VS Process Designer

Tutorial 4: Temporary Variables, Arithmetic, and Review

CONTENTS

This tutorial assumes that you have completed the first through third tutorials and builds upon them.

In this tutorial you will create and initialize temporary variables, and use new arithmetic operators. You will also practice several concepts covered in the previous three tutorials, including creating complex input and output types, using If and For Each activities, and using the XPath Expression editor.

You will create a new BPEL project to accept one or more numbers as inputs. The process will output the sum, average and product of all numbers input, as well as the sum of every other number input.

Prerequisites:

- Micro Focus Verastream Process Design Studio
- An installed and running Micro Focus Verastream Process Server
- Internet browser
- Some familiarity with XML Schema, WSDL, XPath, BPEL, and Web service standards

Let’s get started.

Setting Up

From the Welcome screen, choose Start Now to open the Process Design Studio.

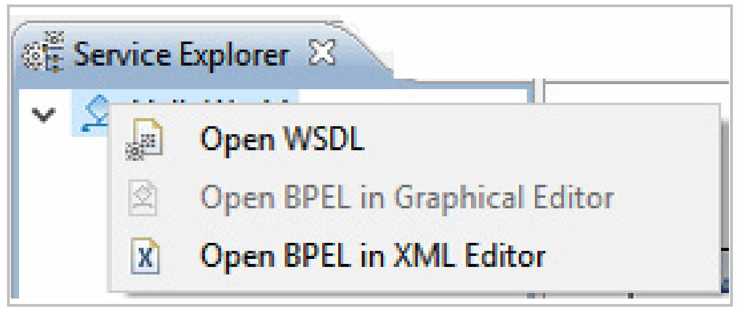
The steps for starting a new project should be familiar by now:

1. Start the Process Design Studio (**Start** > **Micro Focus Verastream** > **Process Designer** > **Process Design Studio**).
2. From the File menu, click **New Project**.
3. Name the new project **Arithmetic**, then click **OK**.
4. Delete the DoSomethingHere activity (and the AssignValue it contains) from your default project.

Modifying the input type

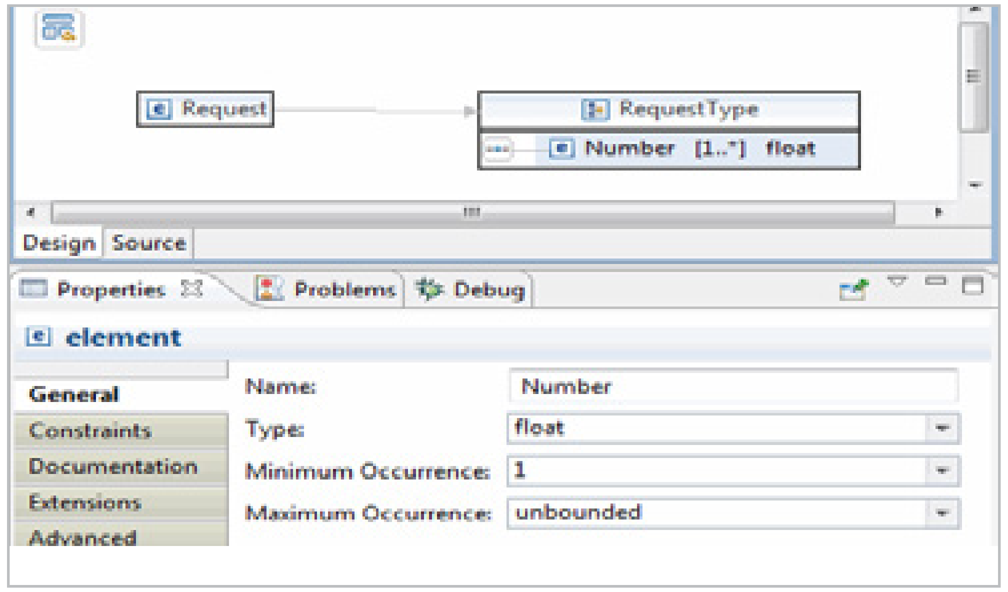
In Tutorial 3 you created an input data type that would accept any number of strings as input. This lesson follows the same steps with slight variation to create a complex input type and an input variable that will accept one or more numbers.

1. In the Service Explorer, right-click 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 any item in the Project Explorer that ends with 'wsdl', and select* **Open WSDL**.

2. Open the Schema Editor by double-clicking the arrow to the right of Request.
3. Select the **Input** element, and in the General tab of the Properties view, change the name to **Number**, change the Type to **Float**, and for Minimum Occurrence select **1**, and in Maximum Occurrence select **Unbounded**.



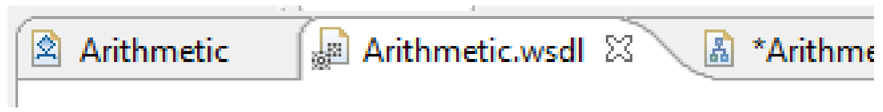
Name: Number
Type: float
Minimum Occurrence: 1
Maximum Occurrence: unbounded

Modifying the output type

Inputs are by default called Requests and outputs are called Responses.

The output type will include several different values (sum, average, product, etc.).

1. Select the Arithmetic.wsdl tab to open the WSDL Editor.

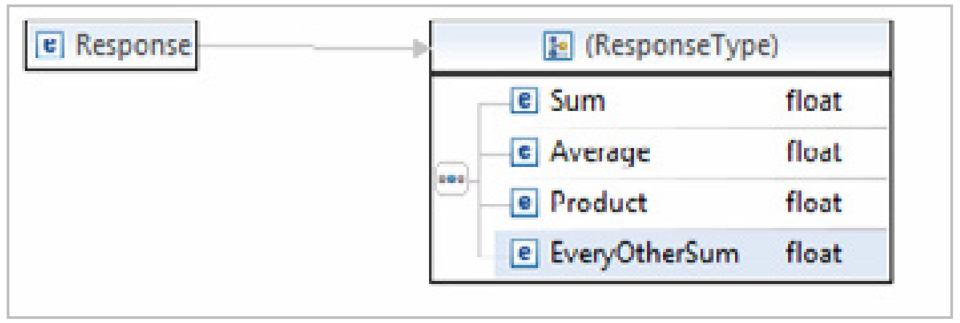


2. Open the Schema Editor by double-clicking the arrow to the right of Response.

3. Right-click ResponseType and choose **Add Element**. Repeat this until the type has four elements, including the original Output element.

