WebSphere® Business Modeler Advanced

Version 6.0.2.1

Using Loops in a Process

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices and Trademarks" on page 31.

This edition applies to Version 6, Release 0, Modification 2, Fix Pack 1 of IBM® WebSphere Business Modeler Advanced (5724-I75) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send them to the following address:

IBM Canada Ltd. Laboratory
Information Development for WebSphere Business Modeler
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7

Include the title and order number of this book, and the page number or topic related to your comment.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Using loops in processes

Loops are repeating sequences of activities within a process. They enable you to repeat an exact sequence of activities as many times as you require within a process.

Loops are typically used to indicate a sequence of tasks or activities that must be performed a certain number of times before the process as a whole can continue. Many business processes require an action or sequence of actions to be repeated a number of times. A payment process will continue issuing checks as long as there are invoices in a particular database; a shipping process will add boxes to a shipping container until the container is full; a packaging process will add a specific number of items to a carton. Any activity that requires an exact repetition of activities can be modeled as a loop.

You can choose from three types of loops in WebSphere® Business Modeler:

For loop
: For loops repeat the same sequence of activities a specified number of times.

While loop
: While loops repeat the same sequence of activities as long as a certain condition is satisfied. While loops will never run if the specified condition is false from the start. The while loop is the only type of loop supported in WebSphere Process Server and WebSphere Business Integration Server Foundation.

Do-while loop
: Like while loops, do-while loops repeat the same sequence of activities as long as a certain condition is satisfied. Do-while loops are guaranteed to run at least once. The do-while loop is the only type of loop supported in WebSphere MQ Workflow.

In this tutorial, you will add an instance of each type of loop to a process. You will learn how to add conditions to your loops, and you will set up and run simulations based on the expressions and values that you specify.

Note: This tutorial is accompanied by a sample project that you can import into WebSphere Business Modeler. The sample contains final versions of each of the processes and loops described in the individual exercises. You can choose to complete each exercise, or you can review the sample to see the results of completing the exercise steps.

Before you begin

This tutorial does not provide instruction on the basic functions of WebSphere Business Modeler. It is assumed that you are familiar with the product and can complete basic tasks, such as adding modeling elements to the to the drawing area. For instructions on basic WebSphere Business Modeler functions, see the help topics in the information center included with the product.

Before completing the exercises in this tutorial, you must import the Loops sample into your WebSphere Business Modeler workspace. Download the sample from the WebSphere Business Modeler Advanced tutorials Web page (www.ibm.com/

software/integration/wbimodeler/library/tutorials.html), and then follow the instructions in product’s information center for importing a project.

Ensure that your modeling mode is set to Advanced before beginning the first exercise. To set your modeling mode, click **Modeling** → **Mode** → **Advanced**.

This tutorial should take a maximum of 30 minutes to complete.

For loops

A for loop is a loop that repeats a specific number of times. To use a for loop, you must set a counter (sometimes called an iterator) to tell the loop how many times to run the same sequence of activities. The counter has the following three numeric values:

- Initial counter value
- Increment (the amount to add to the counter each time the loop runs)
- Final counter value

The for loop ends when the counter reaches the final counter value, or, if there is an associated test condition, when the test condition is false.

Although the for loop is the simplest type of loop, you must be careful to use it only when you are repeating the exact same set of activities for a *specified number of instances*. You must know the number of times you need to repeat the activities, otherwise a for loop is not appropriate for your process. Business analysts need to avoid the common error of using a loop when what they are actually modeling is rework of a task, or a different set of tasks.

In this tutorial, you will add a for loop to handle some of the activities in a telemarketing company's call procedure. In the call procedure, the telemarketer contacts a particular customer 4 different times. When the telemarketer has attempted to reach the customer 4 times, no further attempts are made. The for-loop keeps a counter that records how many iterations have been performed. In this example, each iteration represents a call to the customer. The initial value of the counter is 0, indicating that the customer has not yet been called. The counter increment is 1, and the task is repeated until the total number of attempts equals 4.

For loops also support a test condition. In this same telemarketing scenario, you can add a condition so that when the customer answers one of the calls, no further calls will be made. When the test condition becomes true—that is, the customer answers—the loop stops. You can model the for loop as follows:

- Initial counter value: 0 calls
- For as long as the counter is not at the final value (3 calls)
  - Check that the customer has not answered a call
  - Make a call
  - Increment the counter (add 1)

The sequence of activities in the loop repeat until the customer has been called a total of 4 times, except that if the customer responds to any of the calls, the condition ("the customer has responded") becomes true and the loop stops.

Adding a for loop to a process

Before you can complete this exercise, you need to import the Loop sample from the WebSphere Business Modeler Web site. Remember to set your business modeling mode to Advanced by click **Modeling** → **Mode** → **Advanced**.

In the telemarketing scenario, a telemarketer attempts to contact a customer 4 times before removing that customer from the call list. The telemarketer repeats the same activities each time she calls the customer. In the Call Procedure process included in the Loop sample, the basic activities that the telemarketer must repeat are modeled, but no provision has been made to show that they must repeat a specified number of times.

It is a good practice for business analysts to make copies of their processes as they work, ensuring that they can refer to an earlier version if necessary. You will use that method in this tutorial. To show that part of this process must repeat before the process as a whole can complete, you will first make a copy of the Call Procedure process. In the new copy of the process, you will use a for loop to indicate the repeating activities.

To add a for loop to the Call Procedure process, complete the following steps:

1. If it is not already open, double-click the Call Procedure process. Notice that there is a Contact Customer task included in the process, but no provision has been made to repeat the task 3 times before moving to the next task in the process.
2. Make a copy of the Call Procedure process:
   a. In the Project Tree, right-click the Call Procedure process and select **Copy**.
   b. Right-click the Processes folder and select **Paste**. Rename the copy of the process to Call Procedure - For Loop.
3. Open the Call Procedure - For Loop process.
4. Delete the Contact Customer task.
5. In the palette, select the For Loop icon. Alternatively, you can right-click on the drawing area of the diagram and select **New** → **For Loop**.
6. Move your cursor to the point where the Contact Customer task was located and click to add a for loop to the diagram.
7. Rename the for loop to Contact Customer.
8. Connect the available Choose Script output to the Contact Customer for loop input. Similarly, connect the Contact Customer output to the available Update Information input.
9. Click **File** → **Save** to save the process.

By replacing the Contact Customer task with the Contact Customer for loop, you will be able to model the individual tasks that must be completed by the telemarketer when she contacts a customer. Note that you could also show these tasks with a local or global process, but the for loop gives you the capability of repeating these activities a predetermined number of times before the process moves on to the next activity.

Adding activities to a for loop

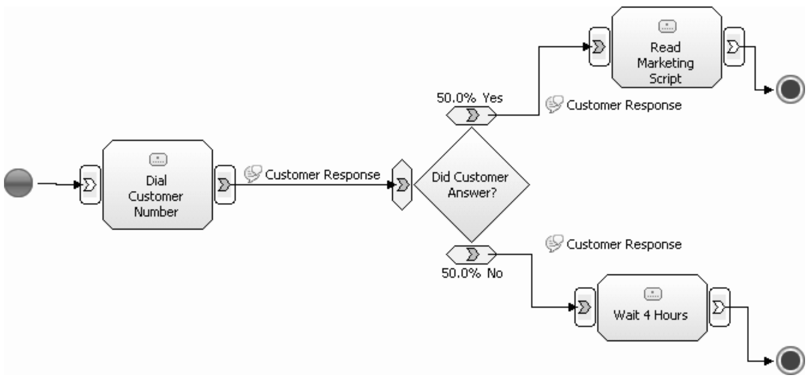
The for loop that you have added to the Call Procedure process replaces the single Contact Customer task. However, there is more than one activity involved in contacting a customer, and these activities must be repeated each time that the telemarketer attempts to reach the customer until the customer answers.

In the telemarketing procedure, the telemarketer first dials the customer number and waits for the customer to answer. If the customer answers, the telemarketer reads her marketing script; if the customer does not answer, the telemarketer indicates the time that she called, waits at least four business hours, and then attempts to contact the customer again. If the telemarketer is unable to make contact with the customer after 4 attempts, she updates the customer information to indicate that the customer is unreachable.

