



## Tutorial 3: Complex Types, If and ForEach Activities

---

### CONTENTS

Opening the WSDL Editor  
Using the Schema Editor,  
and complex schema types  
Changing schema types  
from simple to complex  
Verifying the type change  
Handling no input  
Using an If activity  
Counting nodes in a node  
set  
Clearing the output  
Using a For Each activity  
Looping over each node in  
a node set  
Creating a concatenate  
expression  
Assigning the final output  
Deploying and testing the  
process  
Reference image of the  
completed process

This tutorial assumes that you have completed the first and second tutorials and builds upon that knowledge.

In this tutorial, you create a process that accepts an unlimited number of strings, then puts them all together in a single string with a space between each. You will test the process by entering four names. To accept an unlimited number of strings, you will have to create a complex type for the process input. This tutorial is also an introduction to other activities, such as For Each, and If, that you will use on a regular basis.

Prerequisites:

- Micro Focus Verastream Process Design Studio
- An installed and running Micro Focus Verastream Process Server
- Internet browser
- Experience using the XPath and Copy Rule Editors from previous tutorials
- Some familiarity with XML Schema, WSDL, XPath, BPEL, and Web service standards

Let's get started.

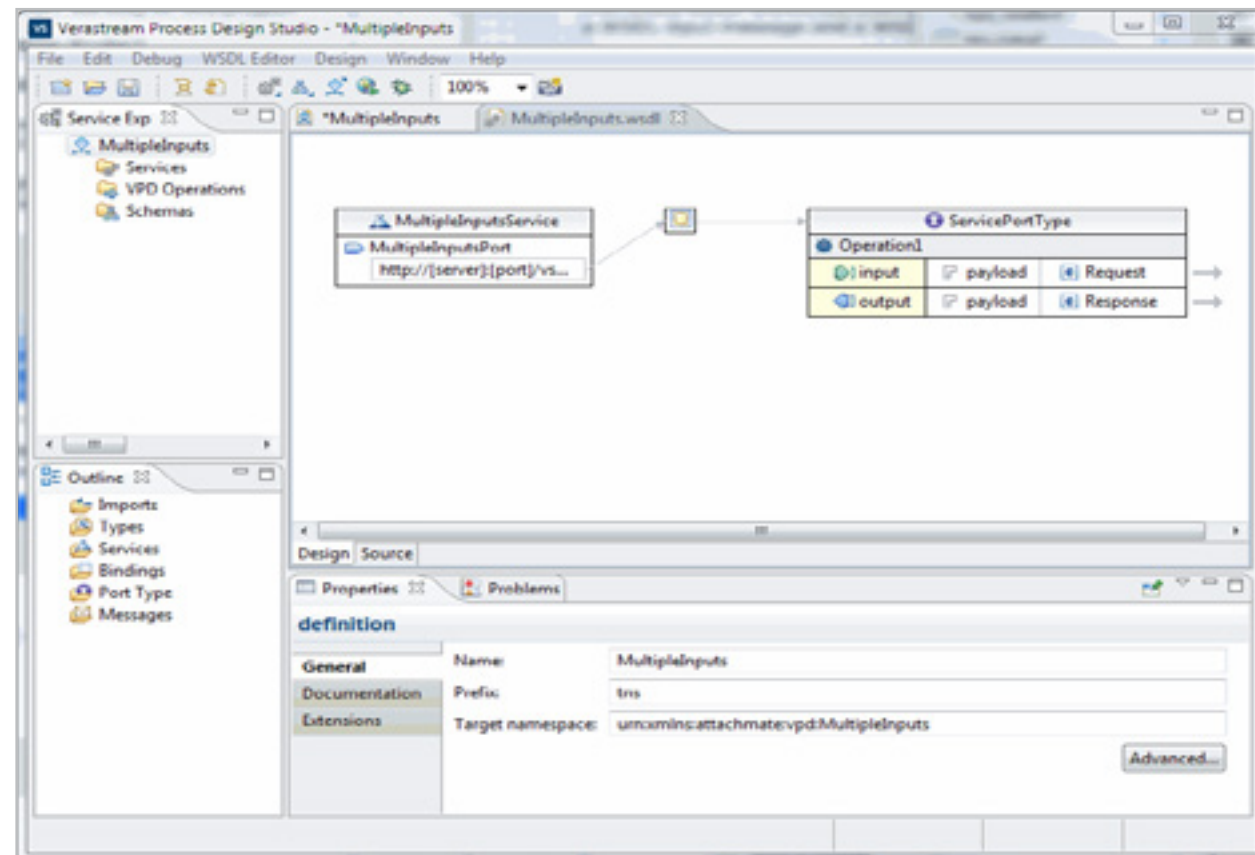
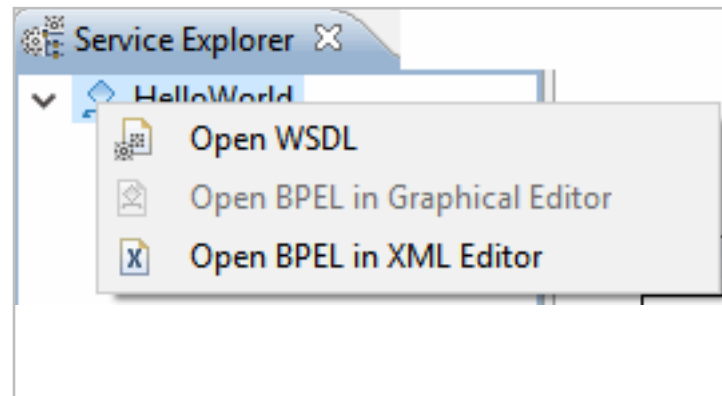
## Opening the WSDL Editor

From the Welcome screen, choose **Start Now** to open the Process Designer.

To open the WSDL Editor and start working on this project:

1. From the File menu, click **New Project**.
2. Name the new project **MultipleInputs**, then click OK.
3. Delete the DoSomethingHere activity (and the Assign it contains) from your default project.
4. In the Service Explorer, right click on the top node and select **Open WSDL**. This opens the WSDL Editor.

