Progress Sonic

Sonic ESB Product Family

V 7.6 Tutorial

Batch to Real-time

Progress Sonic Workbench Online Help Tutorial Instructions in PDF Format

PROGRESS SOFTWARE

Progress® Sonic ESB® Product Family V7.6 Tutorial
Batch to Real-time

SonicMQ and Sonic ESB Product Families include files that are subject to the Netscape Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/NPL/. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is Mozilla Communicator client code, released March 31, 1998. The Initial Developer of the Original Code is Netscape Communications Corporation. Portions created by Netscape are Copyright 1998-1999 Netscape Communications Corporation. All Rights Reserved.

SonicMQ and Sonic ESB Product Families include versions 8.3 and 8.9 of the Saxon XSLT and XQuery Processor from Saxonica Limited (http://www.saxonica.com/) which is available from SourceForge (http://sourceforge.net/projects/saxon/). The Original Code of Saxon comprises all those components which are not explicitly attributed to other parties. The Initial Developer of the Original Code is Michael Kay. Until February 2001 Michael Kay was an employee of International Computers Limited (now part of Fujitsu Limited), and original code developed during that time was released under this license by permission from International Computers Limited. From February 2001 until February 2004 Michael Kay was an employee of Software AG, and code developed during that time was released under this license by permission from Software AG, acting as a "Contributor". Subsequent code has been developed by Saxonica Limited, of which Michael Kay is a Director, again acting as a "Contributor". A small number of modules, or enhancements to modules, have been developed by other individuals (either written specially for Saxon, or incorporated into Saxon having initially been released as part of another open source product). Such contributions are acknowledged individually in comments attached to the relevant code modules. All Rights Reserved. The contents of the Saxon files are subject to the Mozilla Public License Version 1.0 (the "License"); you may not use these files except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

SonicMQ and Sonic ESB Product Families include software developed by IBM. Copyright 1995-2002 and 1995-2003 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

February 2008

Batch to Real-time tutorial

The Batch to Real-time tutorial demonstrates how rapidly you can use Sonic Workbench to prototype and implement an enterprise SOA application to process batch files in real time.

Transferring files between systems is a common practice in most organizations. Typical extract, transform, and load (ETL) processes include batch to real-time processing, continuous pipeline processing, straight through processing, and file mediation. As data volumes increase and batch windows decrease, using ETL processes can cause bottlenecks.

With service-oriented architectures (SOA), you break batch files into discrete transactions that flow directly to consumers over standard protocols. SOA decouples applications–not just in time as batch processing does–but also in location, platform choice, and implementation method. SOA uses standard message formats to provide visibility into the information as it moves through the enterprise. Sonic ESB provides a pre-built SOA for flexible integration and re-use of business applications within a SOA application.

In the Batch to Real-time tutorial, you prototype and develop a SOA application that processes batch purchase orders in real time. You can go through the entire tutorial step by step, or, if you prefer, you can go directly to running and testing the fully implemented ESB process using the completed sample process included in Sonic Workbench.

It takes approximately one hour and 15 minutes to work through the entire Batch to Real-time tutorial. The following times for each section of the tutorial are merely estimates:

- Preparing to develop the Batch to Real-time sample application — 15 minutes
- Phase 1: Implementing the first file drop operation — 20 minutes
- Phase 2: Implementing validation and XML splitter operations — 15 minutes
- Phase 3: Implementing the second file drop operation— 20 minutes
- Tracking and debugging the Batch to Real-time sample application — 15 minutes

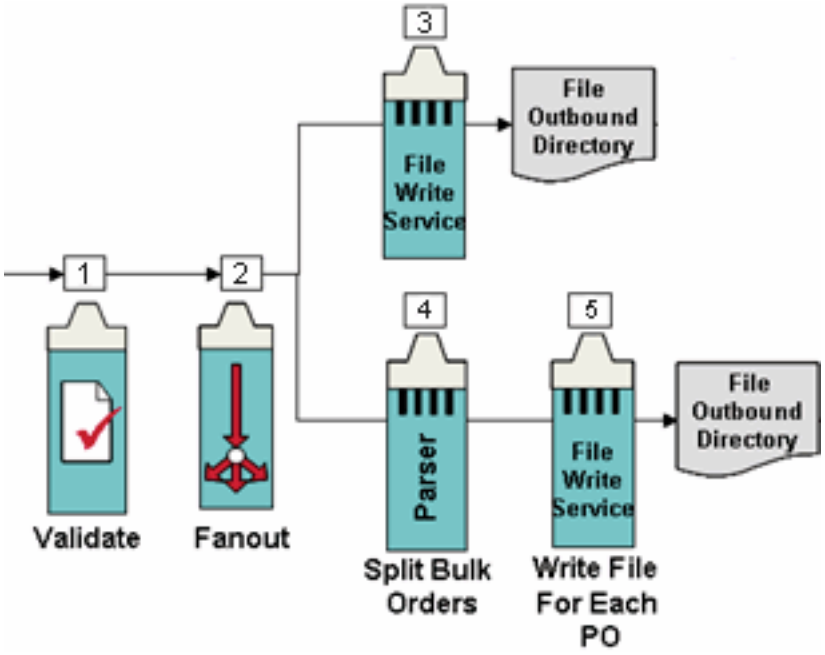
The Batch to Real-time tutorial is available in the following formats:

- Online help — Click the Tutorials link on the Sonic Workbench Welcome page or find the tutorial in the *Progress Sonic ESB Product Family: Developer's Guide* (Sonic Workbench online help) under "Progress Sonic ESB Samples and Tutorials".
- PDF — Click the link from the Documentation page.
- Multimedia demonstration — Click the link from the Documentation page.

Next, see the Batch to Real-time purchase order processing scenario.

Batch to Real-time purchase order processing scenario

In this tutorial you implement a typical batch to real-time purchase order processing scenario. This scenario demonstrates how messages carry purchase order data between services:



This scenario has the following steps:

Step 1. Bulk purchase order files enter the process and go to a service that validates that the bulk purchase order files conform to the organization's standard format.

Step 2. A fanout distributes bulk purchase order files to two branches for processing.

Step 3. In one branch, a service writes the bulk purchase order files to an outbound directory, for example, for auditing purposes

Step 4. In the other branch, a service splits each bulk purchase order file into individual files, one file for each purchase order.

Step 5. Another service writes the resulting purchase order files to a different outbound directory.

