

Cognizant Technology Solutions

Exercises – Basic Flow Steps

WebMethods Integration Workshop

WebMethods CoE

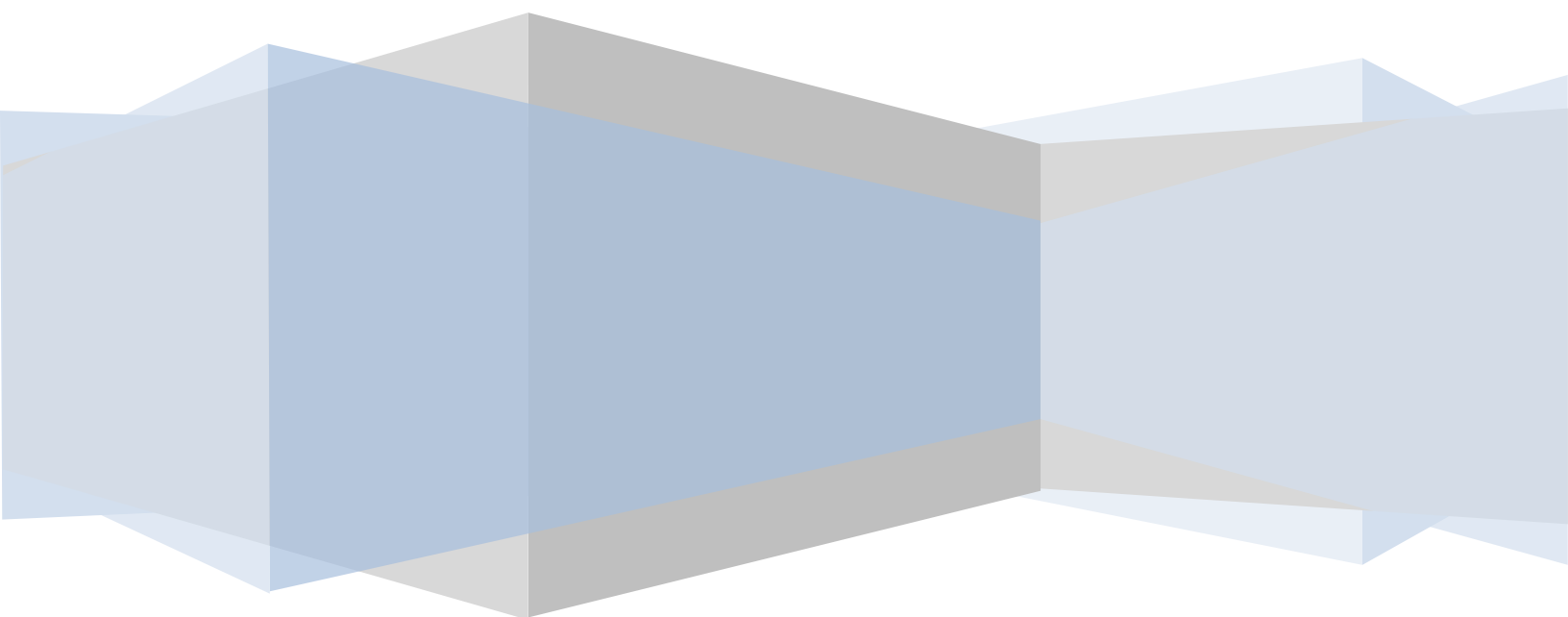


Table of Contents

Introduction	3
Loop	3
Sequence.....	3
Repeat.....	3
Exit	4
Theory.....	4
Practical	5
LOOP Flow Step Exercise.....	5
Task	5
Steps to create LOOP flow step:	5
SEQUENCE Flow Step Exercise	7
Task	7
Steps to create SEQUENCE flow step:.....	8
REPEAT Flow Step Exercise	11
Task	11
Steps to create REPEAT flow step:.....	11
EXIT Flow Step Exercise	13
Task	13
Steps to create EXIT flow step:	13

Flow Step

Introduction

A flow service contains **flow steps**. A flow step is a basic unit of work (expressed in the webMethods flow language) that webMethods Integration Server interprets and executes at run time. The webMethods flow language provides flow steps that invoke services and flow steps that let you edit data in the pipeline.

webMethods' flow language also provides a set of control steps that allow you to direct the execution of a flow service at run time. The control steps allow you to:

- Conditionally execute a specified sequence based on a field value.
- Retry a specified sequence until it succeeds.
- Repeat a specified sequence (loop) for each element in an array field.