You can also open the WSDL Editor through the Project Explorer. Select **Window > Project Explorer**. Right-click on any item in the Project Explorer that ends with 'wsdl', and select **Open WSDL**.



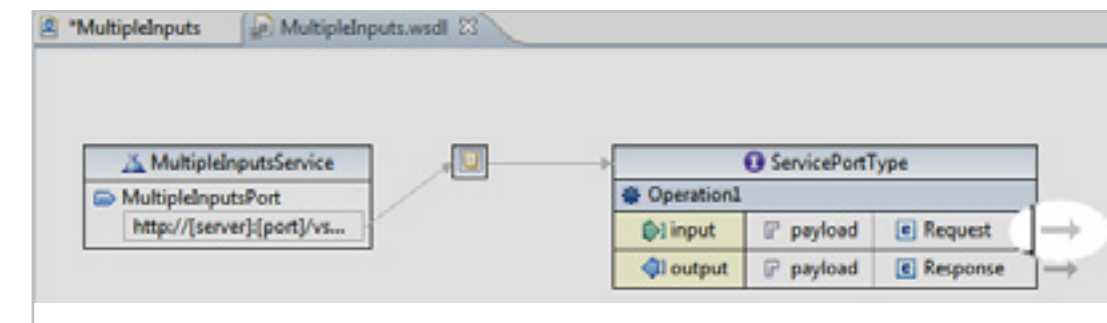
## Using the Schema Editor, and complex schema types

The WSDL editor shows the interface to your project--by default, a WSDL input message and a WSDL output message.

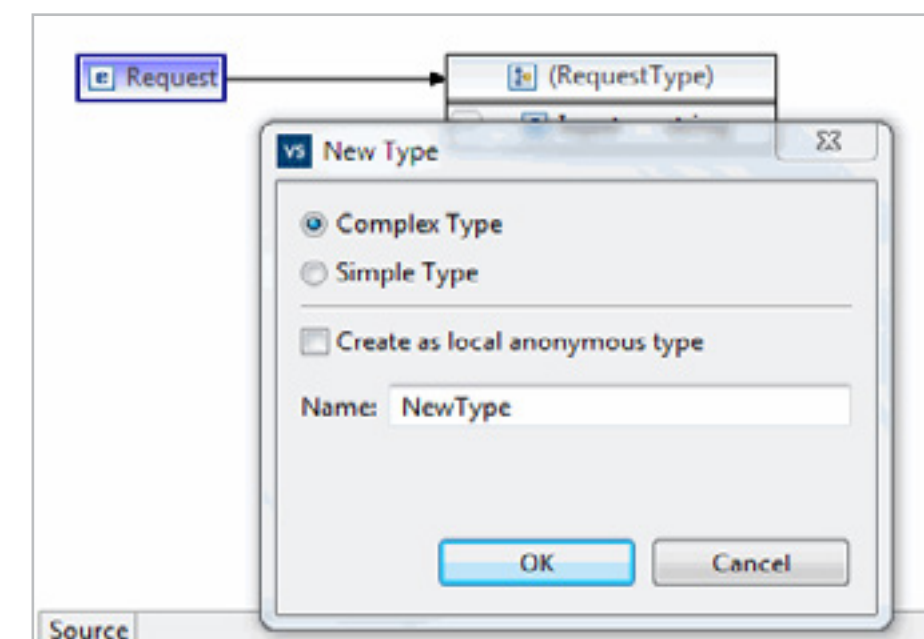
An XML *schema* describes valid structure and content for an XML document. It defines *elements* composed of *types* of data. A WSDL includes a selection of *simple* types, such as *string*, *int*, *float*, *date*, and *double*. A simple type cannot contain sub-elements or attributes; a *complex* type can contain either.

You will create a new complex schema type, then add a single element to it. The element, of type *string*, will be able to occur any number of times. You will then set the WSDL input message to hold an element of your new type.

1. Double-click the arrow to the right of Request. This opens the Schema Editor, where you can change the schema for your process.



2. In the Schema Editor, right-click the **Request** message and choose **Set Type > New** to open the New Type dialog box.



The WSDL Editor displays the input and output of a process.

Because it works with two standards that have slightly different concepts for the word 'type', Process Design Studio uses the word in two ways: to refer to an XML schema concept and to refer to a BPEL concept. These tutorials may sometimes use "BPEL type," or a "schema type" to clarify the relevant context.

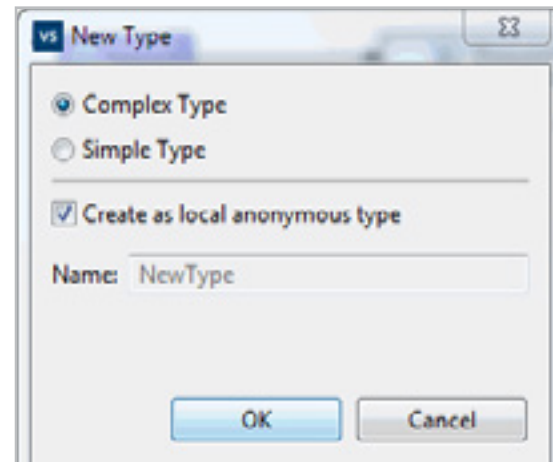
The Schema Editor provides a graphical representation of the underlying structure, or 'schema', of the XML document.

The Schema Editor has two tabs underneath the main window. Click **Source** to view the XML source.

## Changing schema types from simple to complex

A type referenced only once in a schema can be defined as an anonymous type. This is more efficient than naming and referencing each type.