4. To change the default type from 'string' to 'float' for each element, double-click the word **string** and choose **float** from the menu.

5. Change the names of the four new elements. Select an element, then in the Properties view, select the General tab and rename the elements to: **Sum**, **Average**, **Product**, **EveryOtherSum**.



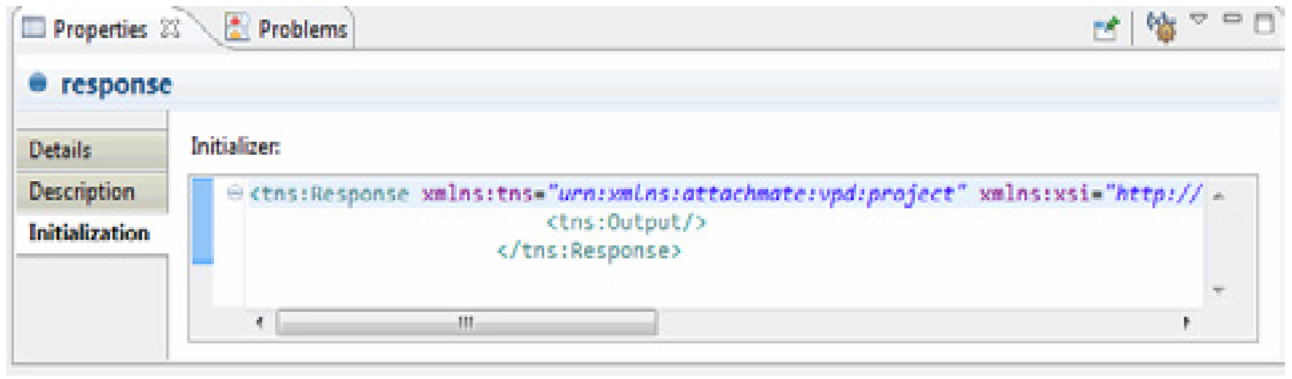
6. When you are done, close the Schema and WSDL editors (click the **X** on the right side of their tabs).

Initializing the output variable

The output variable is initialized by default, however if you modify the structure of the variable, you must reinitialize it before it can be used. To initialize the variable:

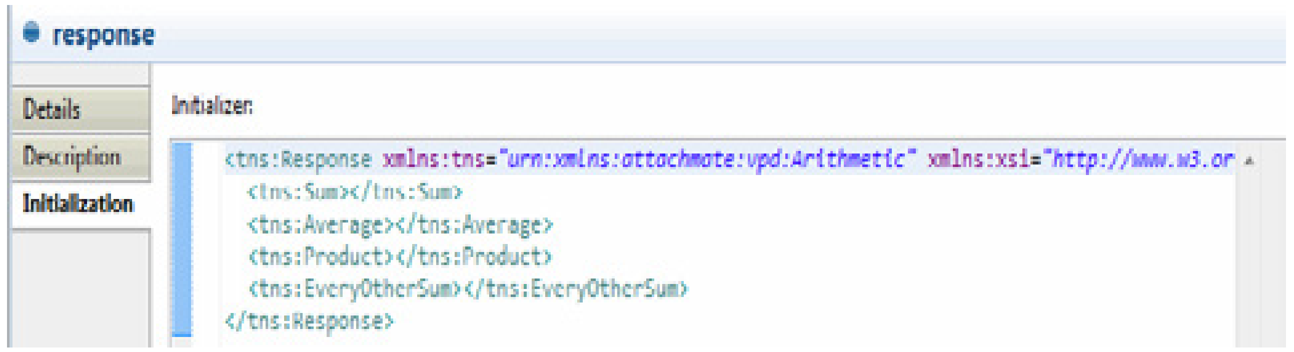
1. In the Outline view, expand Variables, and select **response**.

2. In the Properties view, open the Initialization tab. The response variable looks like this:



```
<tns:Response xmlns:tns="urn:xmlns:attachmate:vpd:project" xmlns:xsi="http://
        <tns:Output/>
    </tns:Response>
```

3. Click Generate XML on the toobar, when the Generate XML dialog box displays, accept the initialization. The response variable should now look like this:



```
<tns:Response xmlns:tns="urn:xmlns:attachmate:vpd:Arithmetic" xmlns:xsi="http://www.w3.or
  <tns:Sum></tns:Sum>
  <tns:Average></tns:Average>
  <tns:Product></tns:Product>
  <tns:EveryOtherSum></tns:EveryOtherSum>
</tns:Response>
```

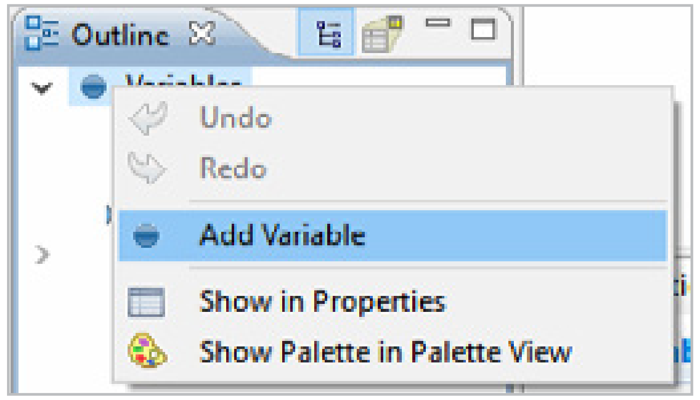
Variables are initialized by defaut, however whenever you modify an output variable, you must reinitialize it, using this same procedure.

Creating and initializing temporary variables

You will now create two temporary variables, one for the product and one for the sum of every other input item. You will build the final values for these items in steps. The temporary variables will hold values as they change.

You may sometimes want to create temporary variables even when you could do without them. The thoughtful use of temporary variables can make your process easier for others to understand, and easier to debug.