Loop

The **LOOP** step repeats a sequence of child steps once for each element in an array that you specify. For example, if your pipeline contains an array of purchase-order line items, you could use a LOOP step to process each line item in the array.

Sequence

You use the **SEQUENCE** step to build a set of steps that you want to treat as a group. Steps in a group are executed in order, one after another. By default, all steps in a flow service, except for children of a BRANCH step, are executed as though they were members of an implicit SEQUENCE step (that is, they execute in order, one after another). However, there are times when it is useful to explicitly group a set of steps.

The most common reasons to do this are:

- To group a set of steps as a single alternative beneath a BRANCH step.
- To specify the conditions under which the server will exit a sequence of steps without executing the entire set.

Repeat

The **REPEAT** step allows you to conditionally repeat a sequence of child steps based on the success or failure of those steps. You can use REPEAT to:

- **Re-execute (retry) a set of steps if any step within the set fails.** This option is useful to accommodate transient failures that might occur when accessing an external system (for example, databases, ERP systems, Web servers, or Web services) or device.
- **Re-execute a set of steps until one of the steps within the set fails.** This option is useful for repeating a process as long as a particular set of circumstances exists (for example, data items exist in a data set).

Exit

The **EXIT** flow step allows you to exit the entire flow service or a single flow step. You specify whether you want to exit from:

- The nearest ancestor (parent) LOOP or REPEAT flow step to the EXIT flow step.
- The parent flow step of the EXIT flow step.
- A specified ancestor flow step to the EXIT flow step.
- The entire flow service.

Theory

- 8-2-SP1_Developer_Users_Guide (Chapter 7 (LOOP, SEQUENCE, REPEAT & EXIT))
- 8-2-SP1_Service_Development_Help (Chapter 10, Chapter 12)

Practical


LOOP Flow Step Exercise



Task

- Count the number of employees in a department:
 - In the services subfolder, create a service **loop_test** with a **document reference** to **otcSupport.documents:Department** named **dept** as input, and a single **String** field **count** as output.
 - Use the **LOOP** flow statement to determine a value for **count**
 - Display the result (**count**) in the Server Log.
- Run** it (Provide your own input data).


Steps to create LOOP flow step:

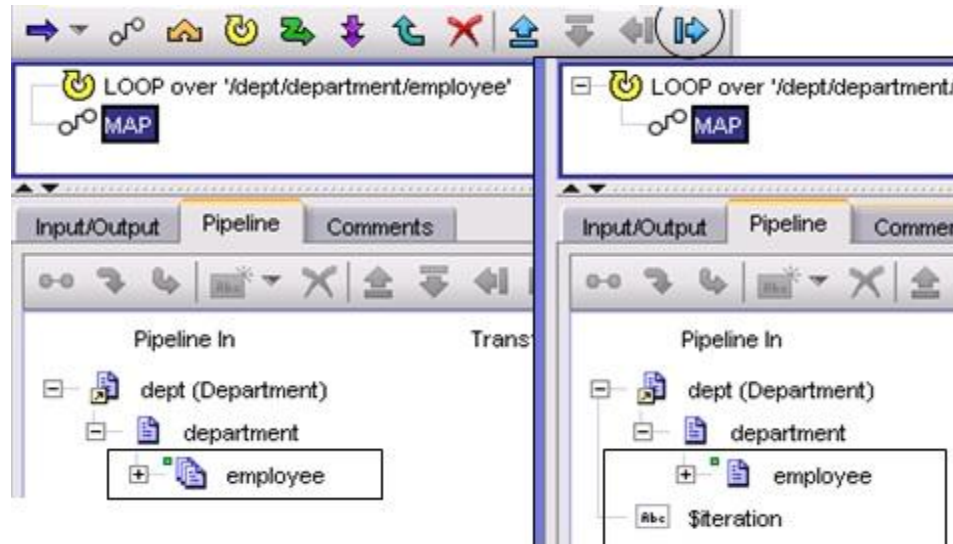
1) a. Select the appropriate folder (**services**). Create a new service by choosing **File | New | Flow Service | Next** | type the name “**loop_test**” | **Finish**.