1. On the New Type dialog box, select **Complex Type**.
2. Check **Create as local anonymous type**, then click **OK**.

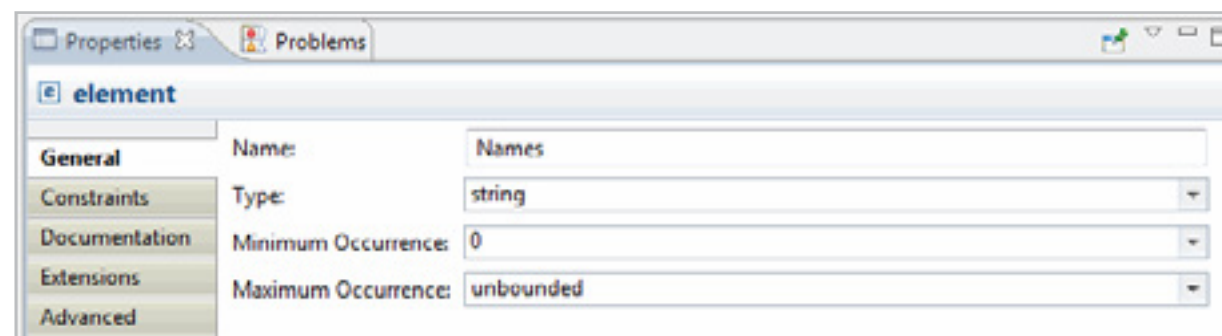


3. The schema editor now shows the new input type. Right-click on the InputType and select Add Element from the context menu. This adds a new element to the XML file.



The Properties view of the element consists of multiple tabs; General, Constraints, Documentation, Extensions, and Advanced.

4. With NewElement selected, in the Properties view, open the General tab and change the name to **Names**.
5. Set the type to **string**.
6. Set the minimum occurrence to **0**.
7. Set the maximum occurrence to **Unbounded** (so the input can accept an unlimited number of strings).
8. Save and close both the Schema and WSDL Editors.



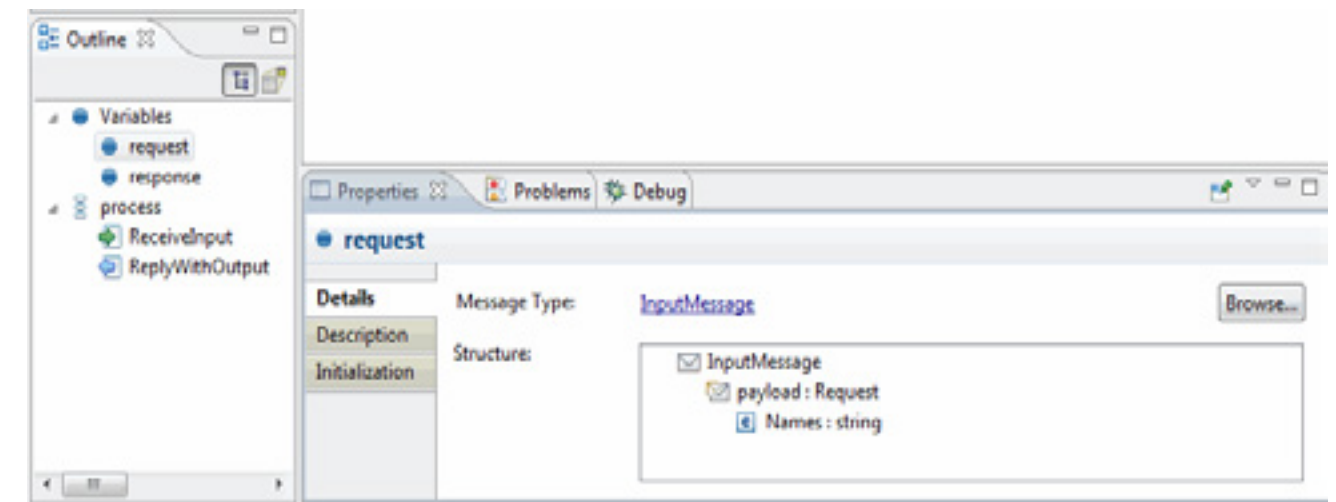
## Verifying the type change

To verify that you've modified the input variable correctly, in the Outline view, under Variables, click **Request**.

The Properties view of the input variable should reflect the new structure you created.

You can also view any problems that may be developing by clicking the Problems tab.

See the online Help for more information on these options.

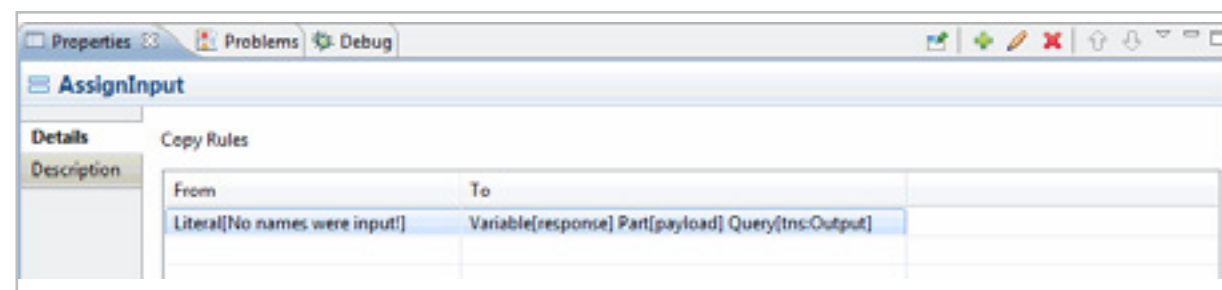
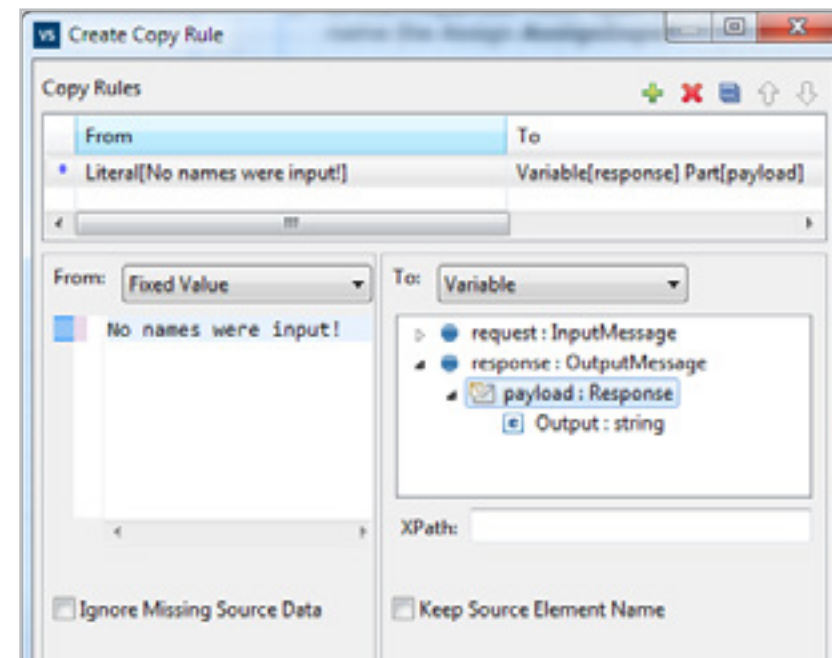


## Handling no input

In the second tutorial you learned to use a BPEL Assign activity for simple data manipulation using XPath expressions. The Assign activity is used to assign values to variables.