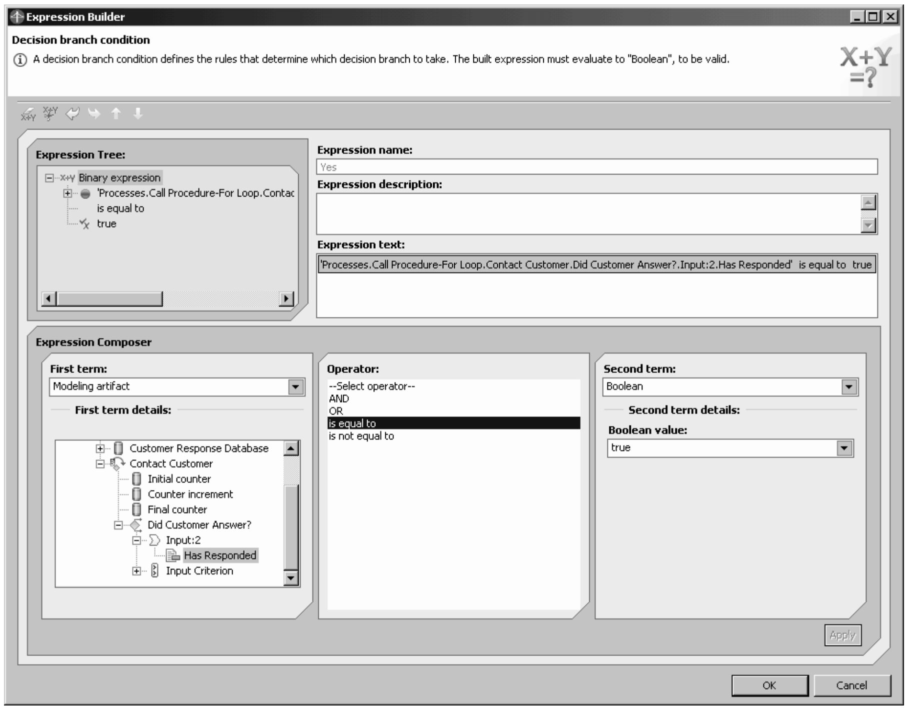
To add the individual tasks and activities that make up the for loop, complete the following steps:

1. In the Contact Customer for loop, click the plus sign ("+") to expand the for loop.
2. Add the activities that must be completed when contacting a customer, as follows:
   a. Dial Customer Number task
   b. Did Customer Answer simple decision
   c. Read Marketing Script task
   d. Wait 4 Hours task. (Select the Wait 4 Hours task, click the **Inputs** tab, and set the task duration to 4 hours.)
   e. Two stop nodes, one for each possible path
3. Connect the activities as follows:
   a. Connect the start node to the input of the Dial Customer Number task
   b. Connect the output of the Dial Customer Number task to the input of the Did Customer Answer simple decision.
   c. Connect the Yes branch of the Did Customer Answer decision to the input of the Read Marketing Script task.
   d. Connect the output of the Read Marketing Script task to the first stop node.
   e. Connect the No branch of the Did Customer Answer decision to the input of the Wait 4 Hours task.
   f. Connect the output of the Wait 4 Hours task to the second stop node.
4. Associate the Customer Response business item with connections in the loop:
   a. Right-click the connection between the Dial Customer Number task and the decision and select **Associate data**.
   b. Select the **Complex type** radio button and then select the Customer Response business item. Click **OK**.
   c. Using the same method, associate the Customer Response business item with the connection between the Yes decision output and the Read Marketing Script task, and the No decision output and the Wait 4 Hours task. In the telemarketing scenario, this data association simply means that the customer response, or lack of customer response, will determine which subsequent task is performed. In technical terms, you are now able to access and use the Customer Response business item attributes in any expression that you create on the decision when you simulate the process.

   The for loop should now look like the one in the following image:



5. Add an expression to the decision:
   a. Click the Did Customer Answer decision to select it.
   b. Click the **Output branches** tab.
   c. In the table, click the row which contains the Yes output branch.
   d. In the Decision Branch Condition section, locate the **Expression** field and click **Edit**. Note that the Decision Branch Condition section is at the bottom of the page; you might have to scroll to locate it.
   e. In the **First term** list, ensure that **Modeling artifact** is selected.
   f. In the **Modeling artifact** tree, click **Processes → Call Procedure-For Loop → Contact Customer → Did Customer Answer? → Input → Has Responded**. This attribute has been predefined in the Customer Response business item.
   g. In the **Operator** list, select **is equal to**.
   h. In the **Second term** list, select **Boolean**. The Has Responded attribute has Boolean as its type, and so you must select a Boolean value to associate with the Yes branch of the decision.
   i. In the **Boolean** value field, select **true**.
   j. Click **Apply**. The Expression Builder should look like the following image:


Expression Builder

Decision branch condition

A decision branch condition defines the rules that determine which decision branch to take. The built expression must evaluate to "Boolean", to be valid.

Expression Tree:
- Binary expression
  - 'Processes.Call Procedure-For Loop.Contac
  - is equal to
  - true

Expression name: Yes

Expression description:

Expression text: 'Processes.Call Procedure-For Loop.Contact Customer.Did Customer Answer?.Input:2.Has Responded' is equal to true

Expression Composer

First term: Modeling artifact

First term details:
- Customer Response Database
- Contact Customer
  - Initial counter
  - Counter increment
  - Final counter
  - Did Customer Answer?
    - Input:2
      - Has Responded
    - Input Criterion

Operator: --Select operator-- / AND / OR / is equal to / is not equal to

Second term: Boolean

Second term details:

Boolean value: true

6. Click **OK** to close the Expression Builder. Note that when you add an expression to one branch of a simple decision, the opposite expression is automatically added to the other branch. If you click the table row containing the No decision output, you will notice that the **Expression** field has been automatically updated to include an expression where the Has Responded attribute is not equal to true. This saves you having to use the Expression Builder to compose an expression for the No output.
7. Click **File → Save** to save the process.

Review the activities that you have added to the Contact Customer for loop. First, the telemarketer dials the customer's number. If the customer does not answer, the Wait 4 Hours task indicates when the telemarketer can call the same customer again. The sequence of tasks will repeat until the customer answers, or until the maximum number of attempts is reached. If the customer answers before the maximum number of attempts, the telemarketer reads the marketing script, and the loop ends.

You also added an expression to the decision within the loop. You can use this expression to run a simulation of the Call Procedure-For Loop process later in this tutorial.

Now that you have added the required elements and expression to the Contact Customer for loop, you need to enable the loop to accept data and to pass data back to the parent process.

Passing data to a for loop

In the previous exercise, you added the required activities to your Contact Customer for loop. However, you still need to enable the for loop to receive incoming data, and to produce corresponding outgoing data. Remember that the Contact Customer for loop is one activity in a larger parent process, and it needs to be able to pass data and control on to the next activity so that the larger process can finish.

You need to take note of one other important characteristic of the Call Procedure-For Loop process. The process contains two local repositories: Customer Record Database and Customer Response Database. These local repositories contain the data that is passed into and out of the Contact Customer for loop. A local repository is required any time that you want to pass data from a parent process to a loop. Similarly, if you want to pass data out of a loop to the parent process, you need to pass that data to a local repository in the parent process. In the telemarketing example, the Customer Record Database local repository holds the Customer Record that the activities within the loop will be working with. The first task in the for loop, Dial Customer Number, retrieves the Customer Record from the repository and passes it to the other activities that need it. You can model this by adding an input to the Dial Customer Number task and associating that input with the Customer Record Database. To pass data to that repository from the loop, you need to add an output to the final task in the loop, Read Marketing Script.

This concept can be confusing to anyone new to loops because a local repository is not directly connected to a loop in the parent process diagram. Instead, there is a virtual connection between the repository and the specified task input or output inside the loop. You need a repository in the process because it allows one iteration of the loop to store information in a shared location that a subsequent iteration can access. To run an expression-based simulation of this process, you must use a local repository to pass data to tasks within the loop (and pass data out of a loop). Later in this tutorial, you will set the loop condition by using the Expression Builder. Because the Expression Builder cannot access a global repository, a loop condition always relies on a local repository that is changed from within the loop. Any loop inputs with associated data can be used only in the loop condition.