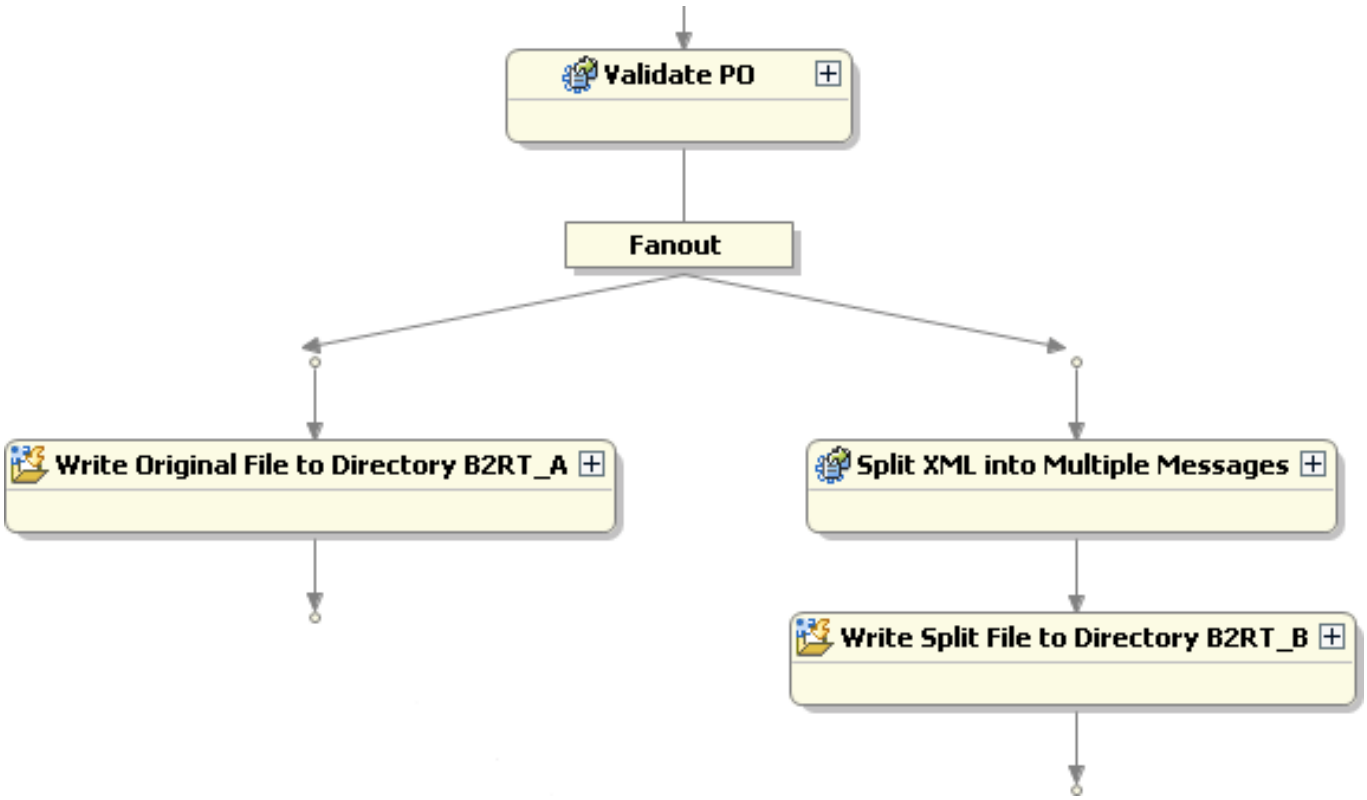
Next, look at the Batch to Real-time sample application that you will develop in this tutorial to implement this scenario.

Batch to Real-time sample application

With the Sonic ESB process model, you integrate services by designing an itinerary—a series of steps comprised of services that route messages from service to service. Messages carry the purchase order data between the steps in the itinerary.

In this tutorial, you develop the FileTransfer ESB process to implement the Batch to Real-time purchase order processing scenario:

1. ESB messages containing bulk purchase orders enter the FileTransfer process. The **Validate PO** step confirms that the purchase order files conform to the organization's standard format.
2. The messages then go to a **Fanout**, which sends the messages to two branches.
3. In the left branch, the **Write Original File to Directory B2RT_A** step drops the bulk purchase order files in a specified directory.
4. In the right branch, the **Split XML into Multiple Messages** step splits the bulk purchase order files into individual purchase order files.
5. Messages containing the resulting individual purchase order files go to the **Write Split File to Directory B2RT_B** step, which drops the files in a different directory.



Next, see how you use phased implementation to develop the Batch to Real-time sample application.

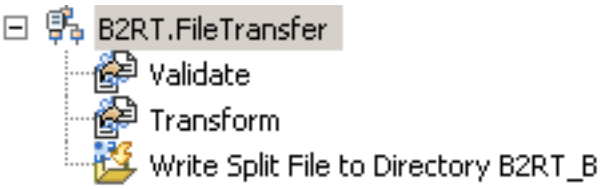
Phased implementation of the Batch to Real-time sample application

The Batch to Real-time tutorial demonstrates top-down design using phased implementation. This approach enables you to develop the itinerary for your business process using prototype steps. Then you replace the prototype steps with services that implement actual operations. You use phased approach to implement the Batch to Real-time purchase order processing scenario from the top down by laying out the steps in an ESB process itinerary.

In this tutorial, you develop the Batch to Real-time sample application in three implementation phases. The following summary shows an outline view of the FileTransfer process at the end of each phase:

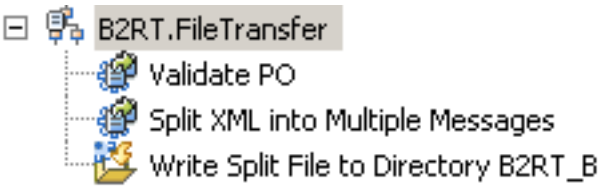
Phase 1. Implement the first file drop operation

- Create the FileTransfer process and use an enterprise integration template to add the first steps.
- Implement a file drop operation to drop bulk purchase order files in a specified directory.
- Test the FileTransfer process and confirm that the bulk purchase order file is dropped in the target directory.



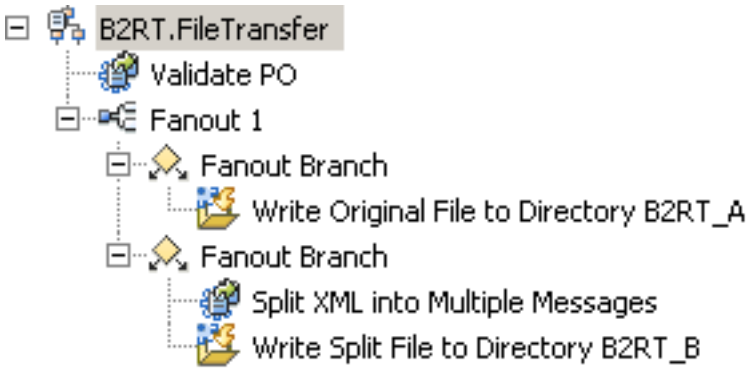
Phase 2. Implement validation and XML splitter operations

- Implement the validation operation to validate the bulk purchase order files against an XML schema.
- Implement the XML splitter operation to split each bulk purchase order file into individual purchase order files.
- Test the enhanced File Transfer process and confirm that the resulting individual purchase order files are dropped in the target directory.



Phase 3. Implement the second file drop operation

- Add a fanout to create a second branch in the FileTransfer process.
- Implement a second file drop operation to drop the bulk purchase order files in another, specified directory.
- Test the completed FileTransfer process. Confirm that the bulk purchase order file is dropped in one target directory and the individual purchase order files are dropped in the other target directory.



Next, prepare to develop the Batch to Real-time sample application.

Preparing to develop the Batch to Real-time sample application

Before developing the Batch to Real-time sample application, you must:

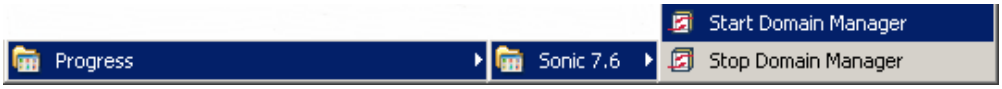
1. Start Sonic Workbench.
2. Import the Batch to Real-time sample project.
3. Examine the Batch to Real-time project.

Begin by starting Sonic Workbench.