What if there aren't any names passed into the process? In that case, the process can show a default message. Use an Assign activity to assign the default message to the Output variable.

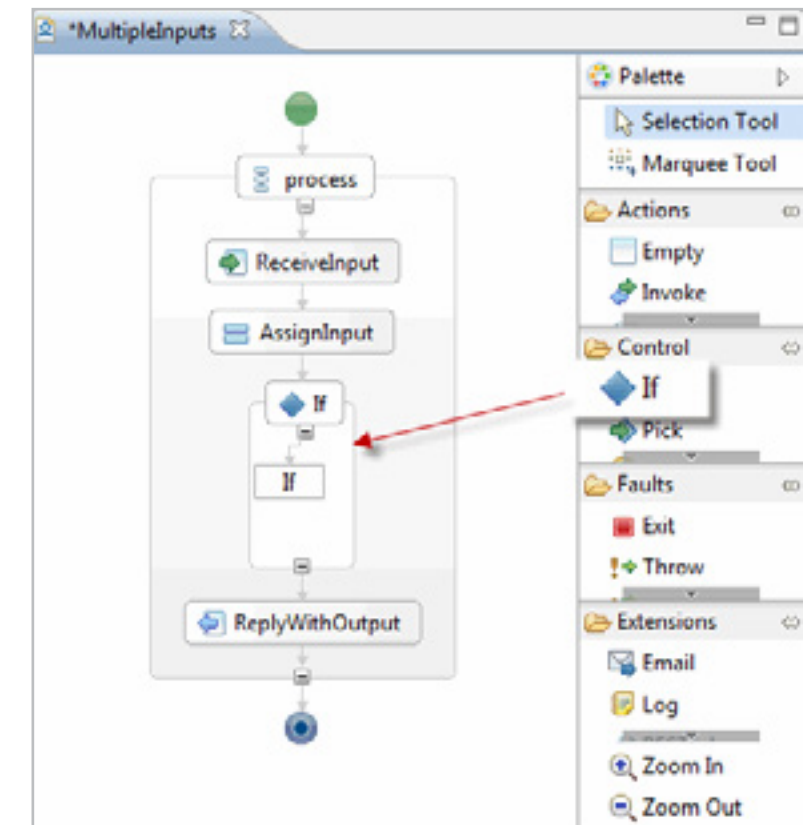
1. Drag an Assign activity from the Palette and drop it on the process flow diagram between the Receive and Reply controls.
2. In the Properties view, open the Description tab and name the Assign **AssignInput**.
3. In the Properties view, open the Details tab, then click the green plus sign ( + ) to create a copy rule that will initialize the output variable.
4. In the From menu, select **Fixed Value**.
5. Type **No names were input!** as the default message.
6. In the To menu, with Variable selected, expand Response:OutputMessage and payload:Response, and then select Output:string. Click **OK**.



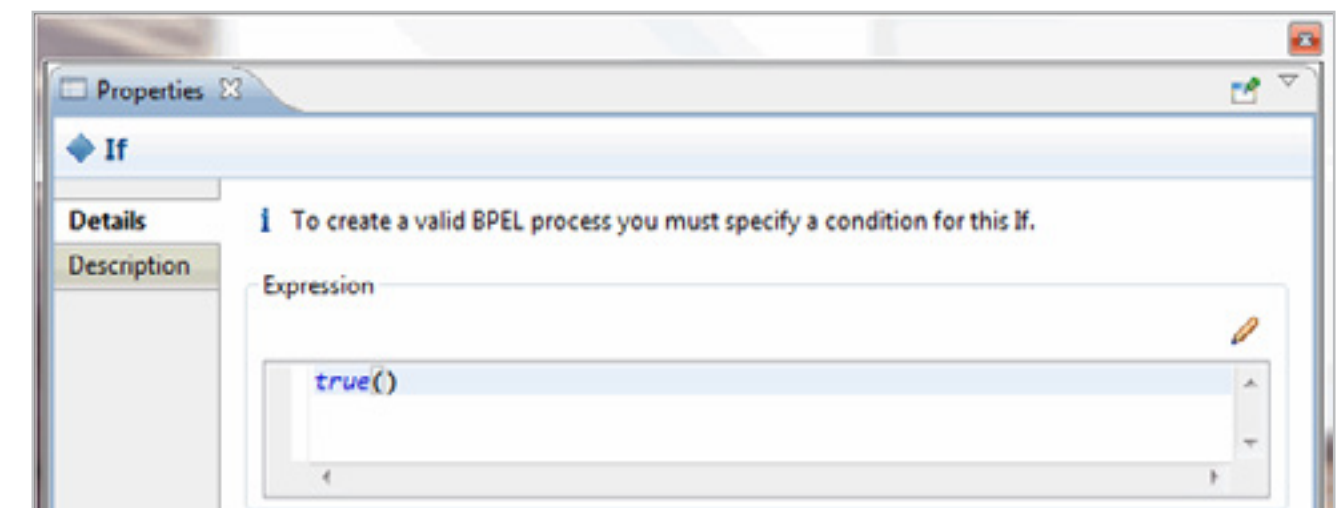
## Using an If activity

You only need to process the list of names if it contains at least one name. To check whether it contains a name, use an If activity. An If activity tests whether a condition is true or false.