You will add inputs and outputs that read from or write to a database in the same way that you add inputs and outputs to any task, but you must specify the local repositories that will send and receive the data. To enable the Contact Customer for loop to receive and output data, complete the following steps:

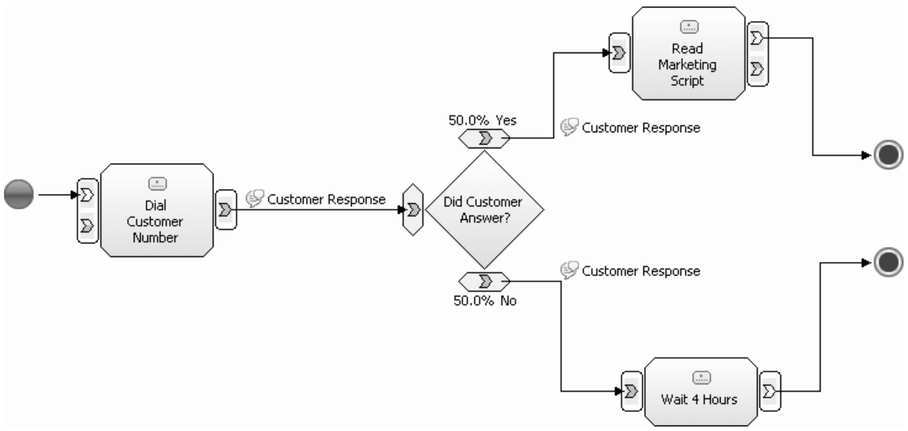
1. In the Contact Customer for loop, click the Dial Customer Number task.
2. In the Attributes view, click the **Inputs** tab.

3. Add an input to the task and set the attributes:
   a. Click **Add** next to the Input table.
   b. In the Associated data column, select **Customer Record**. The Customer Record business item contains the customer’s telephone number, which is required to complete the Dial Customer Number task.
   c. Leave the Minimum and Maximum columns with the default values.
   d. In the Input source column, select **Repository**.
   e. Press Enter on your keyboard. More settings are displayed in the Details section below the Input table.
   f. Click **Browse** next to the **Repository value** field.
   g. Select the **Local repository** radio button, and then select **Call Procedure - For Loop/Customer Record Database** from the menu.
   h. Click **OK**.
   i. Ensure that the **Read** radio button is selected. By selecting this button, you are indicating that the customer record will be removed from the Customer Record Database after it is received by the Contact Customer for loop.

   You have now added an input that will enable your Contact Customer for loop to receive a Customer Record.
4. In the diagram, click the Read Marketing Script task.
5. In the Attributes view, click the **Outputs** tab.
6. Add an output to the task and set the attributes:
   a. Click **Add** next to the Output table.
   b. In the Associated data column, select **Customer Response**. The Customer Response business item indicates that required to complete the activities in the Contact Customer for loop.
   c. Leave the Minimum and Maximum columns with the default values.
   d. In the Input source column, select **Repository**.
   e. Press Enter on your keyboard. More settings are displayed in the Details section below the Input table.
   f. Click **Browse** next to the **Repository value** field.
   g. Select the **Local repository** radio button, and then select **Call Procedure - For Loop/Customer Response Database** from the menu.
   h. Click **OK**.
   i. Leave the **Overwrite** radio button selected. By selecting this button, you are indicating that the customer record will be added to the Customer Response Database after it is received from the Contact Customer for loop.

   You have now added an output that will enable your Contact Customer for loop to pass on a Customer Response business item.

You have now added an output that will enable your Contact Customer for loop to pass on a Customer Response business item. The Contact Customer for loop should now look like the one in the following image:



Before moving on to the next exercise, be sure that you understand the exercise that you have just completed. Although there is no direct connection line in the diagram from the two local repositories in the parent process, data can be passed between the repositories and the loop because of the input and output that you have just added. Without the local repositories, you would not be able to run an expression-based simulation of the Call Procedure-For Loop process.

Setting a condition on a for loop

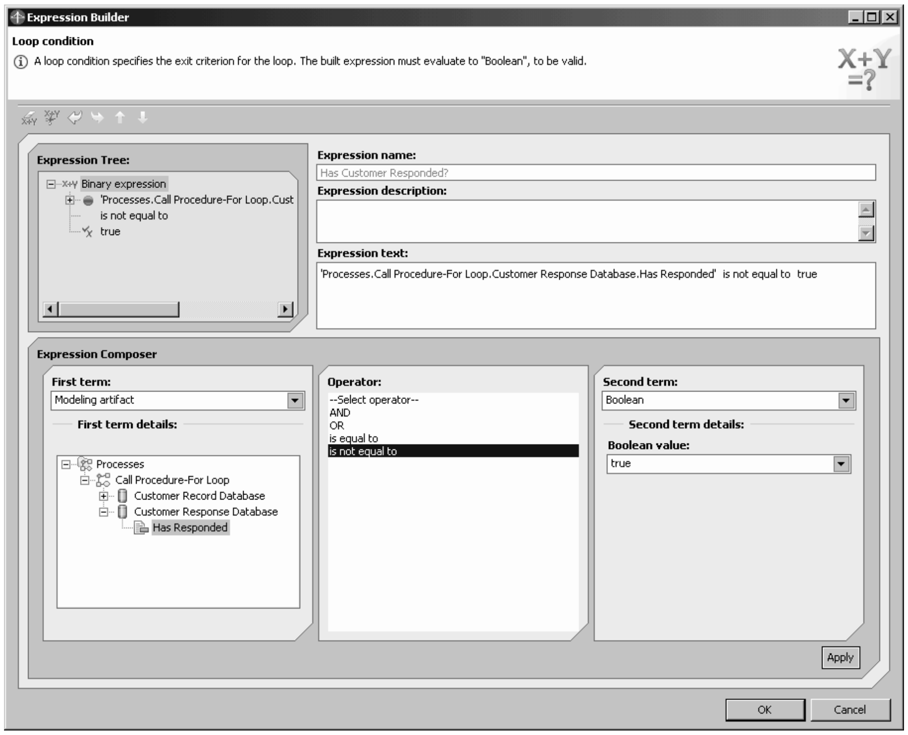
Before you can run a simulation of the Call Procedure-For Loop process, you must first set the loop condition. The loop condition includes the initial setting, which is normally 0 or 1; the final counter, which is the total number of times that you want to perform the activities in the loop, and the counter increment, which is the number added to the counter each time the loop runs. You might assume that the counter increment would always be 1, but consider the example of a packager who loads cartons onto a truck. The packager might load the cartons two at a time, or he might first put the cartons onto a pallet and load them 10 at a time. In these cases, the counter increment would be set to a number other than 1.

In the telemarketing scenario, the telemarketer makes 1 call to the customer at a time (the counter increment) and will call that single customer a maximum of 4 times. If the customer answers before the fourth call, that customer is not called again (an additional condition).

To set the loop condition for the Contact Customer for loop, complete the following steps:

1. Ensure that no activity is selected in the process diagram. In the Attributes view, click the **Loop Condition** tab.
2. Enter the values for the loop counter:
   - **Initial counter**: 0
   - **Final counter**: 3
   - **Counter increment**: 1
3. Click the **Additional condition** check box.
4. In the **Name** field, type the following: `Has the Customer Responded?` This is the name of the expression that you will create to define the condition.
5. Optionally, you can add a description of the condition; for example, "Check to determine if the customer has already responded to a marketing call", or whatever is appropriate.
6. Click **Edit** below the **Expression** field. The Expression Builder opens.

7. Use the Expression Builder to define the additional condition that, if met, will end the loop:
   a. In the **First term** list, ensure that **Modeling artifact** is selected.
   b. In the **Modeling artifact** tree, click **Processes** → **Call Procedure-For Loop** → **Customer Response Database** → **Has Responded**. This attribute has been predefined in the Customer Response business item, and will be used to indicate whether the customer has responded to a telemarketing call.
   c. In the **Operator** list, select **is not equal to**.
   d. In the **Second term** list, select **Boolean**. The Has Responded attribute has Boolean as its type, and so you must select a Boolean value to correctly compose your expression.
   e. In the **Boolean value** field, select **true**.
   f. Click **Apply**. Notice that the Expression Tree and Expression Text panes are automatically populated with the expression that you have just created. The Expression Builder should look like the following image:



8. Click **OK** to close the Expression Builder.

By creating this expression, you have added a condition to the Contact Customer for loop: As long as the Has Responded attribute on the Customer Response business item is not equal to ″true″, the loop will continue until it reaches its final counter value. In terms of the telemarketing scenario, as long as the customer has not answered the telemarketer’s call, the telemarketer will continue to call until she has made a total of 4 attempts.

Now that you have set the loop condition and enabled your loop to receive data, you are ready to set up and run an expression-based simulation of the Call Procedure-For Loop process.

Simulating a process containing a for loop

You can use different methods to simulate a process. Probability-based simulation enables you to set up and run a simulation more quickly. Probabilities require relatively little data setup in a model and are sufficient to perform most simulations, except when very detailed low-level analysis is required. Typically, you can use probability-based simulation for current state modeling and "what if?" analysis.

Alternatively, you can use expressions for low-level future state business models and for models that clearly define the interfaces in and out of tasks and other elements, such as decisions. When you set up an expression-based simulation, you model how specific data will be treated as it passes through the process. For example, an order may be handled differently depending on the customer type, or on the total cost of the order. In an expression-based simulation, you define business item creation rules that determine the specifics of each order that the process handles, and you set expressions on decisions and on activities that evaluate the incoming order and handle it according to the specific details it contains. Expressions are also a good way to document in precise structured language what you know about conditions, or business rules in the model.