1. In the Outline view, right-click **Variables** and select **Add Variable**.

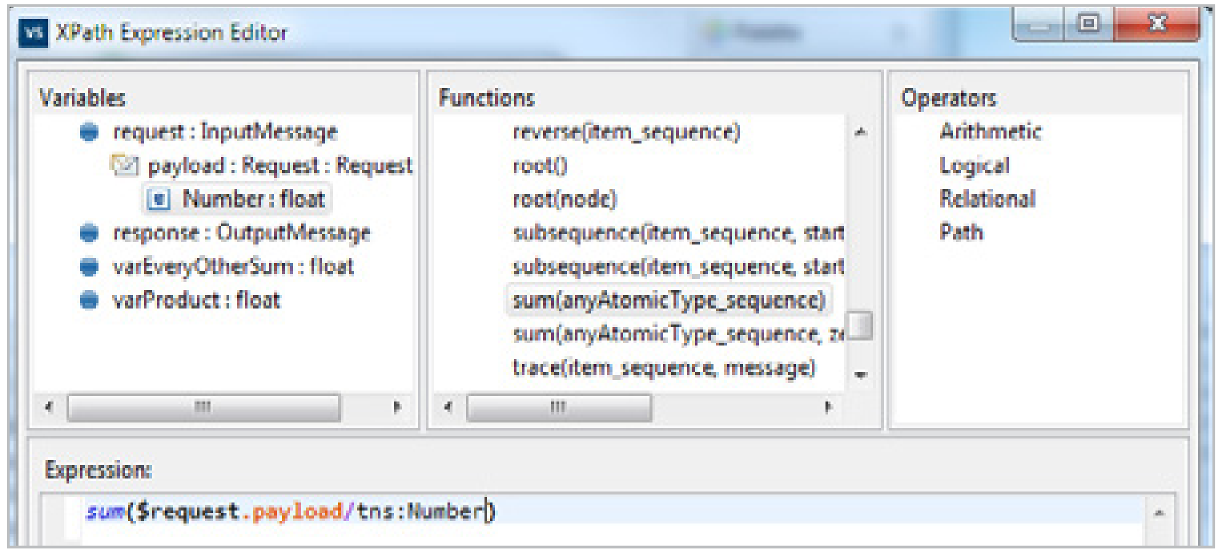


2. Name the first variable **varProduct**, then click **OK**.
3. In the Type Selector dialog, find the Filter Types text field and type in **fl**. When you see **Float** in the Types list, double-click it.
4. Repeat steps 1-3 for a second variable, but name it **varEveryOtherSum**.
5. In the BPEL Editor, add a new Assign activity between the ReceiveInput and ReplyWithOutput. Name it **AssignInputs**.
6. In the BPEL Editor, select the **AssignInputs** activity. In the Properties view, click ✚ to create a copy rule.
7. In the From menu, select **Fixed Value**. In the textbox on the From side, enter a **1**.
8. On the To side, select `varProduct:float`, then click **Apply**.
9. Make a second copy rule. Make it a Fixed Value as well, but set its value to **0**. On the To side, select `varEveryOtherSum:float`. Click OK.

Summing the input

Using the BPEL sum function, you will obtain the sum of all of the numbers input and create an Assign activity to hold copy rules for the Sum and the Average.

1. In the BPEL Editor, add a new Assign activity between the AssignInputs activity and ReplyWithOutput.
2. In the Properties view, open the Description tab to change its name to **AssignSumAverageOutput**.
3. Open the Details tab and click ✚ to create a copy rule.
4. In the From menu, select **Expression**, then click **XPath Expression Editor...**.
5. In the Functions tree, expand **Node**, then double-click the **sum** function to insert `sum(anyAtomicType_sequence)` in the Expression field.
6. In the Variables tree, expand the **request:InputMessage** until you see `Number:float`. Double-click `Number:float` to replace `anyAtomicType_sequence` with `$request.payload/tns:Number[1]`.
7. Because `Number[1]` refers to only the first node in the node set, and you want to sum up all the nodes, delete the `[1]`, leaving `sum($request.payload/tns:Number)`. Click **OK**, to return to the Create Copy Rule dialog.

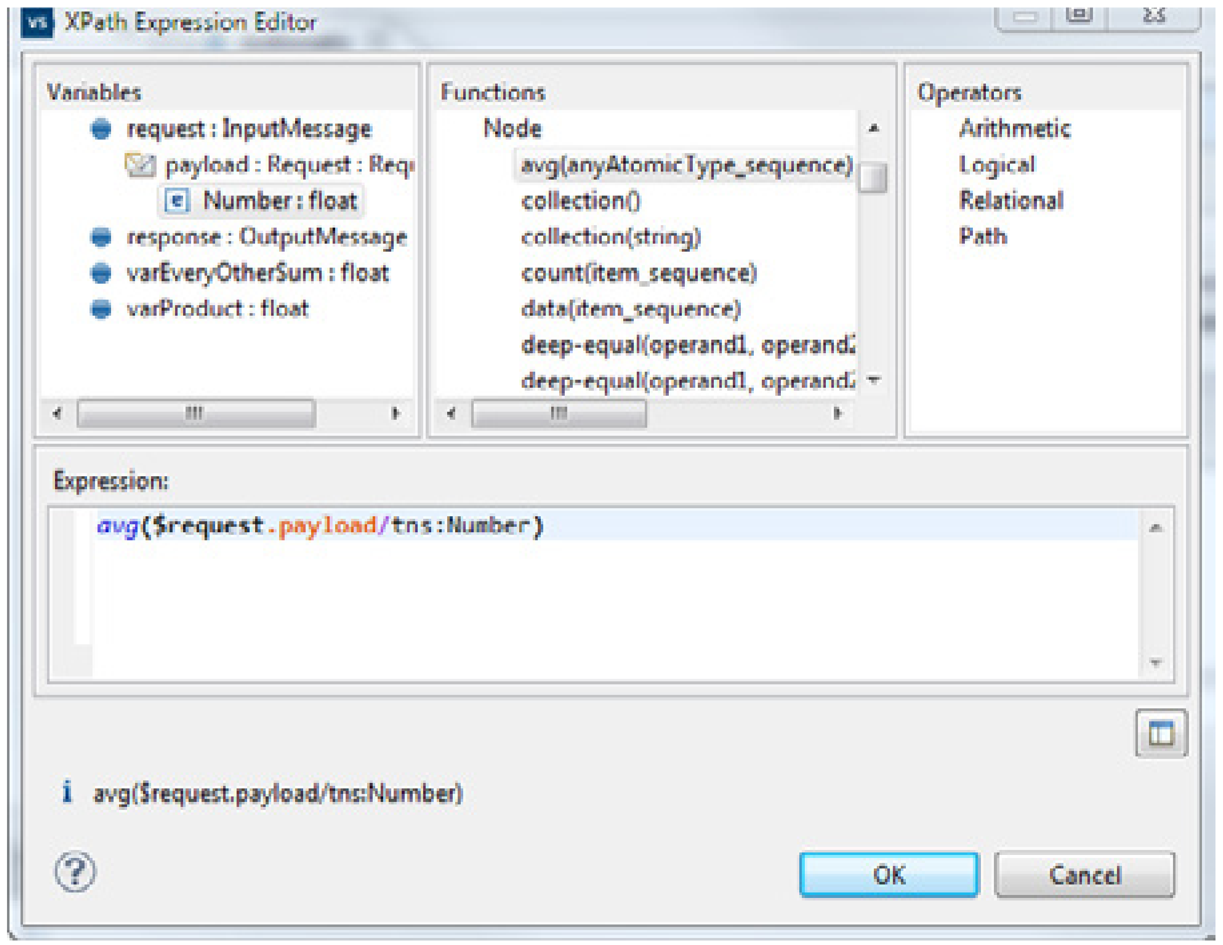


```
sum($request.payload/tns:Number)
```