1. Drag the IF activity from the Palette and drop it after the AssignInput control in the process flow diagram.



2. In the Properties view for the If activity, click the 'pencil' ( ) icon to open the XPath Expression Editor.



A basic If activity evaluates one condition, but <Else If> and <Else> statements can be added to evaluate more than one.

See the online Help or the WS-BPEL specification (<http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>) for more information.



## Counting nodes in a node set

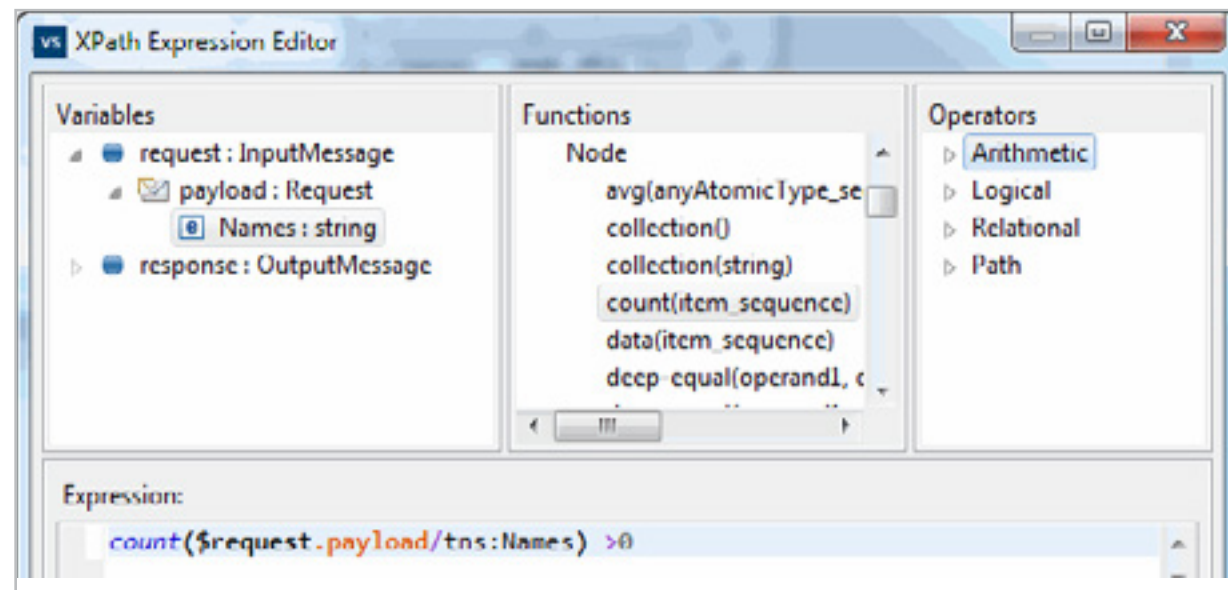
Node sets can be thought of as arrays. An array is a list of things, each identified with a unique key. By default, array keys are numbers, so the first element in the list may be identified with a '1', the second with a '2', and so on.

To refer to a single element in an array, you write the name of the array, followed by a key in square brackets: [ ]. For example, `listOfNames[1]`, refers to the first element in the array `listOfNames`. Sometimes the first key is '0' rather than '1'; if that's the case, then `listOfNames[1]` refers to the second element (because `listOfNames[0]` refers to the first).

The list of names will be passed as a *node set*. Node sets are lists, and *nodes* are items in the list. XPath includes many functions that work with node sets and nodes. For example, the `count()` function counts the nodes in a node set. Your If activity will count the nodes in the list of names. If the count is greater than zero, the list has at least one name.


In the XPath Expression Editor:

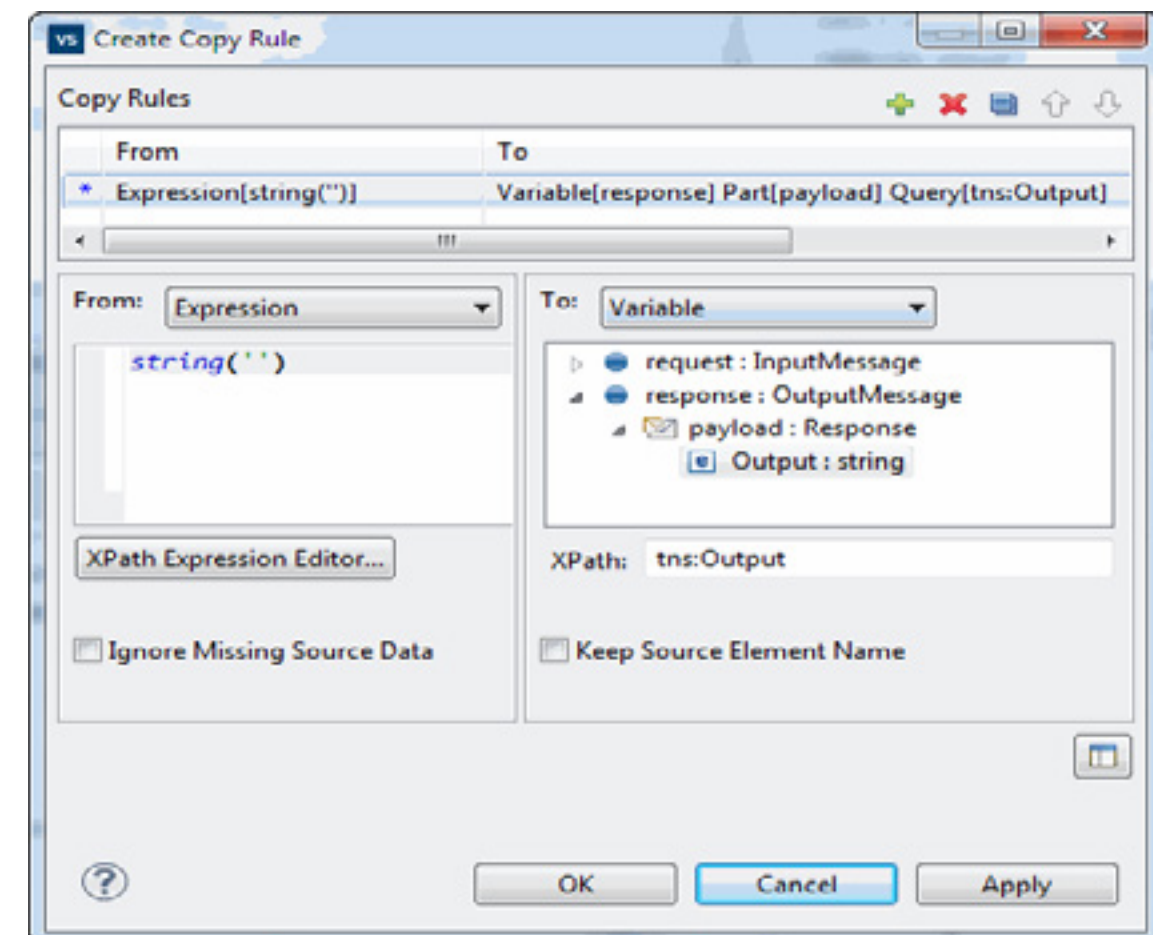
1. In the Expression field, delete the default, `true()`.
2. From the Functions tree, expand Node, then double-click the count function to insert `Count(item_sequence)` (with `item_sequence` highlighted) in the Expression field.
3. In the Variables tree, expand the input variable until the Names element is visible. Double-click `Names:string` to replace `item_sequence` with `$input.payload/tns:Names[1]`, a reference to the first node in the list of names.
4. To count all nodes in the list of names, delete the reference to the first node, `[1]`, leaving: `$request.payload/tns:Names`.
5. Add 'greater than zero' to the expression by typing `>0`.
6. Click OK to close the XPath Expression Editor.