For the purposes of this tutorial, expression-based simulations are also a convenient way to observe how a loop functions. To run an expression-based simulation for the Call Procedure-For Loop process, you must first set an expression on a business item attribute. In the telemarketing scenario, the Customer Response business item contains a Has Responded attribute. This attribute is a Boolean, meaning that it is either true or false. The telemarketing company can simulate the loop by defining the percentages of true and false responses resulting from their marketing calls.

Note: The objective of this tutorial is not to produce an elaborate set of simulation preferences and conditions, but rather to provide an understanding of how loops function. There are many modeling elements that you could add to the Loops Sample to make the simulation more complex: resources, costs, timetables, and durations can all be modeled to produce detailed results, and various types of static and dynamic analysis could subsequently be performed to examine the model in detail. If you want to create more complex simulations, use the Loops Sample as a starting point and add as many other modeling elements as you want. After you have a good understanding of how loops work, you are encouraged to use them in more detailed scenarios.

To create an expression-based simulation for the Call Procedure-For Loop process, complete the following steps:

1. In the Project Tree, right-click the Call Procedure-For Loop process and select Simulate. The simulation diagram opens.
2. In the Attributes view, click the **General** tab.
3. In the **Method of selecting an output path** field, select **Based on an expression**. This method enables the simulation to evaluate any expression that it encounters in the process path. Remember that you previously set an expression on the decision in the Contact Customer for loop.
4. Click the **Inputs** tab. In the table, click the row containing the input.
5. Click **Edit** beside the **Total number of tokens** field.

6. Ensure that the **Generate a specific number of tokens in each simulation run** radio button is selected and type 5 for the value. By setting this value, you are indicating that 5 customers will be called at least one time.
7. Click **OK**.
8. In the **Recurring time interval for bundle creation** field, enter a time interval of 1 day. This delay between tokens will make it easier for you to differentiate between process runs if one is still progressing through the loop when the next one begins.
9. Right-click in the diagram and select Expand All from the pop-up menu. (Ensure that no process activity is selected and that your cursor is over a blank area in the diagram when you right-click; if it is over a process activity, you will see a different pop-up menu.) The Contact Customer for loop is expanded so that its component activities are displayed.
10. In the simulation diagram, click the first task inside the for loop, Dial Customer Number. This task outputs a Customer Response business item.
11. Create simulation values for the Customer Response business item:
    a. Click the **Business Item Creation** tab.
    b. In the table, select the row containing the Output and then click **Create Simulation Values**.
    c. In the Simulation value creation window, select **Define custom rules** from the list. The attributes of the Customer Response business item—which is the output of the Dial Customer Number task—are displayed.
    d. In the Business Item Attributes table, click Has Responded to select the attribute.
    e. In the **Attribute value distributions** list, select **Weighted list**, and then click **Add** at the bottom of the table.
    f. In the **New value** list, select **True** and then click **OK**.
    g. In the Probability column, click the cell next to the value that you just added and type `60`.
    h. Click **Add** again, select **False** as the value, and type `40` for the probability.
    i. Press Enter on your keyboard, and then click **OK** to close the window.
12. Click **File** → **Save** to save the simulation settings.

You are now ready to simulate the Call Procedure-For Loop process. To be able observe the behavior within the Contact Customer for loop, adjust the zoom settings for the simulation diagram and ensure that the simulation is set to display animation. (To adjust the simulation settings, click the Simulation Control Panel tab, click the Menu button, and select Setting.) When you have finished updating the settings, click the Run simulation ▶ button to run the simulation.

As the simulation runs, observe how the loop behaves. The loop runs at least once for each token; if the decision response is Yes, the token is passed to the Read Marketing Script task and that instance of the loop ends; if the decision response is No, the token is passed to the Wait 4 Hours task, and then is sent through the loop again to repeat the activities. If the loop counter for a particular token reaches 4, that instance of the loop ends.

You can continue to experiment with the process, increasing the number of tokens, adding costs to particular tasks, increasing durations, or changing the probabilities for the True and False attributes. You can also add or remove tasks within the loop according to your own organization's processes. Continue adjusting the process and simulation to become familiar with aspects of the for loop.

Do-while loops

A do-while loop is a loop that repeats a set of activities while some condition is satisfied. A do-while loop differs from a for loop in that you cannot specify the number of times that the loop will run; the number of times that the loop runs is determined dynamically when the process is run. A do-while loop contains a condition, and the loop tests its condition at the end of its component activities. This means that its sequence of activities will always run at least once.

In the telemarketing example, you have seen how a for loop can be used to model a situation where a telemarketer contacts a customer a specified number of times. However, if there is no restriction placed on the number of contact attempts, a do-while loop is the appropriate type of loop to use in your model. The do-while loop will repeat as long as a certain condition is satisfied. In the telemarketing scenario, the condition is that there is a customer record in the call queue. If the customer does not answer the call, the process waits for a specified amount of time and then the telemarketer attempts to contact the customer again. The customer record goes back into the queue continually until the customer responds, regardless of how many attempts it takes. This process can be modeled as a do-while loop as follows:

- Do: call the customer
- While: the customer has not responded
  - return to Do
- Update the Customer Responses database

The activity following the While condition—that is, ″return to Do″—continues until the condition is no longer true.

Remember that, as with the for loop, the only information that can both be accessed within the loop condition and be changed from within the loop is data stored in a local repository, so a well-formed loop condition always relies on a local repository.

Adding a do-while loop to a process

Do-while loops are ideal for modeling a repeated set of activities for an unspecified number of times. You have already used a for loop in the Call Procedure process to repeat the Contact Customer activities 3 times. But what if the telemarketer has no restrictions on the number of times that she attempts to contact the customer? If there is no predetermined number of times that the telemarketer will repeat the activities, a for loop is not the appropriate model element to use. In this case, a do-while loop will repeat the Contact Customer activities until the customer responds.

The do-while loop in the telemarketing scenario is added in the same way as the for loop. First, you need to make a copy of the process as a whole, and then you can add the do-while loop and its component tasks.

Tip: The tasks that you will add to the Contact Customer do-while loop are identical to the ones that you added to the for loop earlier in this tutorial. Although you are encouraged to complete all the steps of this tutorial in order to become more familiar with the components and characteristics of loops, the Call Procedure-Do While (Working) process included in the Loops Sample already contains the do-while loop and component model elements. If you choose to do so, you can open that process and skip ahead to “Setting a condition on a do-while loop” on page 18

To model the Call Procedure process using a do-while loop, complete the following steps:

1. Make a copy of the Call Procedure process:
   a. In the Project Tree, right-click the Call Procedure process and select **Copy**.
   b. Right-click the Processes folder and select **Paste**. Rename the copy of the process to Call Procedure - Do-While Loop.
2. Open the Call Procedure - Do-While Loop process.
3. Delete the Contact Customer task.
4. In the palette, select the Do-While Loop icon. Alternatively, you can right-click on the drawing area of the diagram and select **New** → **Do-While Loop**.
5. Move your cursor to the point where the Contact Customer task was located and click to add a for loop to the diagram.
6. Rename the do-while loop to Contact Customer.
7. Connect the available Choose Script output to the Contact Customer do-while loop input. Similarly, connect the Contact Customer output to the available Update Information input.
8. Click **File** → **Save** to save the process.

By adding a do-while loop to the Call Procedure process, you can model the series of activities that will be repeated until the customer answers the telemarketer’s call. Next, you will add the activities for the do-while loop, just as you did with the for loop.

Adding activities to a do-while loop

Because the do-while loop is similar to the for loop in the telemarketing scenario, you will use the same activities within the loop. The only difference will be in the conditions that you set on the loop in the next exercise.

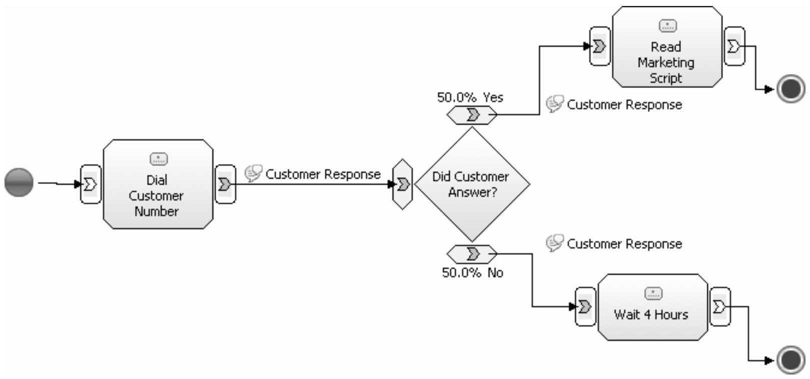
Tip: Although you are encouraged to complete all the steps of this tutorial in order to become more familiar with the components and characteristics of loops, some of the steps that you completed when adding activities to the for loop are repeated here. If you are comfortable with adding activities to a loop and adding inputs and outputs to the loop’s component activities, the Call Procedure-Do While (Working) process included in the Loops Sample has been set up so that you do not have to repeat those steps. If you choose to do so, you can open that process and skip ahead to “Setting a condition on a do-while loop” on page 18.