- Select the **Input/Output** tab (Input pane) and select the **New variable** icon, , **Document Reference** and select the **otcSupport** package's **otcSupport.documents:Department** | **OK** | Name it **dept**.
- Select the **Input/Output** tab and click once in the **Output** panel white space. Click the “**New variable**” button and add an **Output String** and name it “**count**”.

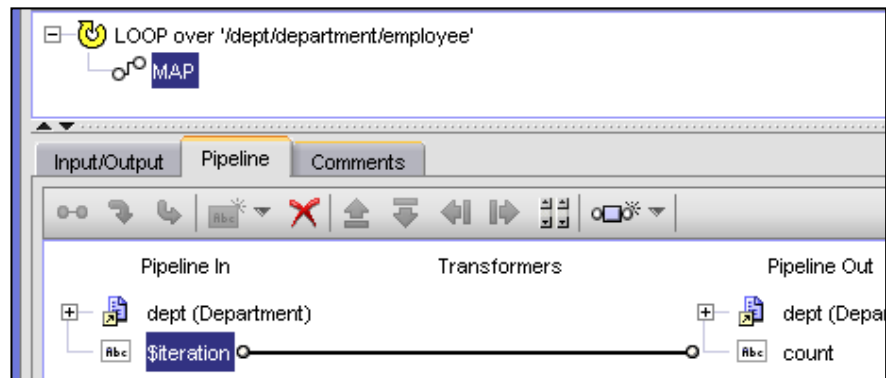
b. Click once in the **white space** of the Service (to establish focus, top panel) and add a **LOOP** (). Next, add a **MAP** (). Select the **Pipeline** tab. Expand the **dept** document and **right-click** the dept/department/employee document list and choose **Copy**. Select the LOOP step, and in the Loop's **Properties** panel **right-click** the **Input array** property, and paste **dept/department/employee**. Press the <CR> key (Carriage Return or Enter key) to confirm the **Input array**. You should see:




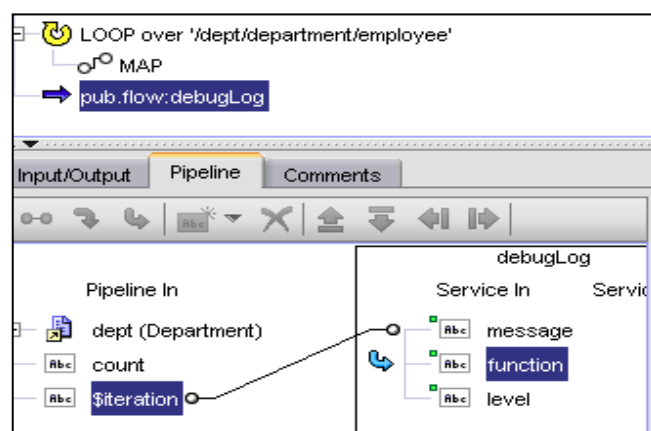
- Select the **Map**  step. Confirm that **employee** is a Document List. Use the **Right arrow** to shift the Map under the LOOP. Note that in your MAP statement, the **Pipeline In** has a field named **\$iteration** and that within this MAP statement under the LOOP, the **dept/department/employee** field appears as a Document and not a Document List (array) as shown:




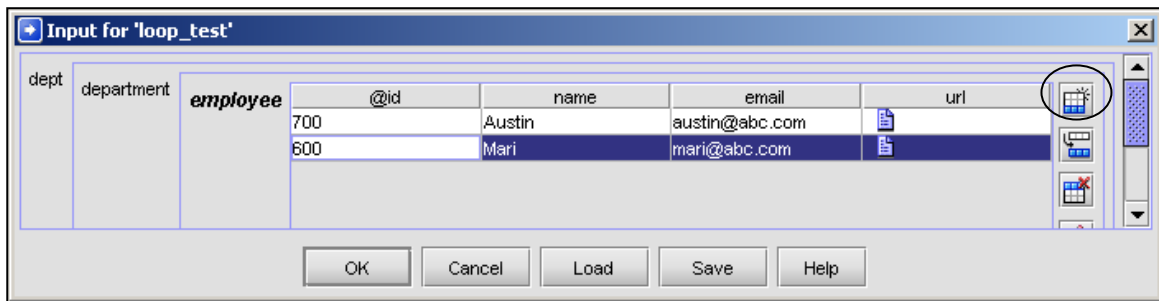
- In the **Pipeline** tab, map the Pipeline In **\$iteration** field to the Pipeline Out **count** field by selecting **\$iteration** and without releasing the mouse button, dragging a line to the target field **count**.



- Insert the **debugLog** built-in Service. It must be NOT be indented (use the Shift-Left icon, , arrow to unindent)! Map the Pipeline In **count** field to the **message** field in **debugLog** Service In. Also set the **function** field = <your name>. **Save**.



- 2). To run the service, choose **Test | Run** or click on the **Run** icon . Expand the input box, and use the **Add Row** button to add two employees. Type some data for each employee. Click the **OK** button. You should see a count of **2** in the Server Log.



dept	department	employee	@id	name	email	url
700				Austin	austin@abc.com	
600				Mari	mari@abc.com	




SEQUENCE Flow Step Exercise

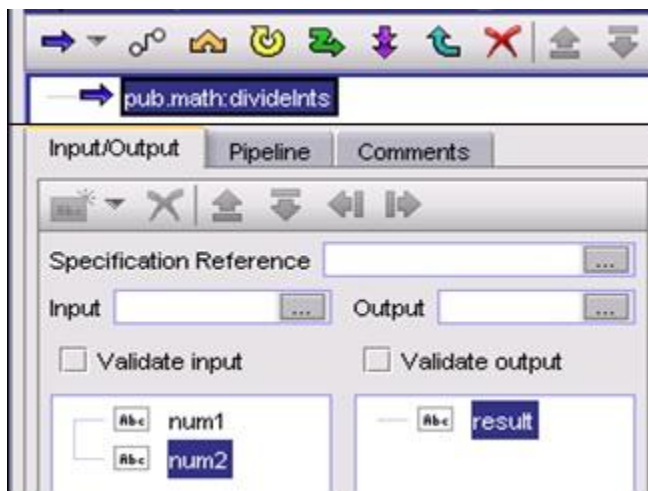
Task


You will now create and test a Try/Catch routine.

- 1) Create a service that will return an exception.
 - a. In the **services** subfolder, create a service **div_numbers** with **two** String inputs **num1** and **num2**, and an output String **result**.
 - b. Use a built-in service, **pub.math:divideInts**, to divide **num1** by **num2** and set the **result** variable.
- 2) **Test**. What happens when you divide by zero?
- 3) Create the **Try/Catch** service.
 - a. Create a service **sequence_test** with **two** String inputs **number1** and **number2**, and an output String **result**.
 - b. Use three **SEQUENCE** steps to create a Flow try/catch block to catch any errors thrown by invoking the **division service** created above.
 - c. Write the error message to the Server Log.
- 4) **Test** to see if divide-by-zero is properly handled.