## Clearing the output

If names have been input, you do not want to show the default Response message. In this step, you clear the default message in the Response variable by assigning it an empty string. (The concatenate expression you create for the Response later must start with an empty Response variable.)

1. Drag an Assign activity within the IF control.
2. In the Properties View, click the Description tab and replace the default name with ClearOutput.
3. In the Properties View, click the Details tab, then click the  symbol to create a copy rule.
4. Create a copy rule that assigns an empty string to the output variable, then click OK.



## Using a For Each activity

The goal of this exercise is to put all of the names input into a single string in the Output, starting with the first name, then adding the second, the third, and so on. That's just the kind of work a For Each activity is designed to do.

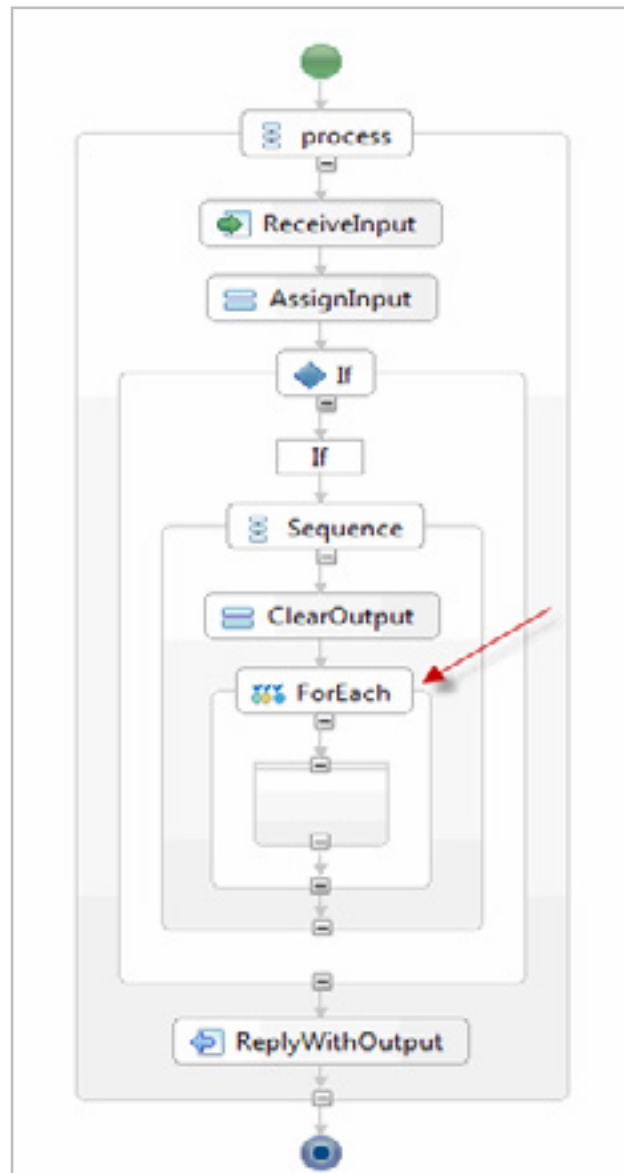
*The small white box inside the For Each activity is its 'scope'. You can put other activities in the scope that you want to run as part of the For Each loop.*

*The If activity also has a scope. (In fact, in this example, the For Each activity is inside an If activity's scope.)*

A For Each activity repeats a sequence (loops) as many times as you tell it to. Every For Each activity includes a counter. Each time it loops, the counter's value increases by one.

For this exercise, you will start the counter at one. The For Each will stop when the counter equals the number of names input. A For Each is a convenient way to do something that uses each node in a node set.

- Drag a For Each activity from the palette and drop it inside the If activity.