To add the individual tasks and activities that make up the do-while loop, complete the following steps:

1. In the Contact Customer do-while loop, click the plus sign (⌂+⌂) to expand the do-while loop.
2. Add the activities that must be completed when contacting a customer, as follows:
   a. Dial Customer Number task

   b. Did Customer Answer simple decision
   c. Read Marketing Script task
   d. Wait 4 Hours task. (Select the Wait 4 Hours task, click the **Duration** tab, and set the processing time to 4 hours.)
   e. Two stop nodes, one for each possible path
3. Connect the activities as follows:
   a. Connect the start node to the input of the Dial Customer Number task
   b. Connect the output of the Dial Customer Number task to the input of the Did Customer Answer simple decision.
   c. Connect the Yes branch of the Did Customer Answer decision to the input of the Read Marketing Script task.
   d. Connect the output of the Read Marketing Script task to the first stop node.
   e. Connect the No branch of the Did Customer Answer decision to the input of the Wait 4 Hours task.
   f. Connect the output of the Wait 4 Hours task to the second stop node.
4. Associate the Customer Response business item with connections in the loop:
   a. Right-click the connection between the Dial Customer Number task and the decision and select **Associate data**.
   b. Select the **Complex type** radio button and then select the Customer Response business item. Click **OK**.
   c. Using the same method, associate the Customer Response business item with the connection between the Yes decision output and the Read Marketing Script task, and the No decision output and the Wait 4 Hours task. In the telemarketing scenario, this data association simply means that the customer response, or lack of customer response, will determine which subsequent task is performed. In technical terms, you are now able to access and use the Customer Response business item attributes in any expression that you create when you simulate the process.

      The do-while loop should now look like the one in the following image:



5. Add an expression to the decision:
   a. Click the Did Customer Answer decision to select it.
   b. Click the **Output branches** tab.
   c. In the table, click the row which contains the Yes output branch.
   d. In the Decision Branch Condition section, locate the **Expression** field and click **Edit**. Note that the Decision Branch Condition section is at the bottom of the page; you might have to scroll to locate it.
   e. In the **First term** list, ensure that **Modeling artifact** is selected.

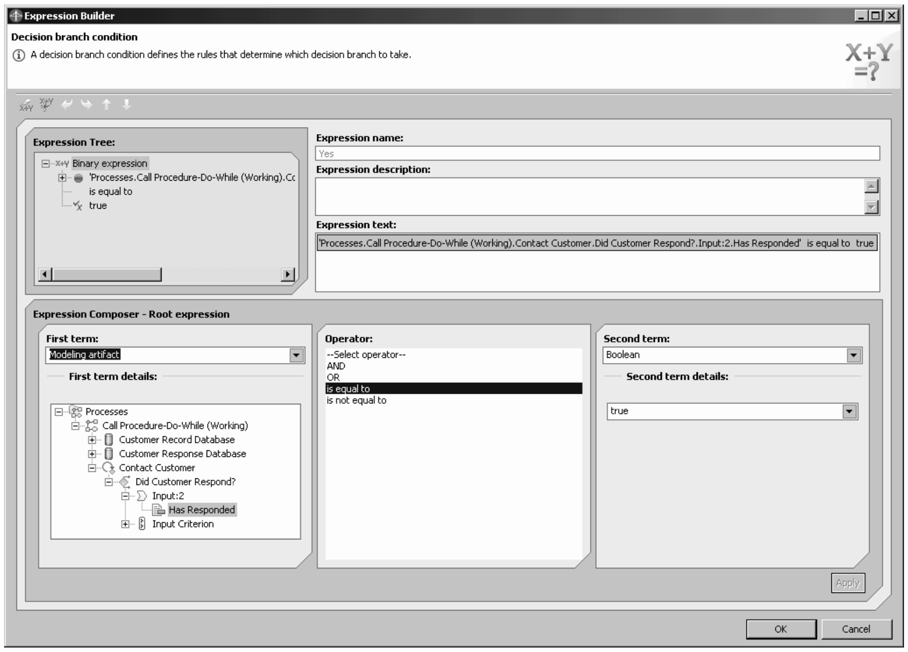
f. In the **Modeling artifact** tree, click **Processes** → **Call Procedure-Do-While Loop** → **Contact Customer** → **Did Customer Answer?** → **Input** → **Has Responded**. This attribute has been predefined in the Customer Response business item.

g. In the **Operator** list, select **is equal to**.

h. In the **Second term** list, select **Boolean**. The Has Responded attribute has Boolean as its type, and so you must select a Boolean value to associate with the Yes branch of the decision.

i. In the **Boolean value** field, select **true**.

j. Click **Apply**. The Expression Builder should look like the following image:



Expression Builder

Decision branch condition

A decision branch condition defines the rules that determine which decision branch to take.

Expression Tree:

Binary expression
'Processes.Call Procedure-Do-While (Working).Cc
is equal to
true

Expression name: Yes

Expression description:

Expression text: 'Processes.Call Procedure-Do-While (Working).Contact Customer.Did Customer Respond?.Input:2.Has Responded' is equal to true

Expression Composer - Root expression

First term: Modeling artifact

First term details:

Processes
Call Procedure-Do-While (Working)
Customer Record Database
Customer Response Database
Contact Customer
Did Customer Respond?
Input:2
Has Responded
Input Criterion

Operator:
--Select operator--
AND
OR
is equal to
is not equal to

Second term: Boolean

Second term details: true

6. Click **OK** to close the Expression Builder. Note that when you add an expression to one branch of a simple decision, the opposite expression is automatically added to the other branch. If you click the table row containing the No decision output, you will notice that the **Expression** field has been automatically updated to include an expression where the Has Responded attribute is *not* equal to true. This saves you having to use the Expression Builder to compose an expression for the No output.
7. Click **File** → **Save** to save the process.

Now that you have added the required elements to the do-while loop, you must add an input to the Dial Customer Number task and an output to the Read Marketing Script task, just as you did for the for loop earlier in this tutorial. This will enable the loop to accept and produce data from the local repositories in the parent process.

Passing data to a do-while loop

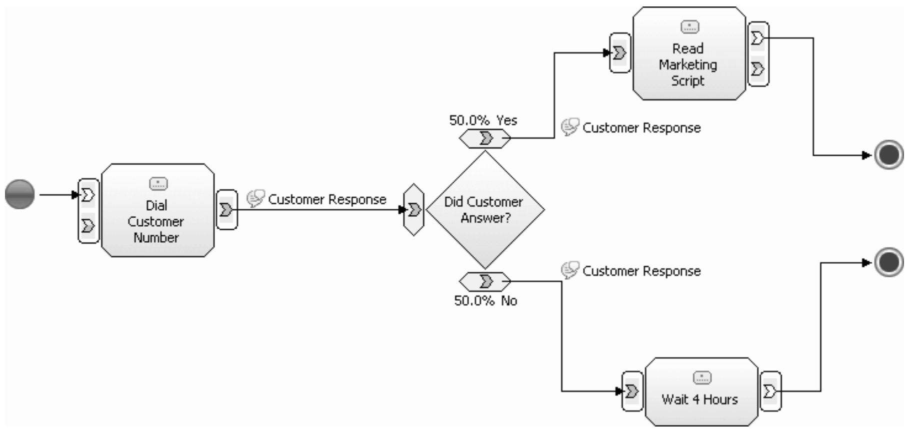
Remember that if you want to perform an expression-based simulation for your process, you need to use a local repository to pass data to and from the loop. Just as you did when you created a for loop, you need to enable your do-while loop to receive data from and pass data to a local repository. You will then be able to access this data using the Expression Builder and set the condition for the Contact Customer do-while loop.

Tip: Although you are encouraged to complete all the steps of this tutorial in order to become more familiar with the components and characteristics of loops, the Call Procedure-Do While (Working) process included in the Loops Sample already contains the do-while loop and component model elements. If you choose to do so, you can open that process and skip ahead to *"Setting a condition on a do-while loop" on page 18*.