8. On the To side of the copy rule, expand `Response:OutputMessage`, then expand `payload:Response`. Select `Sum:float` and click **OK**.

Averaging the input

1. Select **AssignSumAverageOutput**, open the Details tab and click ➕ to create a copy rule.
2. In the From menu, select Expression, then click **XPath Expression Editor...**.
3. In the Functions tree, expand **Node**, then double-click the **avg** function to insert `avg(anyAtomicType_sequence)` in the Expression field.
4. In the Variables tree, expand the input variable until `Number:float` is visible. Double-click `Number:float` to replace `node` with `$request.payload/tns:Number[1]`.
5. Delete the `[1]`.
6. On the To side of the copy rule, expand `response:OutputMessage`, then expand `payload:Output:ResponseType`. Select `Average:float` and click **OK**.



```
avg($request.payload/tns:Number)
```

avg($request.payload/tns:Number)

Checking your work

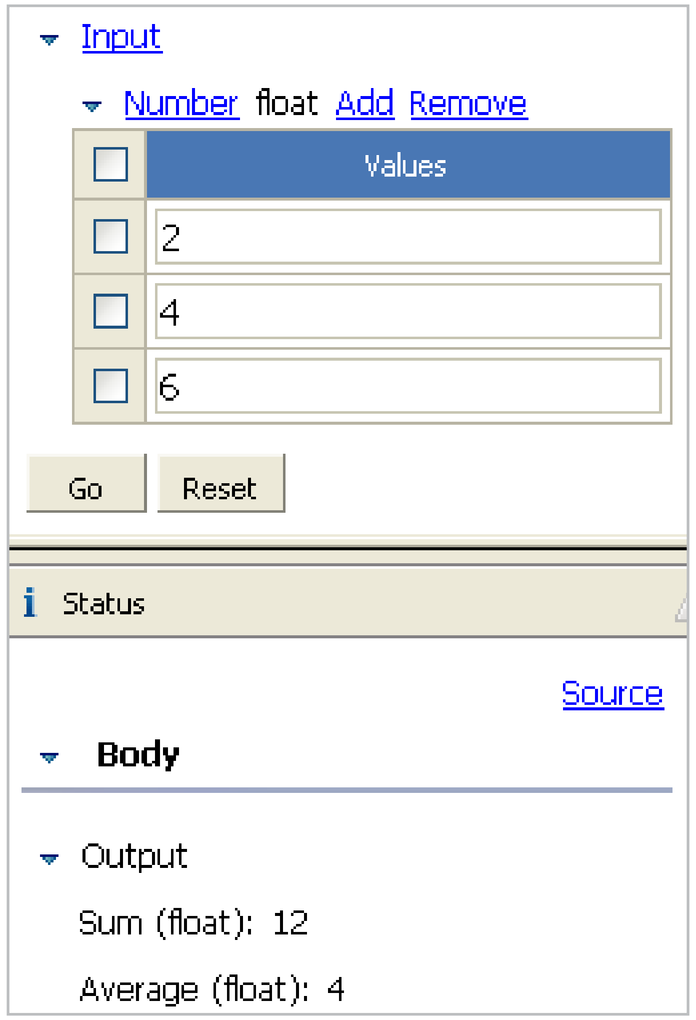
Often, during the course of creating a project you will come to points where it is easy to deploy and test the project. You should plan these points and take advantage of them by deploying and testing your project even though it may not be complete. If the project behaves as expected, you can continue with confidence. If the project responds in an unexpected way, there are fewer places to look to correct an error.

Remember to save your project regularly.

It is good practice to check your work as often as possible. The earlier you can catch errors, the more smoothly your projects are likely to progress.

1. Save your project, then 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. You should see one field in which you can enter numbers. Click the **Add** link twice, then enter the numbers 2, 4 and 6. Click the **Go** button.
5. The service should return a sum of 12 and an average of 4. If it does, congratulations, time to move on to the next steps. If it does not, check your work up to this point in the tutorial.