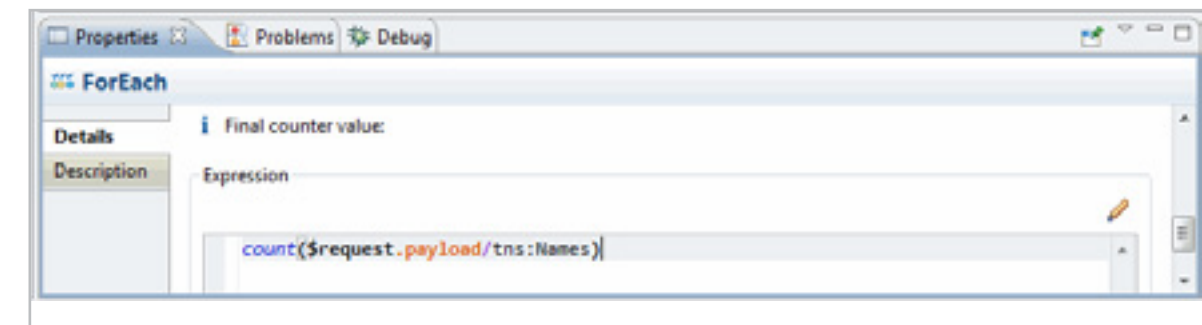
## Looping over each node in a node set

Now that the For Each activity is in place, you need to set its counter so it will loop over each node in the list of names.


By default, the counter variable in the For Each is named 'Counter', it starts at 1, and its final value is 1. You need to change the final counter value. You want it to start at 1, and loop once for each name input, so the final counter value will equal the number of names input.

You will replace the final value with an expression that counts the number of names input. This is the same expression you created earlier, as part of the If activity.

1. Select the For Each activity; the For Each counter is visible in the Details tab of the Properties view.
2. Under Final counter value, click the pencil icon (  ) to open the XPath Expression Editor.
3. Delete the default, **1**.
4. From the Functions tree, expand Node, then double-click the count function to insert `Count(item_sequence)` (with `item_sequence` highlighted) in the Expression field.
5. In the Variables tree, expand the input variable until the Names element is visible. Double-click `Names:string` to replace `item_sequence` with `$request.payload/tns:Names[1]`, a reference to the first node in the list of names.
6. To count all nodes in the list of names, delete the reference to the first node, `[1]`, leaving: `$request.payload/tns:Names`, then click OK.




## Creating a concatenate expression

Sometimes the  icon scrolls offscreen. If you do not see it, make sure you are on the Details tab, then look for scroll bars and try to scroll until you see it again.

The final step before testing is to use an Assign control to create the concatenated string. Every time the For Each loop executes, a name will be added to the string.

The expression will select names in the node set sequentially by using the Counter variable you just created. Remember, to refer to the first item in a node set you use notation like: `Names[1]`, where the 1 is a key. If you replace the 1 with the Counter variable, then each time the value of the Counter variable increments, the next item in the node set is selected.

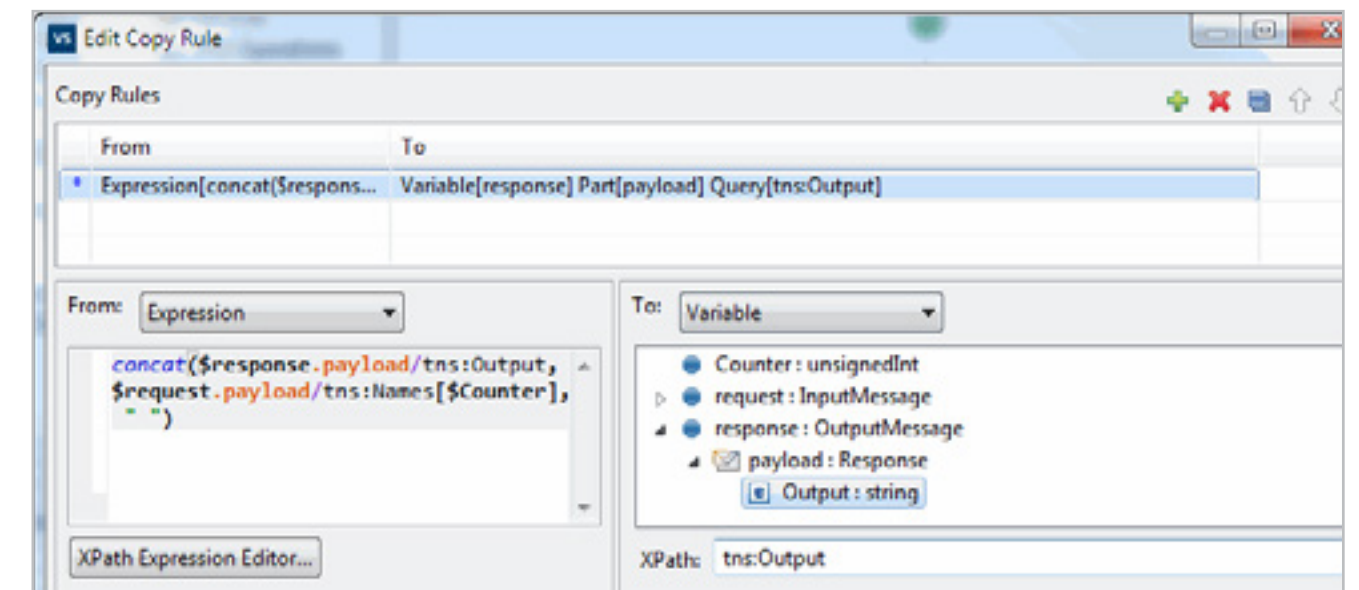
1. From the palette, drag an Assign activity into the For Each.
2. On the Description tab of the Properties view, name the Assign activity **BuildOutput**.
3. With the Details tab selected, click the  on the Properties view toolbar to open the Create Copy Rule dialog box.
4. In the From menu, select **Expression**.
5. Click **XPath Expression Editor...**
6. From the Functions tree, expand the String node and double-click `concat`.
7. With `anyAtomicType_Arg1` highlighted, from the Variables tree, expand `response:OutputMessage`, and `payload:response`, and then double-click `Output:string`.
8. With `anyAtomicType_Arg2` highlighted, expand `request:InputMessage`, and `payload:Request`, then double-click on `Names:string`. The second parameter should now be: `$request.payload/tns:Names[1]`.
9. Highlight the 1 (but not the brackets around it) then, under Variables, double-click on `Counter:unsignedInt`. The 1 is replaced by `$Counter`.
10. Replace `anyAtomicType_OptionalArgs` with quote-space-quote (" ").
11. When the expression looks like the example below, click OK.