To enable the Contact Customer do-while loop to receive and produce data, complete the following steps:

1. In the Contact Customer do-while loop, click the Dial Customer Number task.
2. In the Attributes view, click the **Inputs** tab.
3. Add an input to the task and set the attributes:
   a. Click **Add** next to the Input table.
   b. In the Associated data column, select **Customer Record**. The Customer Record business item contains the customer's telephone number, which is required to complete the Dial Customer Number task.
   c. Leave the Minimum and Maximum columns with the default values.
   d. In the Input source column, select **Repository**.
   e. Press Enter on your keyboard. Additional settings are displayed in the Details section below the Input table.
   f. Click **Browse** next to the **Repository value** field.
   g. Select the **Local repository** radio button, and then select **Call Procedure-Do-While Loop/Customer Record Database** from the menu. Click **OK**.
   h. Ensure that the **Read** radio button is selected. You do not want to remove the customer record from the database because if the customer does not answer, the record will have to be accessed again.
4. In the diagram, click the Read Marketing Script task.
5. In the Attributes view, click the **Outputs** tab.
6. Add an output to the task and set the attributes:
   a. Click **Add** next to the Output table.
   b. In the Associated data column, select **Customer Response**. The Customer Response business item is required to complete the activities in the Contact Customer do-while loop.
   c. Leave the Minimum and Maximum columns with the default values.
   d. In the Input source column, select **Repository**.
   e. Press Enter on your keyboard. More settings are displayed in the Details section below the Input table.
   f. Click **Browse** next to the **Repository value** field.
   g. Select the **Local repository** radio button, and then select **Call Procedure-Do-While Loop/Customer Response Database** from the menu. Click **OK**.
   h. Leave the **Overwrite** radio button selected. By selecting this button, you are indicating that the customer record will be added to the Customer Response Database after it is received from the Contact Customer do-while loop.

      You have now added an output that will enable your Contact Customer do-while loop to pass on a Customer Response business item. The Contact Customer for loop should now look like the one in the following image:



7. Click **File** → **Save** to save the process. You have now added an output that will enable your Contact Customer do-while loop to pass on a Customer Response business item.

This is the second time that you have added an input and output to enable your loop to receive and produce data. Just as with the for loop, there is no direct connection line in the diagram from the two local repositories in the parent process. Data can be passed between the repositories and the loop only because of the input and output that you have just added. Without the local repositories, you would not be able to run an expression-based simulation of the Call Procedure-Do-While Loop process.

Setting a condition on a do-while loop

A do-while loop has no limit on the number of times it will run. Remember that when you set the condition for the for loop, you added parameters for a loop counter. There is no such counter for a do-while loop. To set the loop condition for the Contact Customer do-while loop, you will not set an initial value or final value because you are not specifying the number of times that the telemarketer will contact the customer. Instead, you will simply add an expression that the loop will check each time it runs.

Important: Remember that a do-while loop checks its condition at the *end* of the loop. Because of this behavior, a do-while loop will always run at least once.

To set the loop condition for the Contact Customer do-while loop, complete the following steps:

1. In the parent process, click the Contact Customer do-while loop to select it. (To get to the parent process from within the loop, right-click and select **Parent**.)
2. In the Attributes view, click the **Loop Condition** tab.
3. In the **Name** field, type the following: `Has the Customer Responded?` This is the name of the expression that you will use to define the condition.
4. Optionally, you can add a description of the condition; for example, ″Check to determine if the customer has already responded to a marketing call″, or whatever is appropriate.
5. Leave the **Probability (%)** field blank. You will use an expression to determine the path flow in your simulation.
6. Click **Edit** below the **Expression** field. The Expression Builder opens.
7. Use the Expression Builder to define the condition that will enable the loop to continue:

a. In the **First term** list, ensure that **Modeling artifact** is selected.

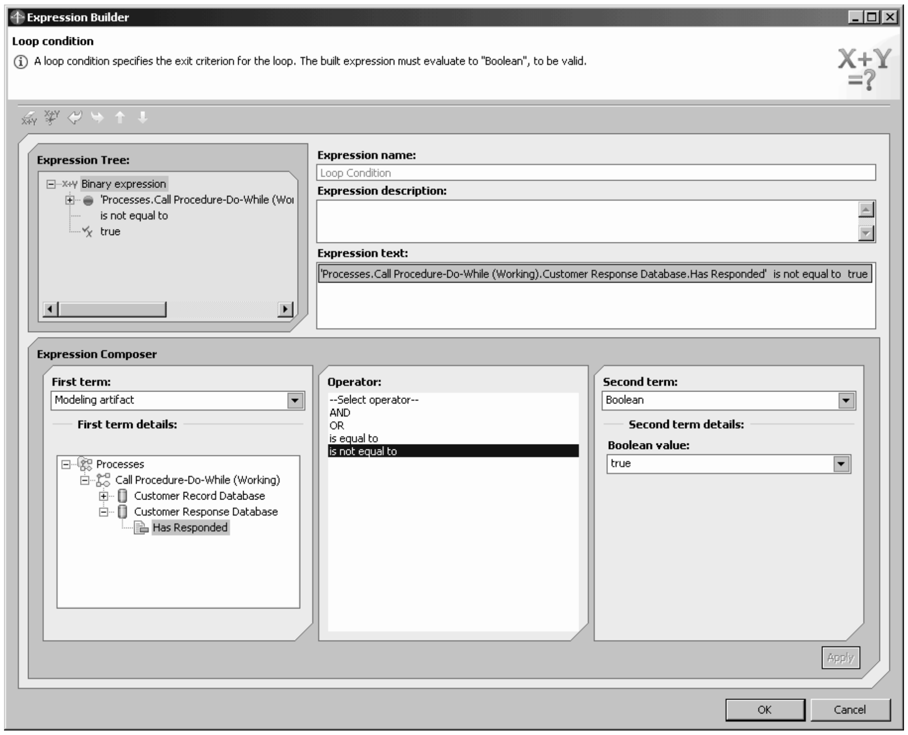
b. In the **Modeling artifact** tree, click **Processes** → **Call Procedure-Do-While Loop** → **Customer Response Database** → **Has Responded**. This attribute has been predefined in the Customer Response business item, and will be used to indicate whether the customer has responded to a telemarketing call.

c. In the **Operator** list, select **is not equal to**.

d. In the **Second term** list, select **Boolean**. The Has Responded attribute has Boolean as its type, and so you must select a Boolean value to correctly compose your expression.

e. In the **Boolean value** field, select **true**.

f. Click **Apply**. Notice that the Expression Tree and Expression Text panes are automatically populated with the expression that you have just created. The Expression Builder should now look like the following image:



Loop condition

A loop condition specifies the exit criterion for the loop. The built expression must evaluate to "Boolean", to be valid.

Expression Tree:

- Binary expression
  - 'Processes.Call Procedure-Do-While (Wor
  - is not equal to
  - true

Expression name: Loop Condition

Expression description:

Expression text: 'Processes.Call Procedure-Do-While (Working).Customer Response Database.Has Responded' is not equal to true

Expression Composer

First term: Modeling artifact

First term details:

- Processes
  - Call Procedure-Do-While (Working)
    - Customer Record Database
    - Customer Response Database
      - Has Responded

Operator:

--Select operator--
AND
OR
is equal to
is not equal to

Second term: Boolean

Second term details:

Boolean value: true

8. Click **OK** to close the Expression Builder.

By creating this expression, you have added a condition to the Contact Customer do-while loop: As long as the Has Responded attribute on the Customer Response business item is not equal to ″true″, the loop will continue. In terms of the telemarketing scenario, as long as the customer has not answered the telemarketer’s call, the telemarketer will continue to call every 4 hours. Clearly, this makes for a rather persistent telemarketer, especially if you do not create a timetable that specifies the times that the telemarketer could call. (Presumably, you would restrict your telemarketers from calling in the middle of the night, unless you are really trying to make an impression.) For the purposes of this tutorial, you do not need to add any more modeling elements to the Loops Sample project. However, if you were modeling a similar scenario, you might indicate the resources who would carry out the tasks, their associated costs, and the times that they are available to work. It should be apparent that do-while loops must be used carefully in your business models.

Now that you have set the loop condition and enabled your loop to receive data, you are ready to set up and run an expression-based simulation of the Call Procedure-Do While Loop process.