Starting Sonic Workbench

To start Sonic Workbench:

1. Select **Start > Programs > Progress > Sonic 7.6 > Start Domain Manager**:

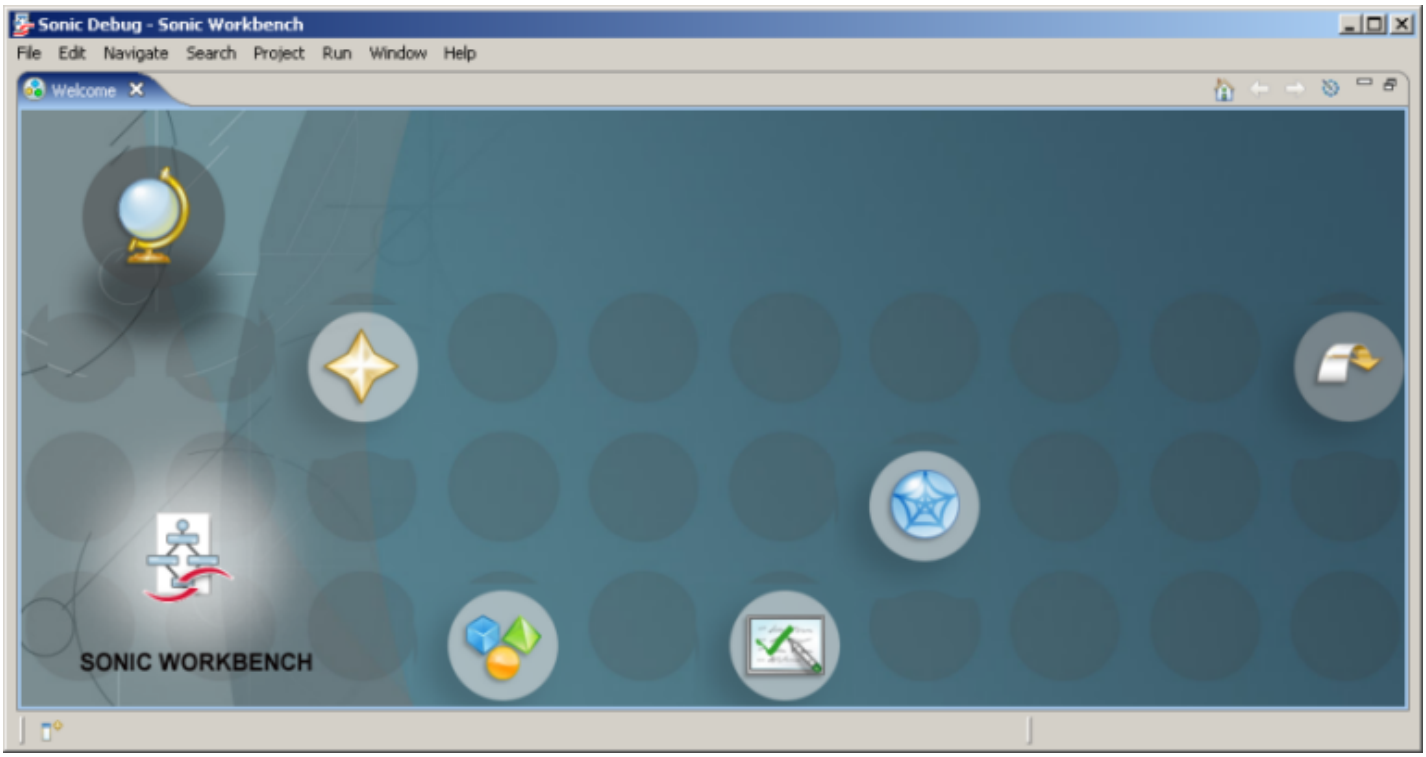
   

   A console window opens showing that Sonic Workbench is starting the domain manager, configuration repository, and development containers.

2. Select **Start > Programs > Progress > Sonic 7.6 > Workbench**:

   The Sonic Workbench Welcome screen opens:

3. Click the icons on the Welcome screen to see how you can access the following information:

	View an overview of the features of Sonic Workbench, Eclipse, and Java development.
	Find out what is new in this release of Sonic Workbench.
	Link to the documentation on the sample applications for Sonic ESB, Sonic BPEL Server, Sonic Database Server, and Sonic XML Server.
	Link to the documentation on the tutorials for Sonic ESB and Sonic BPEL Server.
	Access web resources, including the home pages for the Progress Sonic products, tech support, Eclipse updates, and PSDN (Progress Software Developers Network).

4. Click  to close the Welcome screen and start using Sonic Workbench.

Note: You can reopen the Welcome screen by selecting **Help > Welcome**.

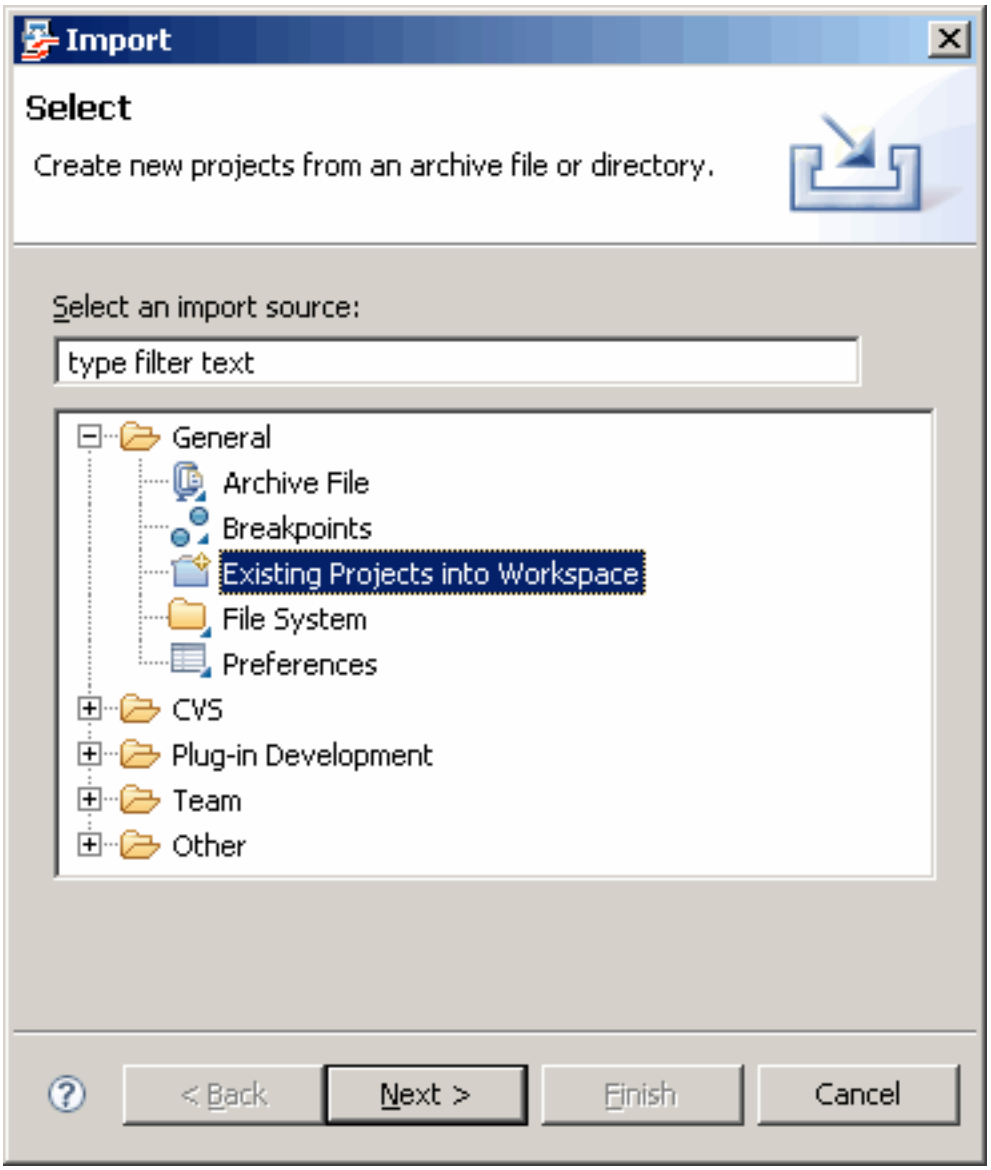
Next, import the tutorial projects.