Steps to create SEQUENCE flow step:

- 1) a. Select the appropriate folder (**services**) and create a new **service** by choosing **File | New | Flow Service | Next**, type the name “**div_numbers**”. **Finish**.
 - Select the **Input/Output** tab (Input pane) and select the **New variable** icon, , | **String** | name it “**num1**”. Again, select the **New variable** icon, , | **String** | name it “**num2**”.
 - Click once in the **Output** pane. Click the “**New variable**” button and add an output **String** and name it “**result**”.
- 1) b. Click the **Insert** button, , **Browse | WmPublic | pub | math | divideInts | OK**. You should see:






- 2) . **To run the service.** **Test | Run** or click on the **Run** icon . Provide integer numbers as inputs for **num1 = 6** and **num2 = 0**. **OK**. You should get the following error message:



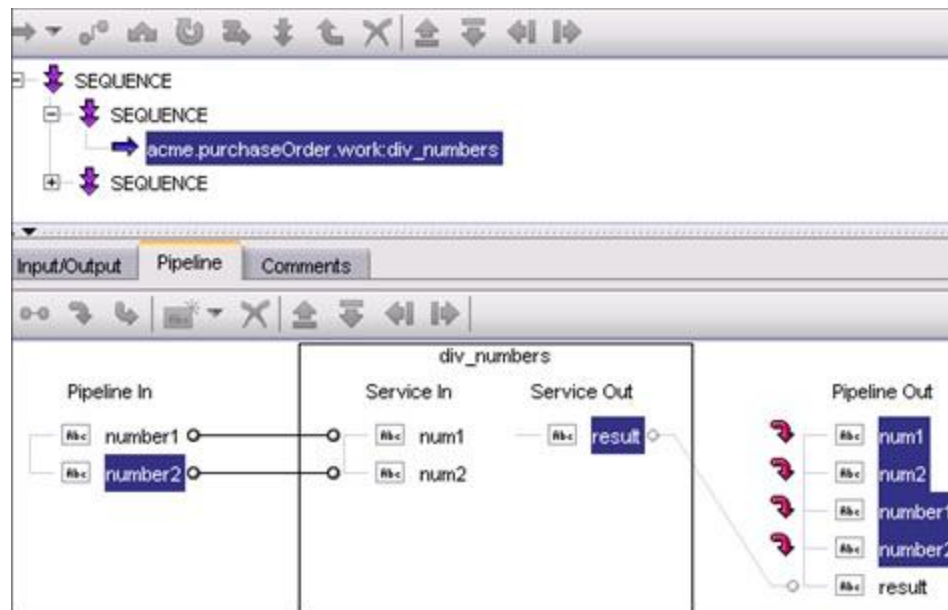
Now you need to write code to catch and display this message.

3) a. Select the appropriate folder (**services**) and create a new **service** by choosing **File | New | Flow Service | Next**, type the **name** “**sequence_test**” | **Finish**.

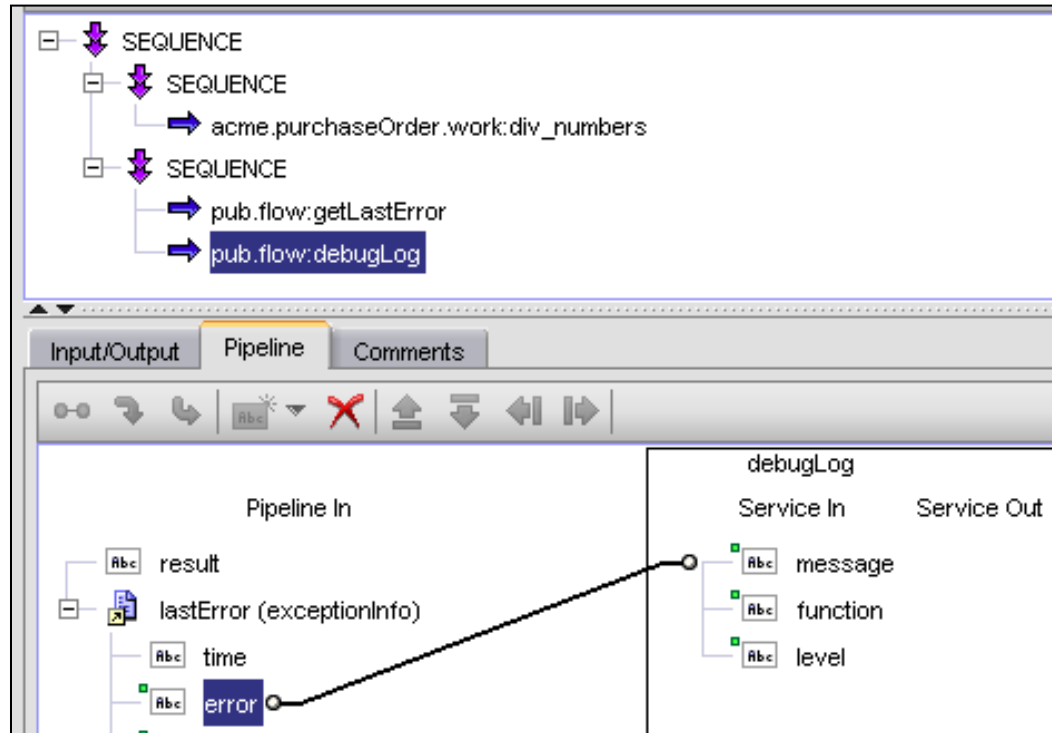
- Select the **Input/Output** tab (Input pane) and select the **New variable** icon, , | **String** | name it “**number1**”. Again, select the **New variable** icon, , | **String** | name it “**number2**”.
- Select the **Input/Output** tab (Output panel) and click the “**New variable**” button. Add **one** output String **result**.