Simulating a process containing a do-while loop

Just as you did for the process containing a for loop, you will run an expression-based simulation for the Call Procedure-Do-While Loop process. You will set up and run the simulation of your do-while loop the same way that you ran the for loop simulation, creating an expression based on the Customer Response business item. Ensure that your simulation settings are set to display animation so that you can view the inner workings of the do-while loop:

To set up and run the simulation, complete the following steps:

1. In the Project Tree, right-click the Call Procedure-Do-While Loop process and select **Simulate**. The simulation diagram opens.
2. In the Attributes view, click the **General** tab.
3. In the **Method of selecting an output path** field, select **Based on an expression**. This method enables the simulation to evaluate any expression that it encounters in the process path. Remember that you previously set an expression in your loop condition.
4. Click the **Inputs** tab.
5. In the table, click the row containing the input.
6. Click **Edit** beside the **Total number of tokens** field.
7. Ensure that the **Generate a specific number of tokens in each simulation run** radio button is selected and type 5 for the value. Click **OK**.
8. Right-click in the diagram and select **Expand All** from the pop-up menu. (Ensure that your cursor is over a blank area in the diagram when you right-click; if it is over a process activity, you will see a different menu.) The Contact Customer do-while loop is expanded so that its component activities are displayed.
9. In the process diagram, click the first task inside the do-while loop, Dial Customer Number. This task outputs a Customer Response business item.
10. Create simulation values for the Customer Response business item:
    a. Click the **Business Item Creation** tab.
    b. In the table, select the row containing the Output and then click **Create Simulation Values**.
    c. In the Simulation value creation window, select **Define custom rules** from the list. The attributes of the Customer Response business item—which is the output of the Dial Customer Number task—is displayed.
    d. In the Business Item Attributes table, click Has Responded to select the attribute.
    e. In the **Attribute value distributions** list, select **Weighted list**, and then click **Add** at the bottom of the table.
    f. In the **New value** list, select **True** and then click **OK**.
    g. In the Probability column, click the cell next to the value that you just added and type `60`.
    h. Click **Add** again, select **False** as the value, and type `40` for the probability.
    i. Press Enter on your keyboard, and then click **OK** to close the window.
11. Click **File** → **Save** to save the simulation settings.

You are now ready to simulate the Call Procedure-Do-While Loop process. To be able to observe the behavior within the Contact Customer do-while loop, adjust the zoom settings for the simulation diagram and ensure that the simulation is set to display animation. (To adjust the simulation settings, click the **Simulation Control Panel** tab, click the **Menu** button, and select **Setting**.) When you have finished updating the settings, click the **Run simulation** ▶ button to run the simulation.

As the simulation runs, observe how the loop behaves. The loop runs at least once for each token; if the decision response is Yes, the token is passed to the Read Marketing Script task and that instance of the loop ends; if the decision response is No, the token is passed to the Wait 4 Hours task, and then is sent through the loop again to repeat the activities. Unlike the for loop that you simulated earlier in the tutorial, there is no limit on the number of times that a telemarketer can attempt to contact a customer, which means that in the simulation there is no limit on the number of times that a token can be passed back to the start of the loop. Run the simulation several times to observe how the number of task activations within the loop can differ between simulations.

You can continue to experiment with the process, increasing the number of tokens, adding costs to particular tasks, increasing durations, or changing the probabilities for the True and False attributes. You can also add or remove tasks within the loop according to your own organization's processes. Continue adjusting the process and simulation to become familiar with aspects of the do-while loop.

While loops

A while loop is a type of loop that repeats as long as a specified condition is satisfied.

The while loop tests its condition at the beginning of every loop. If the condition is false from the start, the sequence of activities contained in the loop never runs at all. In the telemarketing example, a telemarketer contacts customers, updates the customer information if the customer responds, and waits a certain period of time to call again if the customer does not respond. As long as there are customer records in the queue, the telemarketer contacts that customer. If there are no customer records for the telemarketer to access at the beginning of the process, or if all the customer records have been distributed, the condition is no longer true, and the telemarketer stops contacting customers. This example can be modeled as follows:

- While (there is a customer record in the queue) contact the customer

The loop condition is set using the Expression Builder. As with the two other types of loop you have worked with in this tutorial, the while loop can only have data passed to it from a local repository, and similarly, can only pass data to a local repository.

In the telemarketing scenario, customer records need to be allocated so that telemarketers have the information they need to contact the customer. To model this process, you will use a while loop, with the condition being that there must be a customer record in the Customer Record Database. To populate the Customer Record Database you will use a for loop.

Adding a while loop to a process

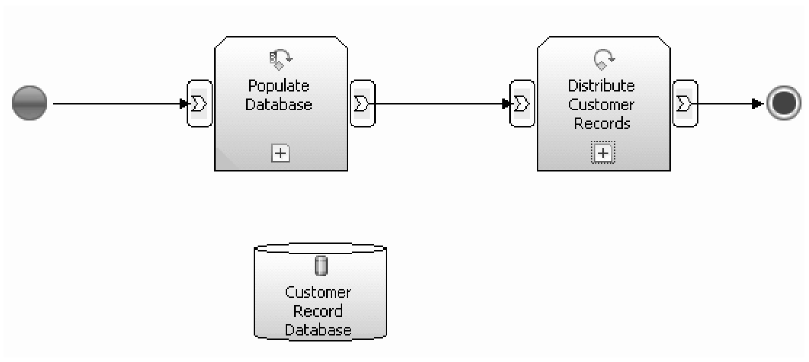
In the Loops Sample, the Distribute Call List-While Loop process has already been set up to populate the Customer Records Database local repository using a for loop. This approach will enable you to simulate the process without having to create a large number of Customer Record business item instances. Expand the Populate Database for loop to see the simple task that is used for populating the Customer Record Database with business items. The for loop should be familiar to you after completing the exercises earlier in this tutorial.

Note: This tutorial has been designed to ensure that the while loop will run when you simulate it. The Populate Database for loop ensures that a business item instance will be added to the Customer Record Database local repository so that you can observe the function of the Distribute Customer Records while loop. In a typical while loop scenario, there will be no guarantee that the while loop will run even once, which is why this particular type of loop would be used instead of a for loop or a do-while loop.

To add a while loop to the process, complete the following steps:

1. In the Project Tree, double-click the Distribute Call List-While Loop to open it in the process editor.

2. In the palette, select the While Loop icon and then click in the diagram to the right of the Populate Database for loop. (Alternatively, you can right-click on the drawing area of the diagram and select **New** → **While Loop**.)
3. Rename the while loop to Distribute Customer Records.
4. Delete the connection between the Populate Database for loop and the stop node.
5. Connect the Populate Database for loop to the Distribute Customer Records while loop, and then connect the Distribute Customer Records while loop to the stop node. The diagram should now look like the one in the following image:



Now that you have added the Distribute Customer Records while loop to the process, you need to add the Distribute Customer Records while loop activities.

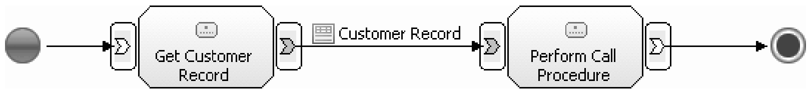
Adding activities to a while loop

Similar to the two other types of loops, while loops can contain as many activities as required. In the telemarketing scenario, the Distribute Customer Records while loop contains just two tasks. First, the customer records are received from the Customer Records Database, and then the call procedure is performed by any assigned telemarketer. In this example, the focus is on creating loops. However, you should keep in mind that you can also model the allocation of resources (for example, assigning individual telemarketers to one or more tasks), the scheduling of those resources (using timetables), or any other business situation.

To add the required tasks to the Distribute Customer Records while loop, complete the following steps:

1. In the Distribute Customer Records while loop, click the plus sign (″+″) to expand the loop.
2. Add the following model elements:
   a. Get Customer Record task
   b. Perform Call Procedure task
   c. Stop node
3. Connect the elements of while loop as follows:
   a. Connect the start node to the Get Customer Record task.
   b. Connect the Get Customer Record task to the Perform Call Procedure task.
   c. Connect the Perform Call Procedure task to the stop node.
4. Right-click the connection between the Get Customer Record task and the Perform Call Procedure task and select **Associate data**.

5. Select the Customer Record complex type and then click **OK**. The Customer Record business item is now associated with this connection, and can be passed between the two tasks. The Distribute Customer Records while loop should now look like the one in the following image:



Now that you have added the loop activities, you must add an input to the Get Customer Record task to enable it to receive data from the Customer Record Database local repository.