Importing the Batch to Real-time sample project

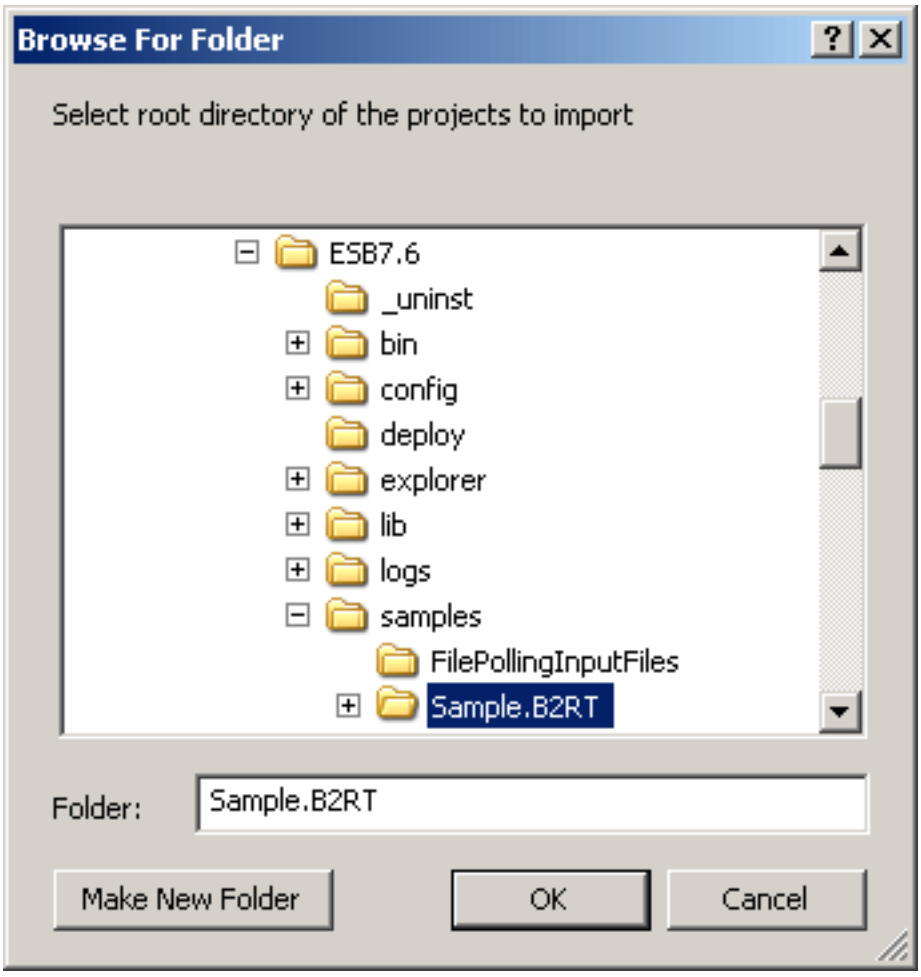
To import the sample project for the Batch to Real-time tutorial:

1. After closing the Welcome screen, you are ready to use Sonic Workbench in the Sonic Design perspective:

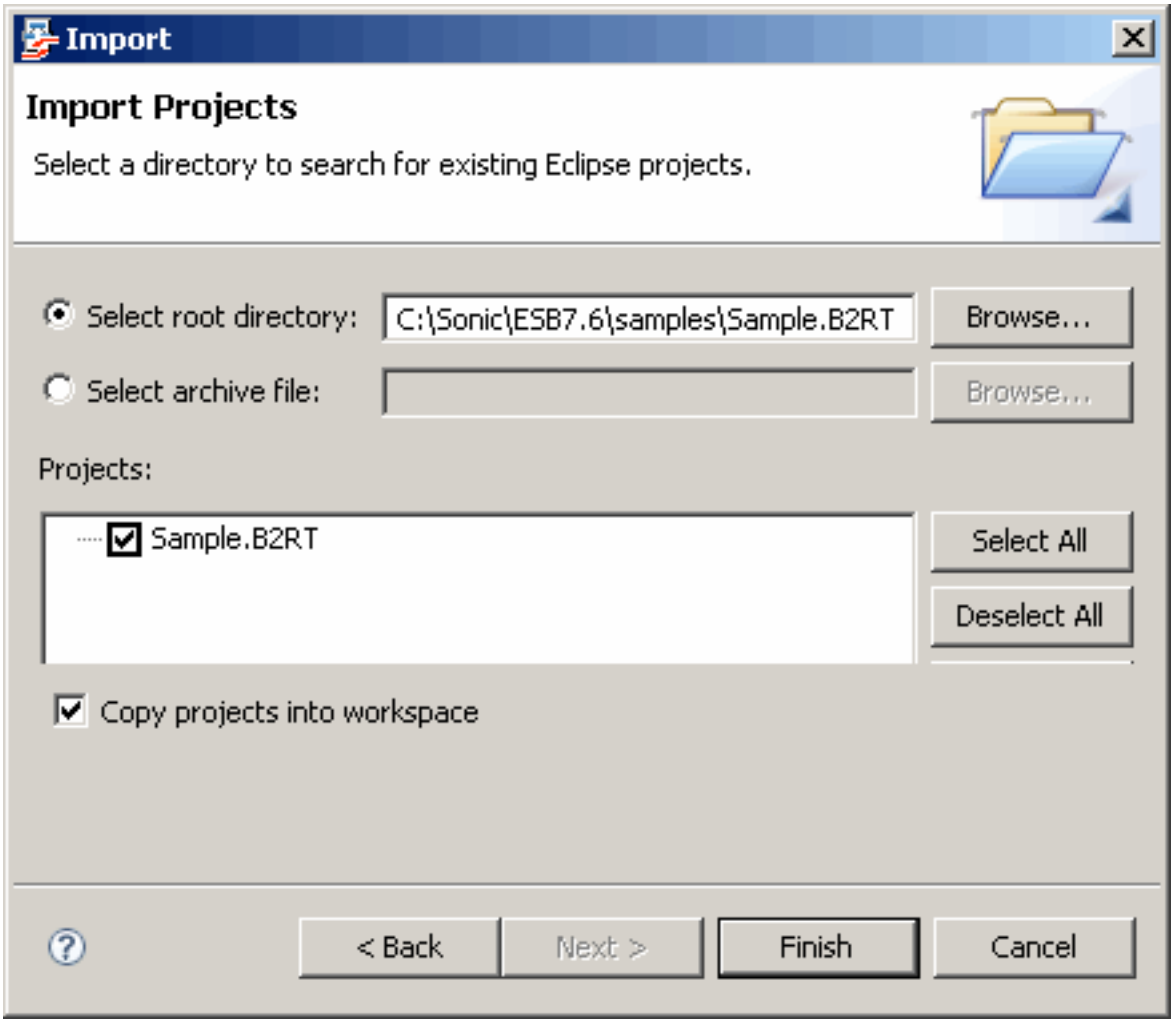
2. Select **File > Import**. The **Select** page of the **Import** wizard opens.
3. Select **General > Existing Projects into Workspace**:



4. Click **Next**. The **Import Projects** page of the **Import** wizard opens. Choose **Select root directory** and click **Browse**. The **Browse for Folder** dialog box opens.
5. Select the **Sample.B2RT** folder under **Sonic > ESB7.6 > samples**:



6. Click **OK**. The **Import Projects** page of the **Import** wizard opens.
7. Be sure the Sample.B2RT project is checked and check **Copy projects into workspace** (this option prevents you from changing the original project if you modify the imported project):



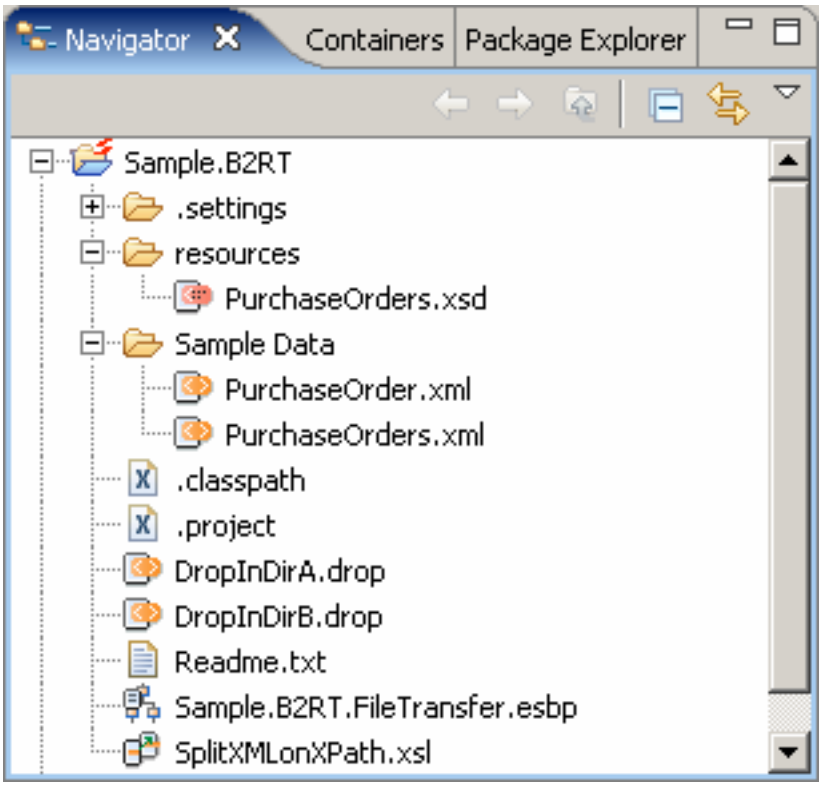
8. Click **Finish**. Sonic Workbench loads the Batch to Real-time sample project.

Next, examine the Batch to Real-time project.

Examining the Batch to Real-time sample project

After you import the Batch to Real-time sample project, you upload the project and examine its files:

1. Go to the Navigator view. Select the Sample.B2RT project and select **Project > Upload all** from the menu bar to upload the project.
2. Click **OK** to confirm the uploading.
3. Expand the Sample.B2RT folder and the \resources and \Sample Data subfolders to view the files in the Sample.B2RT project:



4. You can double-click a file to view it in the appropriate Sonic Workbench editor.

You can learn more about the files in the Batch to Real-time sample project now, or wait until you work with them in the tutorial.

Now you are ready to develop the Batch to Real-time sample application.

Developing the Batch to Real-time sample application

To facilitate developing the Batch to Real-time sample application, this tutorial is divided into three implementation phases and an optional tracking and debugging section. Each phase or section builds on the preceding one. You can stop developing and testing the application at any time and come back to it. Just be sure to save the files you are working on.

After preparing, you develop the sample application in three phases:

- **Phase 1.** Create an initial prototype ESB process and then use an enterprise integration template to add the first prototype steps. Add a file drop step to drop files in a specific directory. Create a scenario and test the process. Then confirm that the file drop step dropped the bulk purchase order file in the target directory. (Approximately 20 minutes.)
- **Phase 2.** Implement the validation operation and add an XML splitter step. Test the enhanced process to confirm that the file drop step dropped three individual purchase order files in the target directory. (Approximately 15 minutes.)
- **Phase 3.** Add a fanout to create a second branch. Implement a second file drop operation. Then test the completed process to confirm that one file drop step dropped the bulk purchase order file in its target directory and the other file drop step dropped three purchase order files in its target directory. (Approximately 20 minutes.)

Then, optionally **track and debug** the completed process. (Approximately 15 minutes.)

Start with phase 1: Implementing the first file drop operation.

Note: If you do not want to develop all phases of the sample application yourself, you can run and test the complete `Sample.B2RT.FileTransfer.esbp` process that is in the sample project you imported. This sample ESB process is the same as the process you develop in this tutorial. Open the sample process by double-clicking the file in the Navigator view and follow the instructions to test and track and debug the sample application.

Phase 1: Implementing the first file drop operation

After preparing, you develop the sample application in three phases. In phase 1 you do the following:

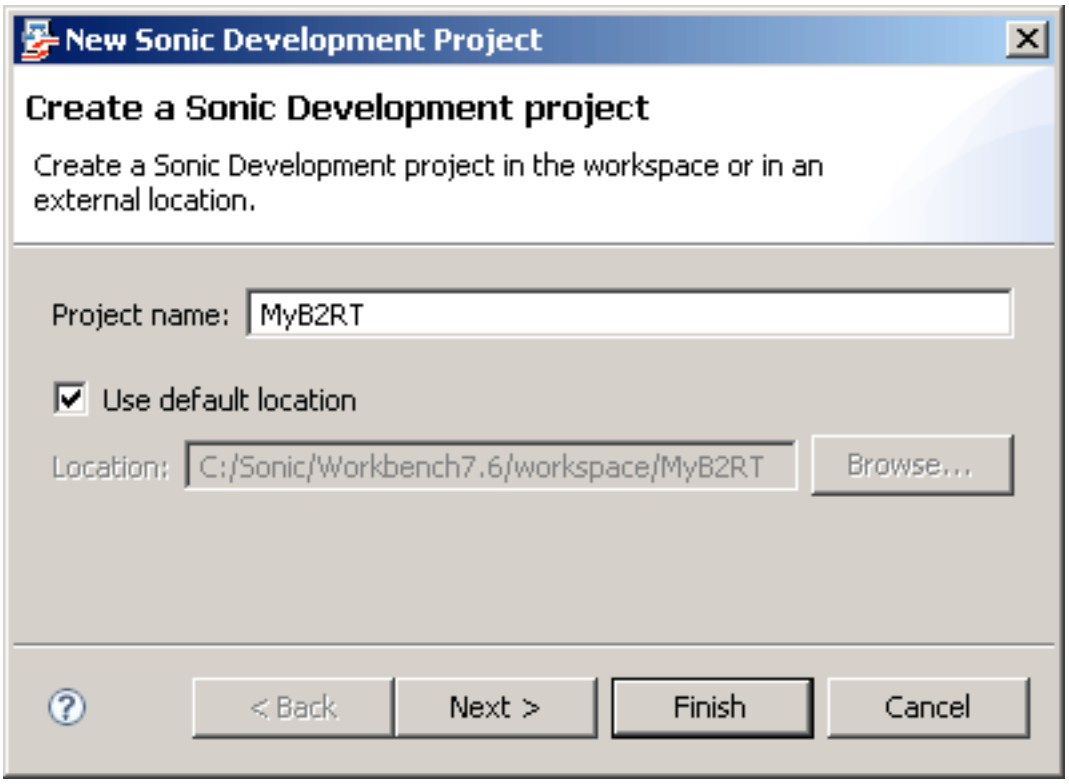
1. Create a new project — Create your own Sonic development project and copy the sample data into your project.
2. Create an ESB process — Create the FileTransfer process.
3. Add the first steps to the FileTransfer process — Use the VETO enterprise integration pattern template to add the first steps to the FileTransfer process.
4. Add a file drop step — Use the DropFileInDirectory enterprise integration template to add a File Drop service to drop files in a specified directory.
5. View the properties of the file drop step — Observe the configuration document used by the File Drop service.
6. Modify the configuration document — Specify the name of the purchase order file and the target drop directory in the configuration document.
7. Create a scenario — Create a scenario that sends the bulk purchase order file through the FileTransfer process.
8. Test the FileTransfer process — Confirm that the bulk purchase order file is dropped in the target directory.

