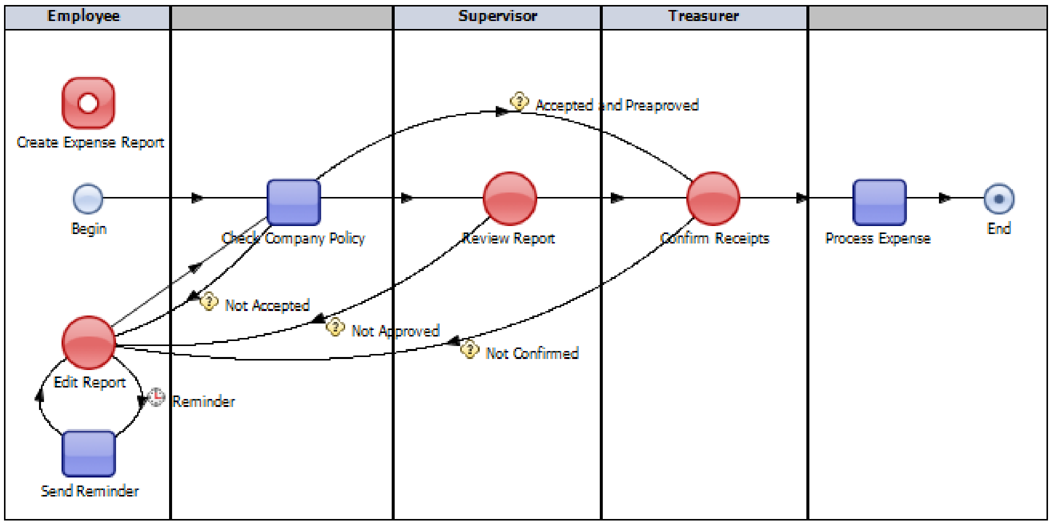


Figure 29: Expense Report process with Reminder Due transition

Defining the sendReminder Method

You now have the Send Reminder activity, but have not yet defined how it will work. You will set its Main Task as a method which will send an 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.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 will also need to configure the Studio engine so that it knows where to send e-mail. In the Project Navigator, right-click on the top of the ExpenseManagent project, and click **Engine Preferences** ( ).
   The Engine dialog box appears.
9. Enter the name of your SMTP mail server in the **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 which 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. Yet the process engine already knows the name of any participant because they had to log in, so this step should not be required.

In this task you will modify the ExpenseReport process so that the `submittedBy` participant is preset. You will also set up the process so that the `submitDate` attribute has a default value.

You will add an automatic activity to the Submit Report screenflow. In this activity, you will 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 box will appear.
4. Set the **Implementation Type** drop-down list to *Method*.
5. In the **Method** panel, click **New**.
   The **New Process Method** dialog box appears.
6. Enter `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:

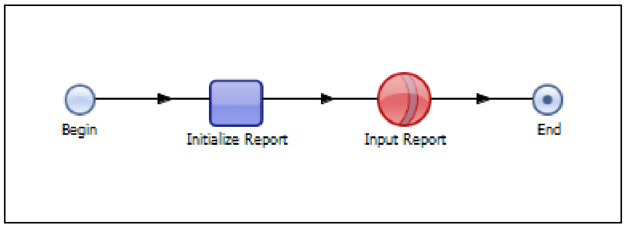


Figure 30: Submit Report screenflow with Initialize Report activity

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*. Do the same with the `reviewedBy` text field in the ReviewReport presentation.