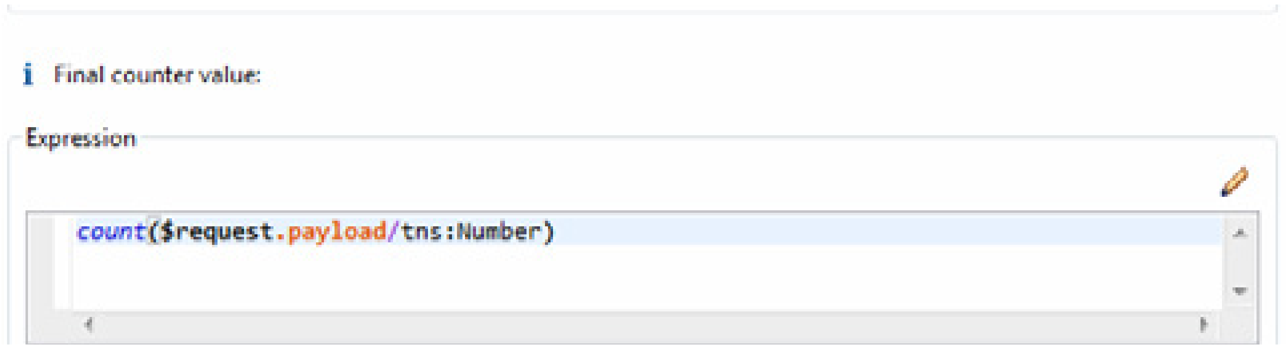
	Values
☐	2
☐	4
☐	6

Sum (float): 12

Average (float): 4

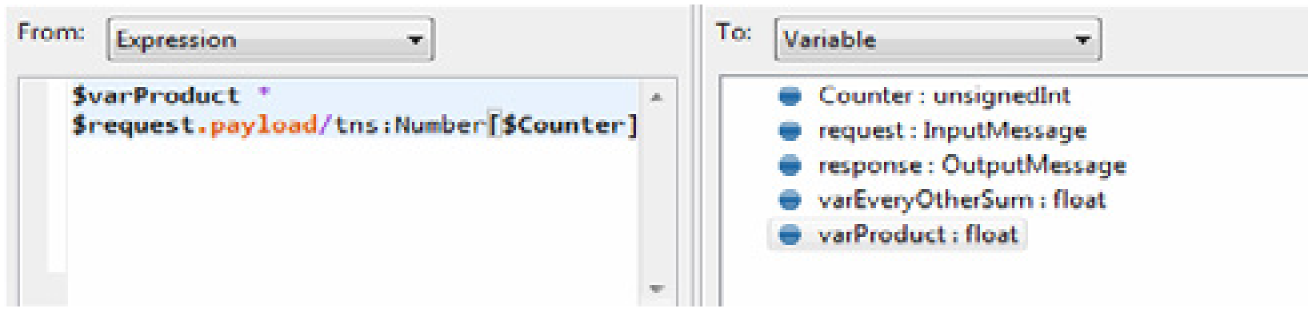
To calculate the product of all the numbers input, you will multiply the first two numbers input, save their product, multiply it by the third number input, save the product, multiply by the fourth number, save the product, and so on. You will do this using a For Each activity. The For Each will contain an Assign activity, calculate the product and assign it to the Response:output.

1. Insert a For Each activity between AssignSumAverageOutput and ReplyWithOutput.
2. With the For Each activity selected, in the Properties view, open the Details tab. Scroll down to Final Counter Value, and click the pencil icon ( ) to open the XPath Expression Editor.
3. In the Expression field, delete the `1`.
4. From the Functions tree, expand **Node**, then double-click **count**.
5. With `item_sequence` highlighted, expand the Request:Input variable until you see `Number:float`. Double-click Number:float. `Item_sequence` is replaced by `$request.payload/tns:Number[1]`.
6. Delete `[1]`, then click **OK**.



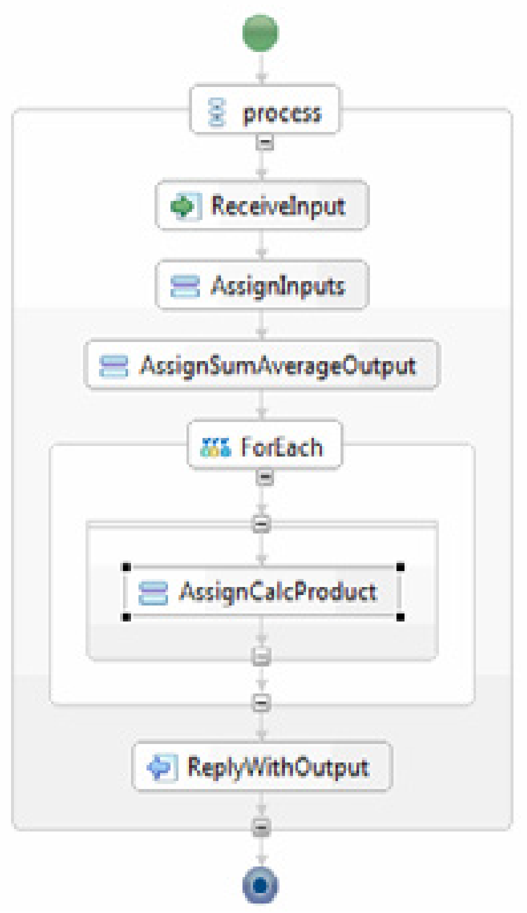
```
count($request.payload/tns:Number)
```

7. On the palette, select an Assign activity then click inside the For Each. Name the assign activity **AssignCalcProduct**.
8. Select **AssignCalcProduct**, open the Details tab and click to create a copy rule.
9. In the From menu, select **Expression**, then click **XPath Expression Editor…**.
10. In the Variables tree, double-click `varProduct:float`.
11. In the Operators tree, expand **Arithmetic** and double-click *** (multiplication)**.
12. In the Variables tree, expand `request:InputMessage`, and `payload:Input`, then double-click `Number:float`. The Expression field should now contain: `$varProduct * $request.payload/tns:Number[1]`
13. Highlight the `1` (but not the brackets), then, in the Variables tree, double-click `Counter:unsignedInt`. The `1` is replaced by `$Counter`. Click **OK**.
14. On the To side of the Create Copy Rule dialog, select `varProduct:float`. Then click **OK**.



```
$varProduct *
$request.payload/tns:Number[$Counter]
```

The process should now look this this:



Summing every other input

The next step is to find the sum of every other value input. To do this, you will use an If activity to test whether an item should be added to the sum. The For Each activity's Counter variable tracks the order of items input. You will use it with the mod (modulus) operation to find every other item.

The mod function returns the remainder of a division operation as an integer. For example, 2 mod 2 equals zero, because there is no remainder; 3 mod 2 returns the remainder of 1; 4 mod 2 equals zero again, 5 mod 2 equals 1, and so on.

1. Insert an If activity inside the For Each activity and after AssignCalcProduct.

2. With the If activity selected, in the Properties view, open the Details tab. Click 🖉 to open the XPath Expression Editor.