Expression:  
`concat($response.payload/tns:Output, $request.payload/tns:Names[$Counter], " ")`

## Assigning the final output

You have completed the concatenate expression. Now you just need to assign its results to the Output variable. The For Each activity is inside the If activity, so it will only create a list of names if names have been input. That list of names then becomes your output. If no names have been input, the process outputs the default message you created earlier.

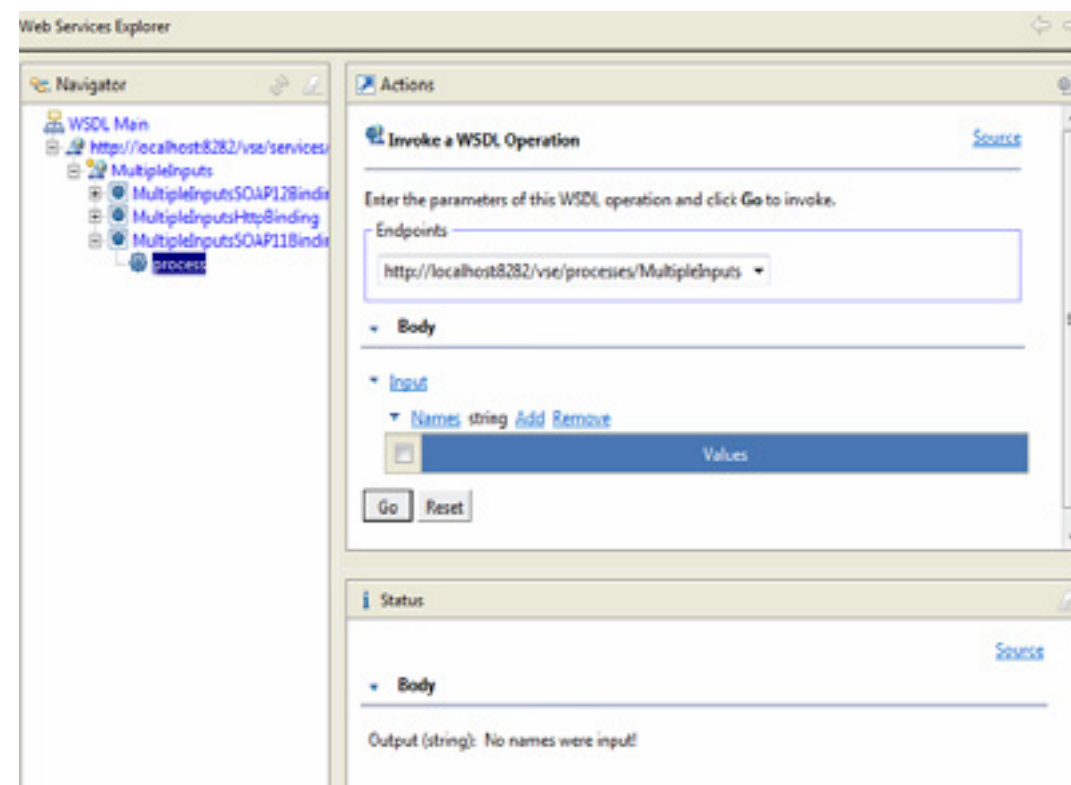
1. In the To side of the Copy Rule, expand `response:OutputMessage`, and `payload:Response`, and then select `Output:string`.
2. Click **OK**.
3. Select **File > Save Project**.



## Deploying and testing the process

You learned how to deploy a BPEL project to a process server in the first Hello World tutorial. A brief recap:

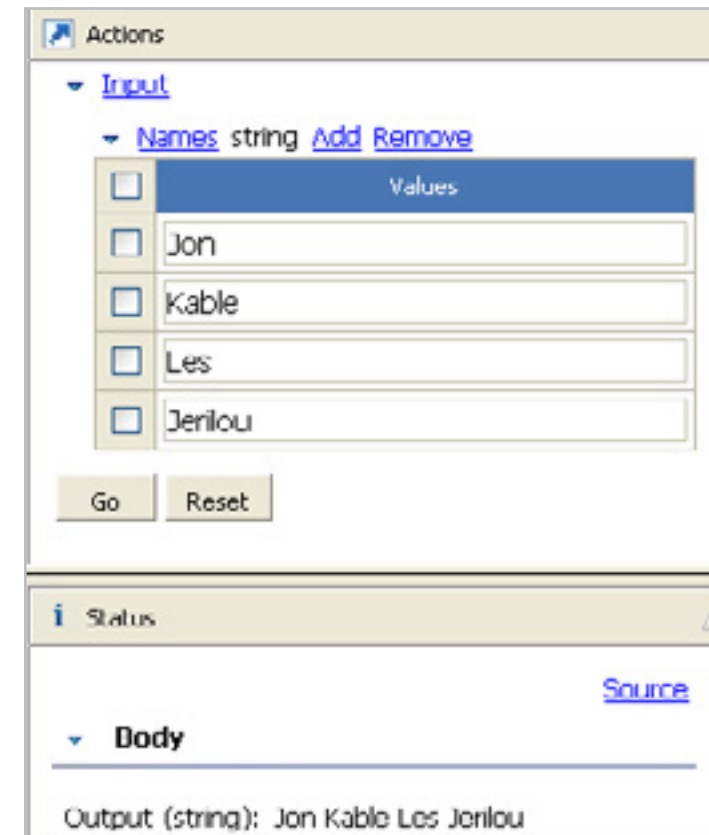
1. From the File menu, select **Deploy to Process Server**.
2. Enter the name, username and password for the server. The defaults are:  
name: **localhost** username: **admin** password: **secret**
3. In the Deployment Succeeded dialog box, click **Test Service** to launch the Web Services Explorer.
4. Select the SOAP11Binding option to test against.
5. Click **Go** to test the response of the service when no



names are entered. In the Status view you should see 'No names were input!'

To test what occurs when a number of different names are input into the process:

1. Click **Add** and enter a name in the field. Repeat this as many times as you want. In this example four names have been added.
2. Click **Go** to test the process. The output should show all of the names you entered, separated by spaces.



Congratulations! You've completed tutorial three. Take a break, then come back for tutorial four.



Reference image of the completed process

---