3) b. In the Flow tab, insert three SEQUENCE () statements and indent the second and third under the first.

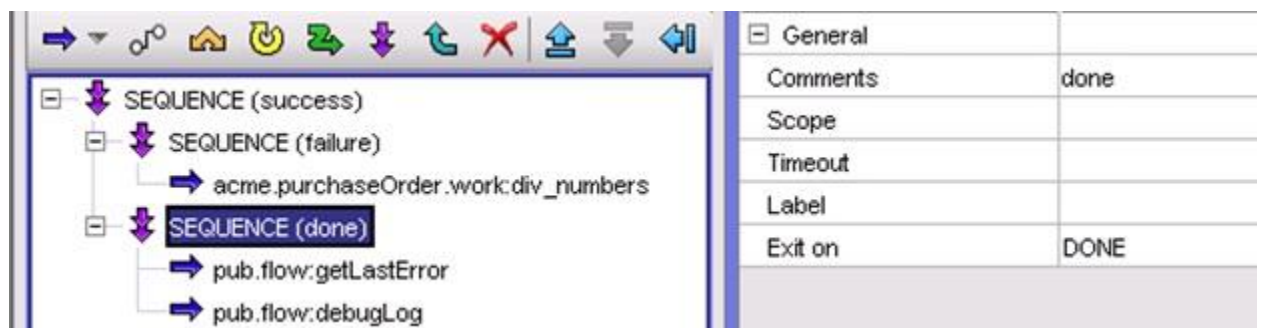
- Under the second SEQUENCE, insert and **indent** an INVOKE of the **OrderCustomer** package’s **OrderCustomer.purchaseOrder.services.div_numbers** service. Select the **Pipeline** tab. **Map** the **number1** and **number2** fields to **num1** and **num2** respectively. Save your services. **Drop** all fields in Pipeline Out EXCEPT the **result** variable as shown below:




3) c. Under the third SEQUENCE, insert and indent an INVOKE of the **WmPublic** package's **pub.flow:getLastError** service. Following this at the same level, insert an INVOKE of the **WmPublic** package's **pub.flow:debugLog** service. In the **debugLog** service, map **lastError/error** to **message**. Set the **function** field as before:



- Select the first SEQUENCE. In its **Properties** window, set the **Comments**=**"success"**, and **Exit-on** to **SUCCESS**.
- Select the second SEQUENCE. In its **Properties** window, set the **Comments**=**"failure"**, and leave the **Exit-on** set to the default value of **FAILURE**.
- Select the third SEQUENCE. In its **Properties** window, set the **Comments**=**"done"**, and **Exit-on** to **DONE**. You should now see:






- 4) . **To run the service.** **Test | Run** or click on the **Run** icon . Provide integer numbers as inputs for **number1=6** and **number2=0**. **OK**. Divide by zero! You should get the following error message in your server log:

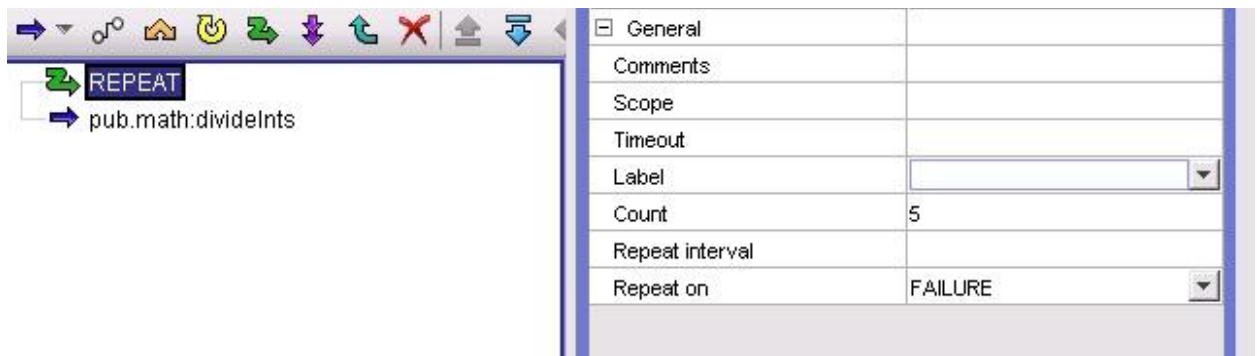
REPEAT Flow Step Exercise

Task

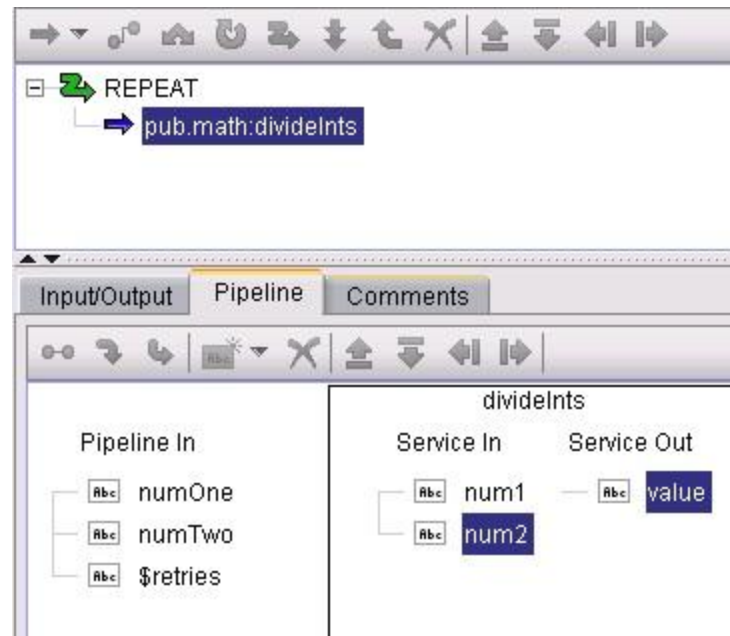
- 1) Create a flow service that simulates repeat flow step on a failure child step.
 - a. In the services subfolder, create a service **repeat_test** with **numOne** and **numTwo** as input.
 - b. Use the **REPEAT** flow statement and **pub.math.divideInts** flow step.
 - c. Check the execution of service.
- 2) **Run** it (Step by Step).