3. Delete the default value, `true()`.

4. Under Variables, double-click `Counter:unsignedInt` to insert the For Each loop's Counter variable.

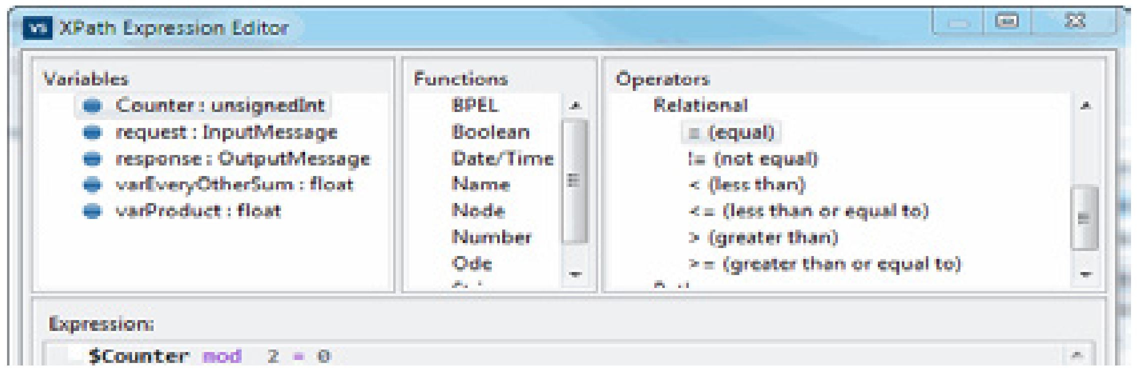
5. Under Operators, expand **Arithmetic** and double-click **mod**, then type a space and then the number `2`.

6. Expand **Relational** and double-click **= (equal)**.

7. Type a zero: `0`. Click **OK**.

Rather than double-clicking on the operators, you can also just type: `$Counter mod 2 = 0`

Selecting operators from the list may reduce errors.



```
$Counter mod 2 = 0
```

8. In the BPEL Editor, add an Assign activity inside the If activity. Name it **AssignEveryOtherSum**.

9. Add a copy rule (➕) to AssignEveryOtherSum.

10. In the From menu, select **Expression**, then click **XPath Expression Editor...**.

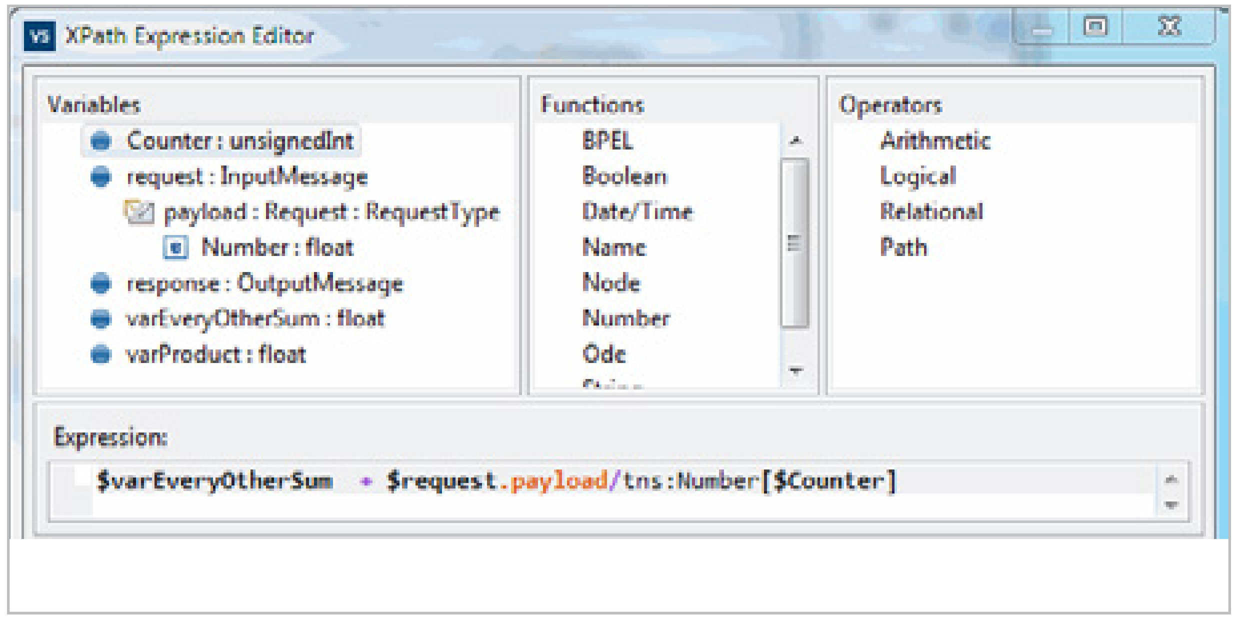
11. In the Variables tree, double-click `varEveryOtherSum:float`.

12. Type a space, a plus sign (**+**), and another space.

13. In the Variables tree, expand `request:InputMessage`, and `payload:Request`, then double-click `Number:float`. The Expression field should now contain:
`$varEveryOtherSum + $request.payload/tns:Number[1]`

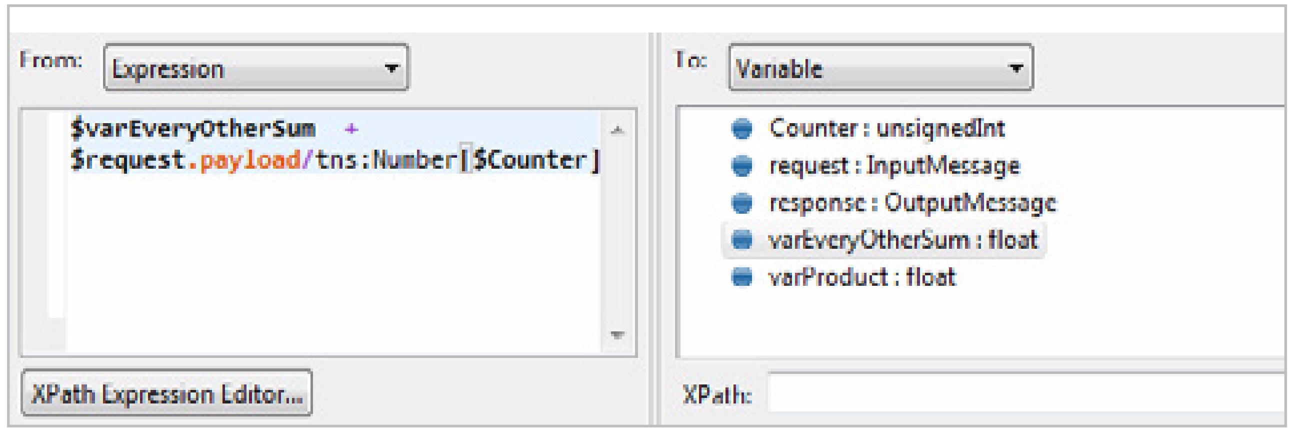
`Counter mod 2 = 0` *will return True for items in even-numbered positions in the list of inputs, and False for items in odd-numbered positions. If you wanted the reverse, with items in odd-numbered positions returning True, you could use:* `$Counter mod 2 = 1`