Start by creating a new project.

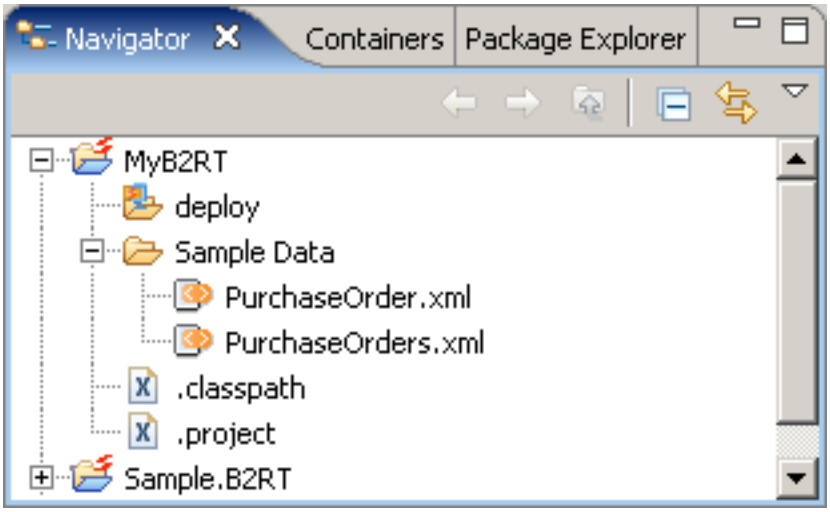
Creating a new project

After preparing to run the tutorial, you create your own Sonic development project for running the B2RT tutorial:

1. In Sonic Workbench, select **File > New > Sonic Development Project**. The **New Sonic Development Project** wizard opens.
2. Enter *MyB2RT* as the name of your project:



3. Accept the default location and click **Finish**. Sonic Workbench creates the new project.
4. In the Navigator view, select the Sample Data folder under the Sample.B2RT folder, right-click, and select **Copy**.
5. Select your MyB2RT folder, right-click, and select **Paste**. Sonic Workbench adds the Sample Data folder to your project:

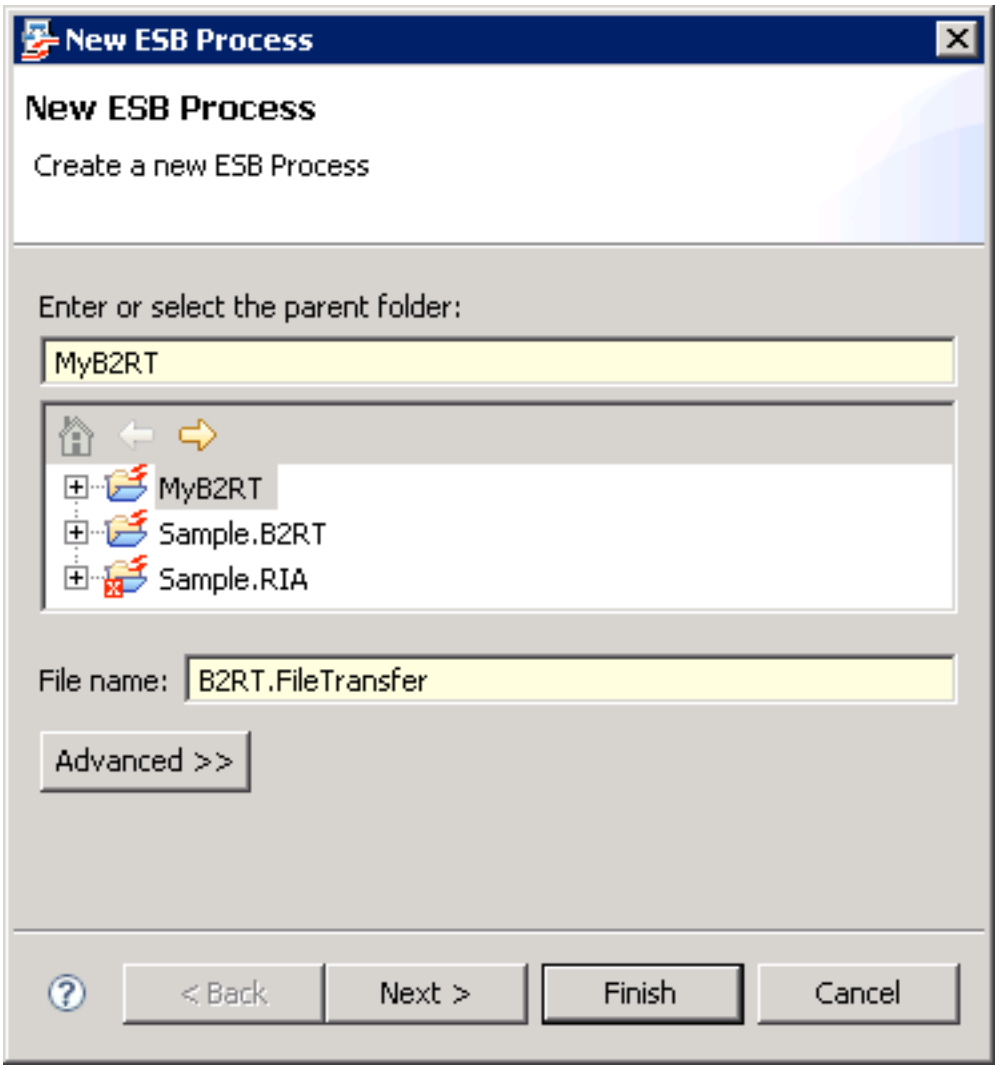


Next, create an ESB process.

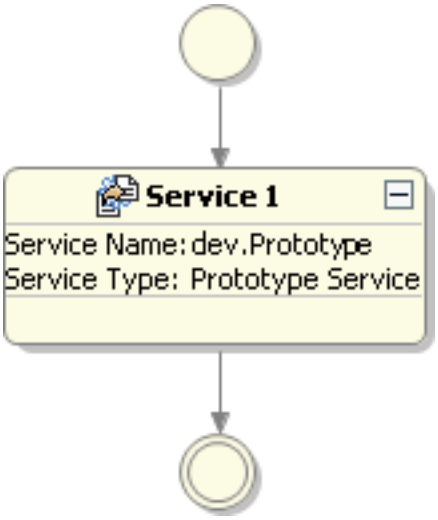
Creating an ESB process

After creating the project, you are ready to create the ESB process you will use to route messages carrying purchase order data from service to service:

1. Select **File > New > ESB Process**. The **New ESB Process** wizard opens.
2. Select the MyB2RT project as the parent folder.
3. Enter *B2RT.FileTransfer* as the name of the ESB process. (You use a different name from Sample.B2RT used in the B2RT sample project to avoid over-writing, in case you have both processes running at the same time.)