Steps to create REPEAT flow step:

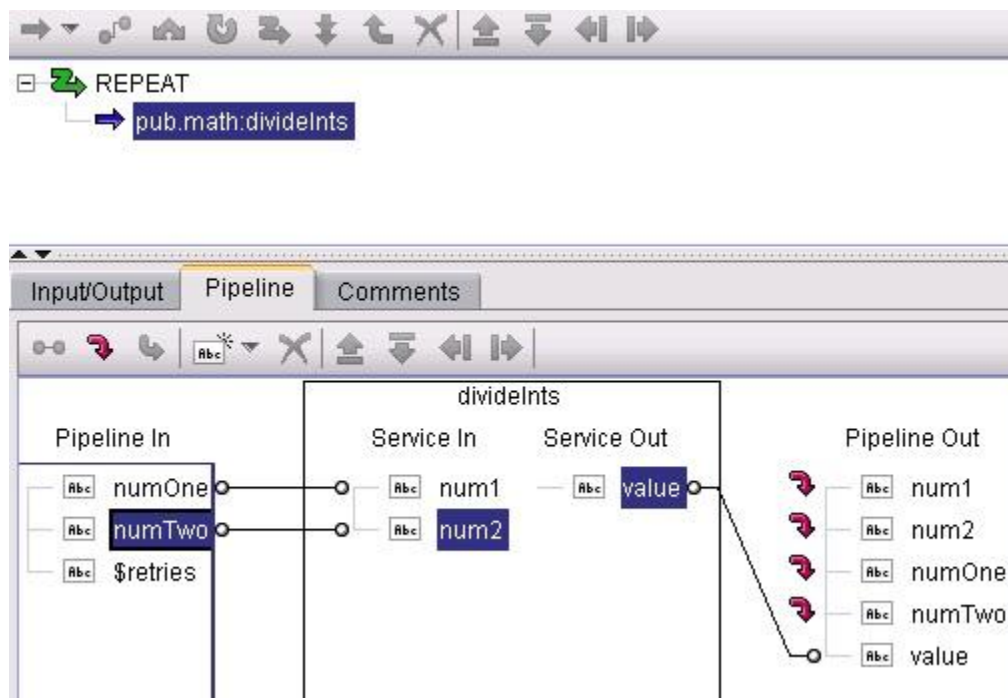
- 1) Select the appropriate folder (**services**) and create a new **service** by choosing **File | New | Flow Service | Next**, type the name “**sequence_test**” | **Finish**.
 - Select the **Input/Output** tab (Input pane) and select the **New variable** icon, , | **String** | name it “**numOne**”. Again, select the **New variable** icon, , | **String** | name it “**numTwo**”.
- 2) **Click once in the white space** of the Service (to establish focus, top panel) and add a **REPEAT** (). Next, add a flow step **pub.math.divideInts** . Select the REPEAT step, and in the REPEAT’s **Properties** panel, Set Repeat on to FAILURE and set Count to 5. You should see:




- 3) Select the **pub.math.divideInts** step. Use the **Right arrow** to shift the **pub.math.divideInts** under the Repeat. Note that in your **pub.math.divideInts** statement, the **Pipeline In** has a field named **\$retries**.



- 4) Select the **Pipeline** tab. Map the **numOne** and **numTwo** fields to **num1** and **num2** respectively. Save your services. **Drop** all fields in Pipeline Out EXCEPT the **value** variable as shown below:



- 5) To run the service. **Test | Run** or click on the **Run Step By Step** icon . Provide integer numbers as inputs for **numOne=5** and **numTwo=0**. **OK**. Check the variable **\$retries** in the output as you run it step by step.


Task 2: Run the same service by setting the **Repeat on** property of **REPEAT** flow step to **SUCCESS** and check the execution by providing **numOne=10** and **numTwo=2**.

EXIT Flow Step Exercise

Task

- 1) Create a flow service that simulates EXIT flow step.

Steps to create EXIT flow step:

- 1) Use the **loop_test** flow service from the exercise 3a, insert **EXIT**() flow step after **MAP** step with property **Exit from** set to **\$flow**.

Properties	
EXIT '\$flow'	
Property	Value
General	
Comments	
Label	
Exit from	\$flow
Signal	SUCCESS
Failure message	

- 2) Run the service and check the output variable **count**.

Task 2: Run the same service by setting **Exit from** property of **EXIT** flow step to **\$parent** and **\$loop** and check execution.