Passing data to a while loop

Just as you did for the previous two loops that you have created, you must enable your while loop to receive data from a local repository, which will in turn enable you to simulate your process based on an expression. To enable the Distribute Customer Records while loop to receive data, complete the following steps:

1. In the Distribute Customer Records while loop, click the Get Customer Record task.
2. In the Attributes view, click the **Inputs** tab.
3. Add an input to the task and set the attributes:
   a. Click **Add** next to the Input table.
   b. In the Associated data column, select **Customer Record**.
   c. Leave the Minimum and Maximum columns with the default values.
   d. In the Input source column, select **Repository**.
   e. Press Enter on your keyboard. Additional settings are displayed in the Details section below the Input table.
   f. Click **Browse** next to the **Repository value** field.
   g. Select the **Local repository** radio button, and then select **Distribute Call List-While Loop/Customer Record Database** from the menu.
   h. Click **OK**.
   i. Ensure that the **Read and Remove** radio button is selected. By selecting this setting, you are indicating that the customer record will be removed from the database after it is distributed.
   j. Leave the **Read from the beginning** radio button selected.
4. Click **File** → **Save** to save the process.

The Distribute Customer Records while loop is now ready to receive data from the Customer Records Database local repository.

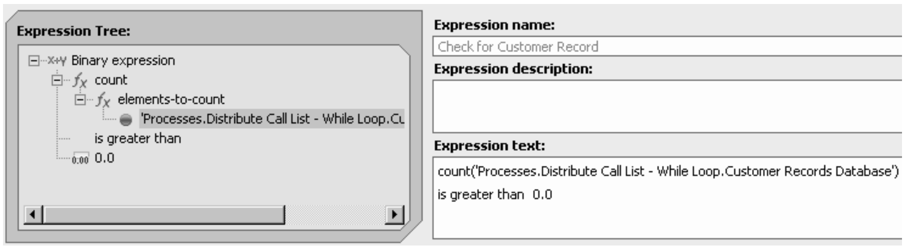
Setting a condition on a while loop

The do-while and for loops that you created earlier in this tutorial will each run at least once. In the case of the for loop, it will run a maximum of 4 times for each process instance. The do-while loop will run until the customer responds, but it too will run a minimum number of times; that is, once. The while loop differs from the other two types of loop in that it tests its condition before it runs for the first time. If the condition is not met, the loop will not run even once.

In the telemarketing example, the Distribute Customer Records while loop will run as long as there are records in the Customer Record Database. The loop needs to check the count of the records in the database to determine if it will run. If the count is greater than 0, the loop will run. To check the database for waiting records, you will need to create an expression that uses a function.

The while loop must determine if there are any customer records in the database. To do this, you need to define an expression that uses a function for one of its terms. To set the condition for the Distribute Customer Records while loop, complete the following steps:

1. In the parent process diagram, click the Distribute Customer Records while loop to select it.
2. In the Attributes view, click the **Loop condition** tab.
3. In the **Name** field, type the following: `Check for Customer Record`. This is the name of the expression that you will create to define the condition.
4. Optionally, you can add a description of the condition; for example, "Check to determine if there are any waiting records in the Customer Records Database", or whatever is appropriate.
5. Click **Edit** below the **Expression** field. The Expression Builder opens.
6. Use the Expression Builder to define the condition that the loop will check to determine if it should run:
   a. In the **First term** list, select **Function**.
   b. In the **Operator** list, select **is greater than**.
   c. In the **Second term** list, select **Number**. Leave the number value as `0.0`.
   d. Now you need to edit the function for the first term of the expression. Click **Edit** in the **Function expression** area of the First term pane.
   e. In the Available functions tree, expand the **Numeric functions** node and select **count**. In the Function description pane, click the **elements-to-count** link. The Expression Composer area now enables you to select the collection of elements that you want to count.
   f. Ensure that **Modeling artifact** is selected in the **First term** field.
   g. In the Modeling artifact tree, select the Customer Record Database local repository.
   h. Click **Apply**. Notice that the Expression Tree and Expression Text panes are automatically populated with the expression that you have just created. The Expression Tree and Expression Text panes should look like the following image:



Expression Tree:

```
⊟ x+y Binary expression
   ⊟ fx count
      ⊟ fx elements-to-count
            'Processes.Distribute Call List - While Loop.Cu
      is greater than
   0.00 0.0
```

Expression name: Check for Customer Record

Expression description:

Expression text:

```
count('Processes.Distribute Call List - While Loop.Customer Records Database')
is greater than  0.0
```

   i. Click **OK** to complete the expression. The **Expression** field is populated with the expression that you just created.
7. Click **File** → **Save** to save the process.

It is important that you understand the steps that you have just completed. The expression that you defined as your loop condition uses a function to determine the number of customer records in the Customer Record Database local repository. The expression compares that number to the value that you added as the second term in the expression, 0. If the count is higher than 0, the loop will run. In this way, the loop determines whether to run *before* the start of its first component activity. This behavior distinguishes the while loop from the other two types of loops.

Now that you have created an expression for your Distribute Customer Records while loop, you can run an expression-based simulation for the Distribute Call List process.

Simulating a process containing a while loop

In this exercise, you will set up and run an expression-based simulation, just as you did previously for the processes containing the for loop and do-while loop. Remember that this simulation requires business items to be created before they can be passed to the Distribute Customer Records while loop. Two different business item instances, Pittsburgh Customers and Hartford Customers, have been created for you to use in this simulation. A typical organization might have many more business item instances defined for a business scenario. Remember, too, that this tutorial uses business item instances in this exercise to demonstrate an alternative method of providing data to the simulation. You could also use the business item creation method to generate data for this simulation.

To simulate the Distribute Call List-While Loop process, complete the following steps:

1. In the Project Tree, right-click the Distribute Call List-While Loop process and select **Simulate**. The simulation diagram opens.
2. In the Attributes view, click the **General** tab.
3. In the **Method of selecting an output path** field, select **Based on an expression**. This method enables the simulation to evaluate any expression that it encounters in the process path. Remember that you previously set an expression containing a count function in the Distribute Customer Records loop condition.
4. Right-click in the diagram and select **Expand All** from the pop-up menu. (Ensure that your cursor is over a blank area in the diagram when you right-click; if it is over a process activity, you will see a different menu.) The two loops are expanded so that their component activities are displayed.
5. Set the business item creation attributes in the for loop so that the local repository can be populated with data:
   a. In the Populate Database for loop, click the Store Customer Record task.
   b. Click the **Business Item Creation** tab.
   c. In the table, select the row containing the Output and then click **Create Simulation Values**.
   d. In the **Value creation rule** field, select **Random List**.
   e. Click **Add**. Select the Pittsburgh Customers business item instance, and then click **OK** Add the Hartford Customers business item instance using the same method. For the purposes of this tutorial, only two business item instances are available. A typical organization would normally have a larger number of business item instances to distribute.
6. Click **File → Save** to save the simulation settings.

You are now ready to simulate the Distribute Call List-While Loop process. To be able to observe the behavior within the two loops, adjust the zoom settings for the simulation diagram and ensure that the simulation is set to display animation. (To adjust the simulation settings, click the **Simulation Control Panel** tab, click the

Menu button, and select **Setting**.) When you have finished updating the settings, click the **Run simulation** button ▶ to run the simulation.

Remember that the Populate Database for loop is a construct specifically for this simulation. The for loop's counter has a maximum of 5, and therefore 5 business item instances are passed to the Distribute Customer Records while loop. The Distribute Customer Records while loop runs as long as its while condition is true; that is, as long as there is at least one customer record in the Customer Record Database local repository.

This is the final exercise of this tutorial.

Summary

A loop is a useful modeling element when used correctly to show activities that are repeated within a larger process. However, it is important to review your process to ensure that you require an activity to be repeated exactly; it is a common business modeling error to use a loop to repeat an activity or sequence of activities when what the process actually requires is a similar but separate task.

In this tutorial, you have learned how to use three different types of loops in your business models. You should now be familiar with the differences between the three types of loops, and how to set loop conditions. You should also know how to use local repositories and task inputs and outputs to pass data between parent processes and loops. Understanding this repository/loop relationship is crucial if you want to run an expression-based simulation of a process that contains a loop.

Focus on the following key points when using loops:
- Be sure that a loop is required in the process.
- Choose the right type of loop for your repeated activities.
- Include a local repository in the process if you want to run an expression-based simulation.
- Add a task input or output for any data that you want to pass between the loop and the parent process.
- Ensure that you have set an appropriate loop condition.

As you continue to work with loops in your own processes, remember that you can add various modeling elements to a loop's component tasks, just as you would for any task in the parent process. You can add costs, resources, durations, or timetables, and produce simulations based on any number of business scenarios. By using loops in your processes, you can model any repeated activity realistically and accurately.

Notices and Trademarks

Notices

Trademarks and service marks

IBM WebSphere Business Modeler Advanced Version 6.0.2.1

IBM WebSphere Business Modeler Basic Version 6.0.2.1