14. Highlight the `1` (but not the brackets), then, in the Variables tree, double-click `Counter:unsignedInt`. The `1` is replaced by `$Counter`. Click **OK**.



```
$varEveryOtherSum + $request.payload/tns:Number[$Counter]
```

15. On the To side of the Create Copy Rule dialog, select `varEveryOtherSum:float`. Then click **OK**.

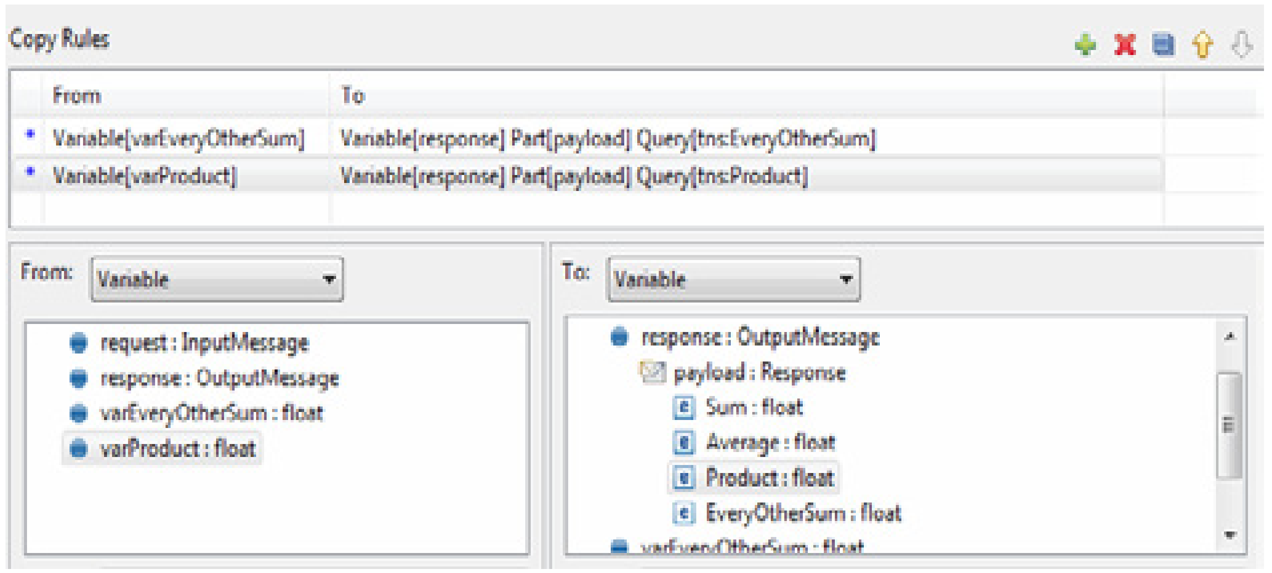


```
$varEveryOtherSum +
$request.payload/tns:Number[$Counter]
```

Assigning output

Now, assign the product of every input and the sum of every other input to their respective output variables.

1. Add an Assign activity between the For Each activity and ReplyWithOutput. Name it **AssignProdEveryOtherSumOutput**.
2. Select it, and then add a copy rule (✚) to it.
3. In the From side, select `varEveryOtherSum:float`.
4. On the To side of the copy rule, expand `output:OutputMessage`, then `payload:Output`, and select `EveryOtherSum:float`.
5. Add another copy rule (✚).
6. In the From side of the copy rule, select `varProduct:float`.
7. On the To side of the copy rule, expand `response:OutputMessage`, then `payload:Response`, and select `Product:float`. Click **OK**.


Copy Rules

From	To
Variable[varEveryOtherSum]	Variable[response] Part[payload] Query[tns:EveryOtherSum]
Variable[varProduct]	Variable[response] Part[payload] Query[tns:Product]

From: Variable

- request : InputMessage
- response : OutputMessage
- varEveryOtherSum : float
- varProduct : float

To: Variable

- response : OutputMessage
  - payload : Response
    - Sum : float
    - Average : float
    - Product : float
    - EveryOtherSum : float
- varEveryOtherSum : float

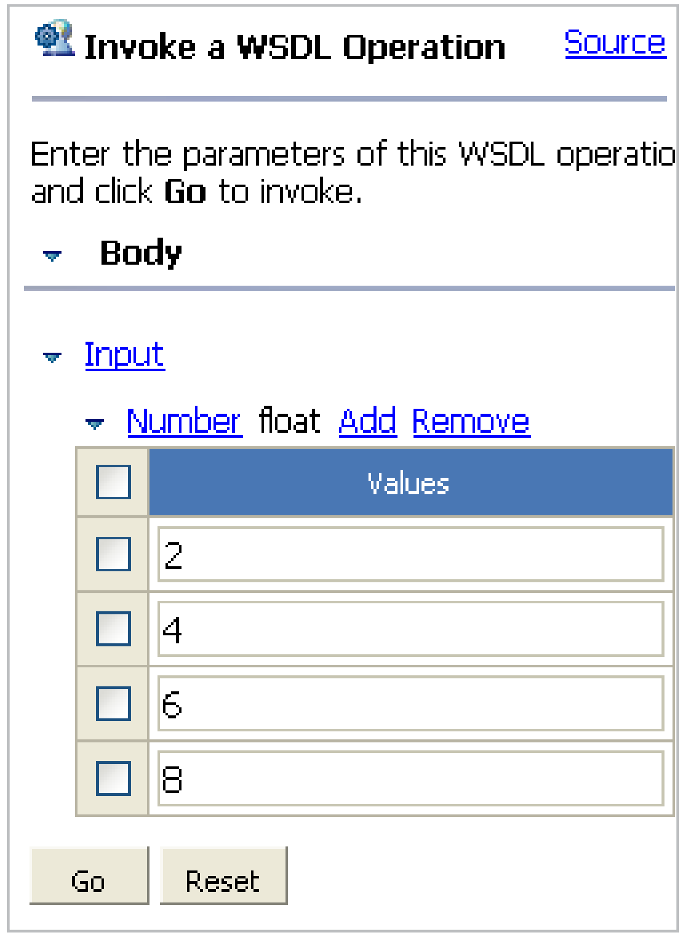
Deploying and testing

It's time to deploy and test your new process. The last page of this tutorial is a reference graphic showing the completed process in the BPEL Editor.

Remember to test against SOAP11Binding in the left panel of the Web Services Explorer.

1. Save your project (**File** > **Save Project**).
2. From the File menu, select **Deploy to Process Server**.
3. Enter the name, username and password for the server. The defaults are:

   name: **localhost**   username: **admin**   password: **secret**
4. In the Deployment Succeeded dialog box, click **Test Service** to launch the Web Services Explorer.
5. Click the **Add** link three times.
6. In each of the Values fields, enter one of the following the numbers: **2**, **4**, **6** and **8**. Enter the numbers in that order. Click the Go button.
7. The service should return these values: Sum = 20, Average = 5, Product = 384, EveryOtherSum = 12 (because the process is adding 4 and 8).


Invoke a WSDL Operation Source

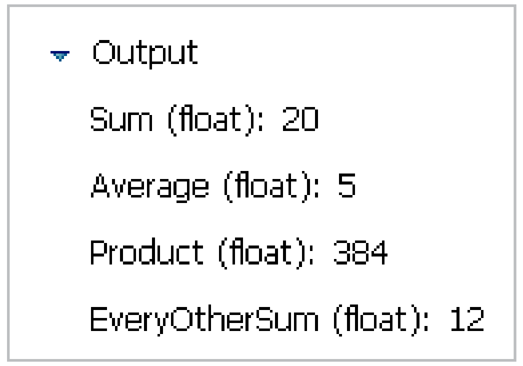
Enter the parameters of this WSDL operatio and click **Go** to invoke.

Body

Input

Number float Add Remove

	Values
☐	2
☐	4
☐	6
☐	8


Output

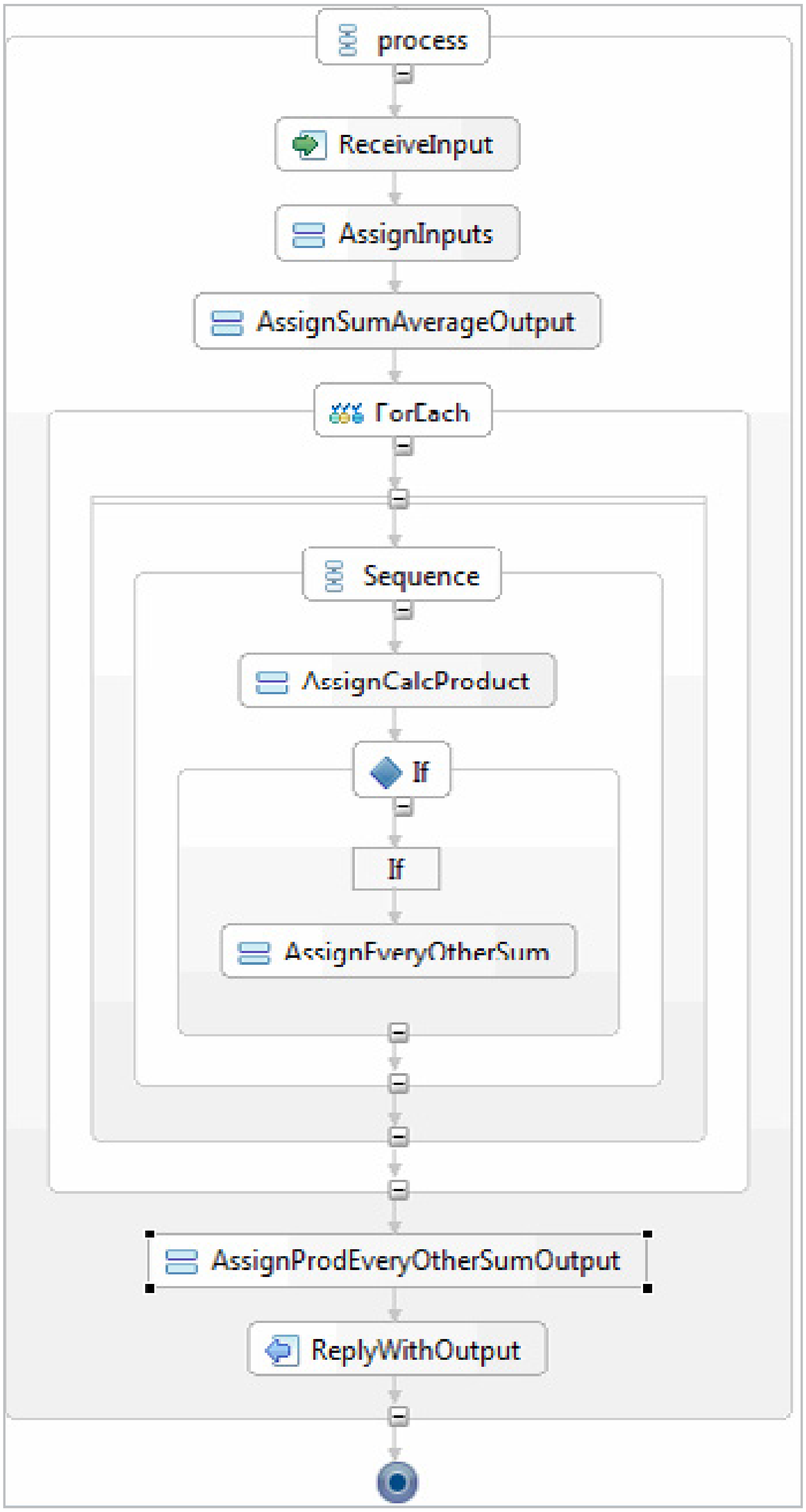
Sum (float): 20

Average (float): 5

Product (float): 384

EveryOtherSum (float): 12

Reference image of the completed process


process
ReceiveInput
AssignInputs
AssignSumAverageOutput
ForEach
Sequence
AssignCalcProduct
If
If
AssignEveryOtherSum
AssignProdEveryOtherSumOutput
ReplyWithOutput