4. Click **Finish**. Sonic Workbench creates the new FileTransfer ESB process, which opens in the Process page in the ESB Process editor:
5. Expand the **Service 1** step and observe that it is a Prototype service:



The new FileTransfer process is a pass-through prototype ESB process that uses the Prototype service to return the input message as the output. You can use a prototype process to quickly create and run an ESB Process before implementing actual services.

Next, start adding steps to the FileTransfer process.

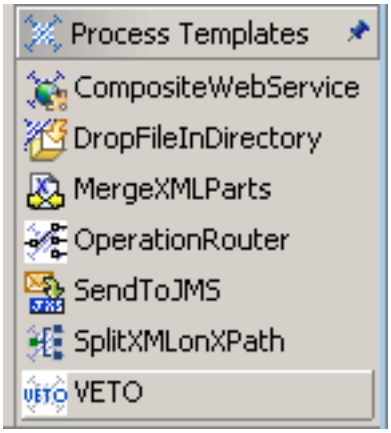
Adding the first steps to the FileTransfer process

The prototype FileTransfer ESB process that you created, contains one step, a Prototype service that simply passes messages through the process. Sonic Workbench provides the Prototype service so you can quickly create and run an ESB Process before implementing actual services.

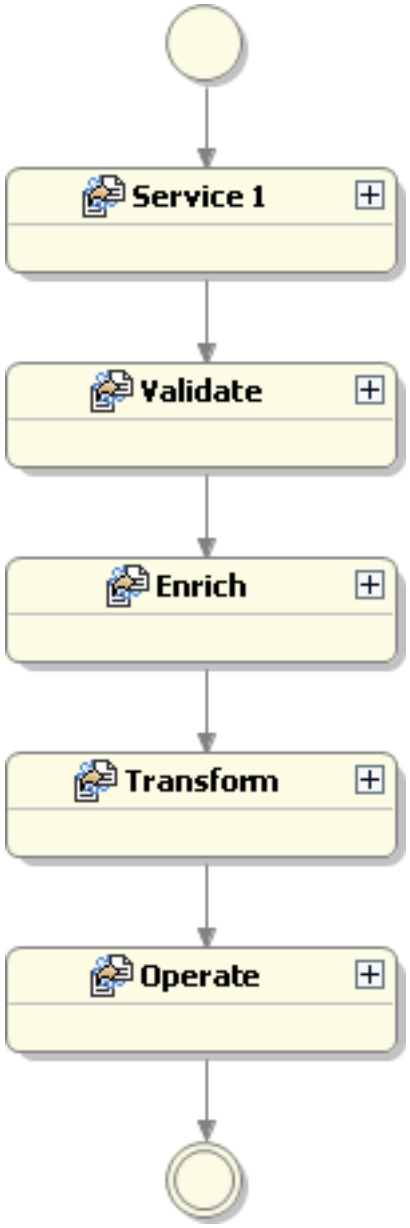
Sonic Workbench provides templates that are standard enterprise information patterns that you can use as skeletons to start developing an ESB process. You are now going to use the VETO enterprise integration pattern template to lay out four steps. The **V**alidate, **E**nrich, **T**ransform, and **O**perate steps are Prototype services that you will replace with actual services as you implement the operations in the File Transfer process.

To add the steps from the VETO template:

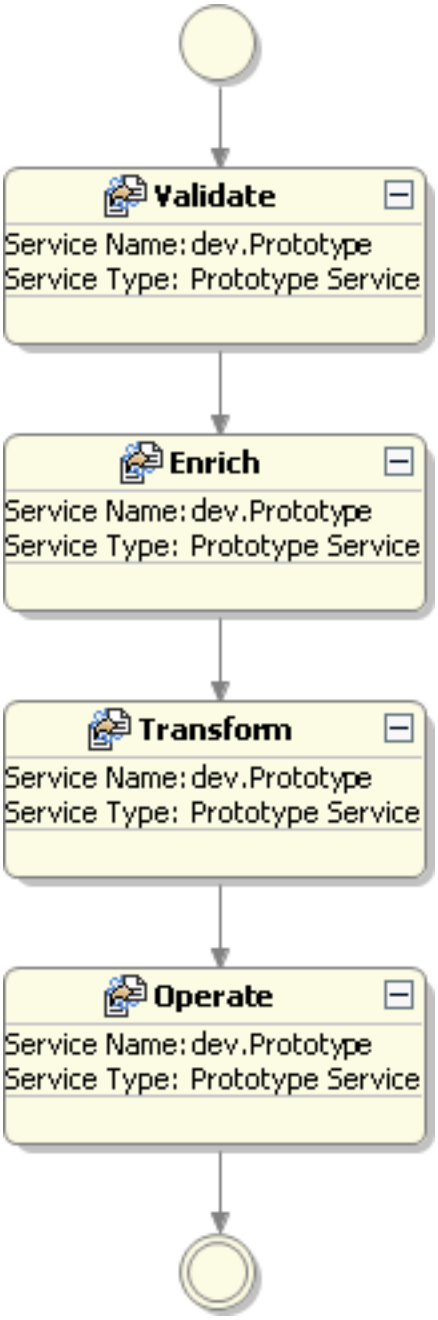
1. View the various steps, service types, and process templates available in the Palette (on the right side of the Process page).
2. Add the VETO template to the process:
    a. Select the **VETO** template in the **Process Templates** section of the Palette:



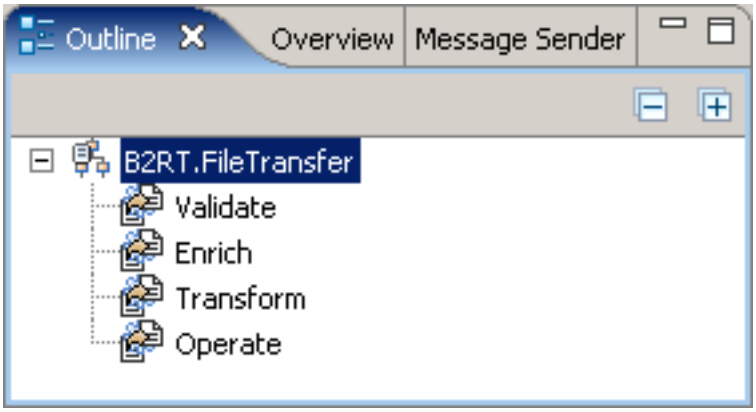
    b. Drag the VETO template onto the connecting line between the **Service 1** step and the terminal circle and release it. Your process should now resemble:



3. Select the unnecessary **Service 1** step, right-click, and select **Delete**. The FileTransfer process now contains just the four steps from the VETO template.
4. Expand the steps and observe that each step is a Prototype service:

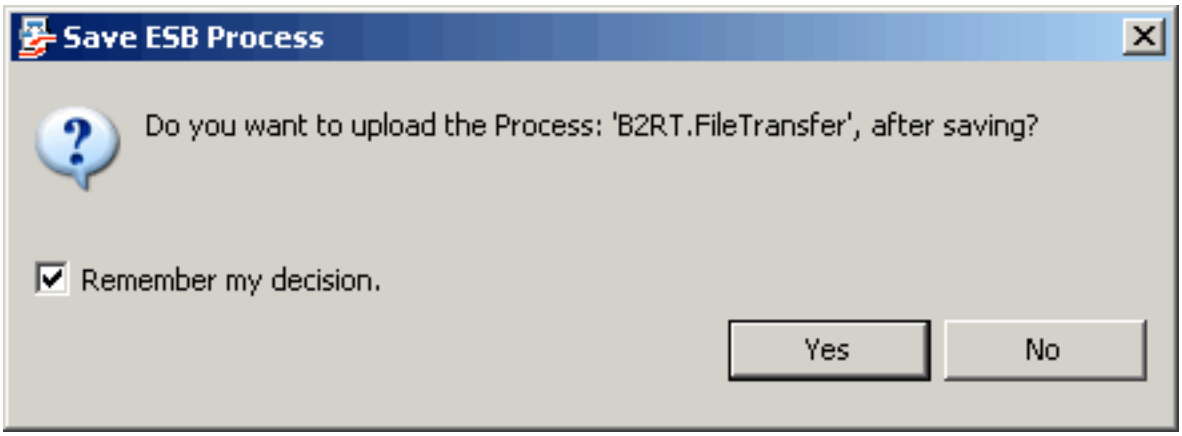


5. Go to the Outline view to see an outline of the ESB process, (Phased implementation shows outlines of the FileTransfer process after each phase):



6. Click **Save** to save the FileTransfer process.

7. The **Save ESB Process** dialog box prompts you to upload the process after saving. Check the box next to **Remember my decision** so Sonic Workbench will automatically upload the process every time you save. Click **Yes**:

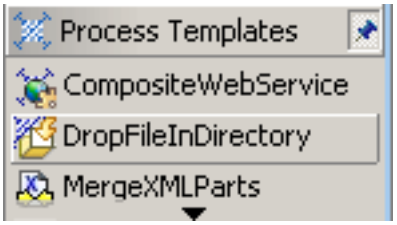


Next, add a file drop step.

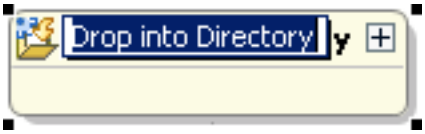
Adding a file drop step

Operate is the fourth step in the VETO template you added to the FileTransfer process. Like all the steps in the VETO template, **Operate** is a Prototype service. You use another enterprise integration pattern template to replace this service with a File Drop service to drop files in a specified directory:

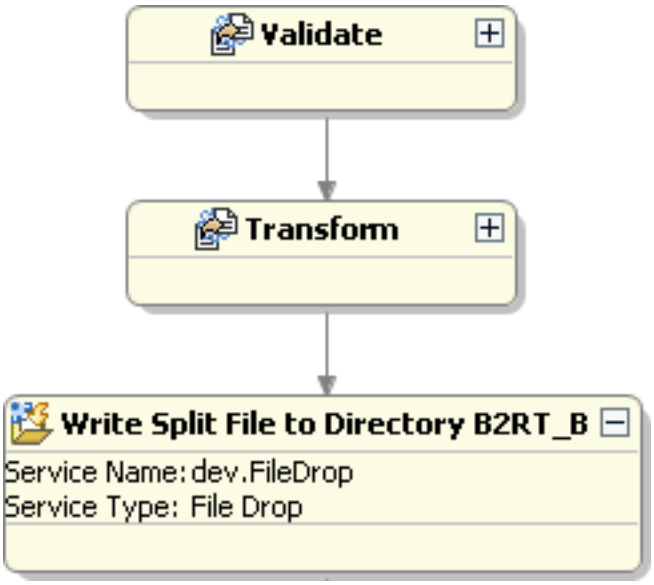
1. Go back to the Process page and select the **DropFileInDirectory** template from the **Process Templates** section of the Palette:



2. Drag the **DropFileInDirectory** template onto your process below the **Operate** step. The ESB Process editor adds the **Drop into Directory** step.
3. Select the **Drop Into Directory** step. Then click the name on the step, **Drop into Directory**, so you can rename it:



4. Change the name to **Write Split File to Directory B2RT_B**.
5. Delete the **Operate** step, since you are replacing this step with the File Drop service.
6. Since you will simply pass the purchase order data through and not enrich it, you can delete the **Enrich** step.
7. The FileTransfer process now has three steps. Expand the **Write Split File to Directory B2RT_B** step and observe that it is a File Drop service:



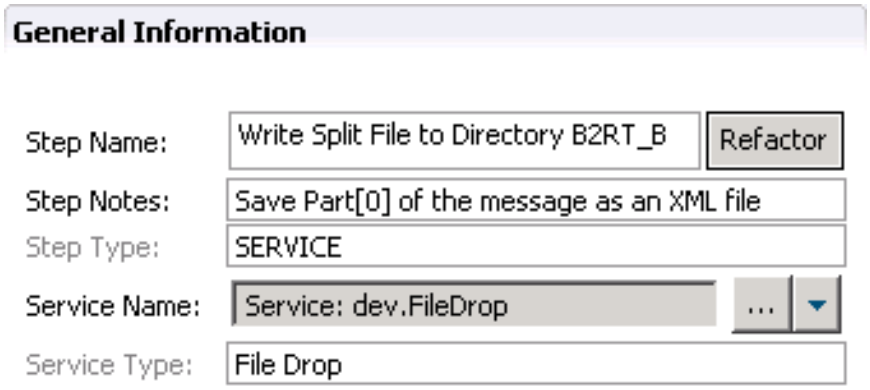
8. Save the modified FileTransfer process.

Next, view the properties of the file drop step.

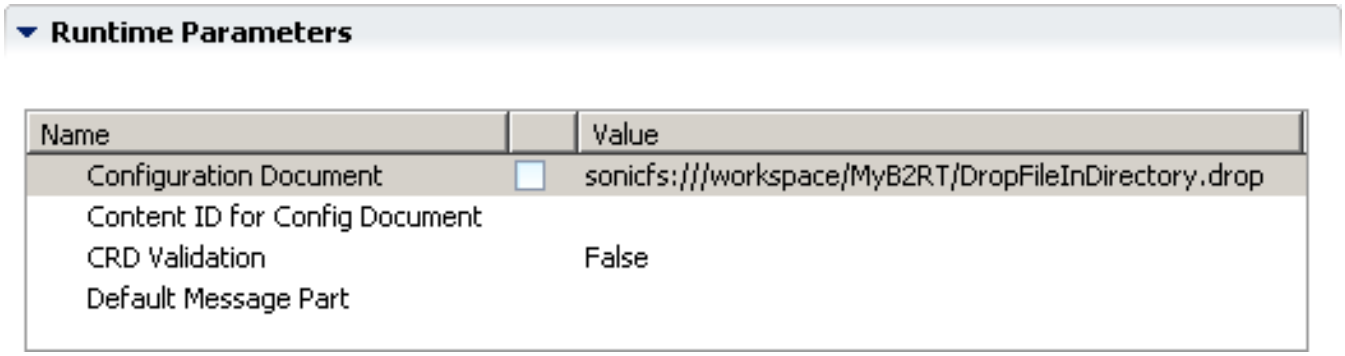
Viewing the properties of the file drop step

The third step in the FileTransfer process is the file drop step that you added. You can view the properties of this step in its Service page:

1. With the FileTransfer process open in the Process page, double-click the **Write Split File to Directory B2RT_B** step to open the **Service** page.
2. In the **General Information** section, observe that the **Service Type** indicates that this is a File Drop service:



3. When you used the DropFileInDirectory template to add the file drop step, Sonic Workbench added a configuration document to the project. In the **Runtime Parameters** section, observe that the **Configuration Document** for this File Drop service is `DropFileInDirectory.drop`:

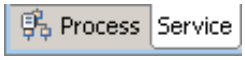


Next, modify the configuration document.

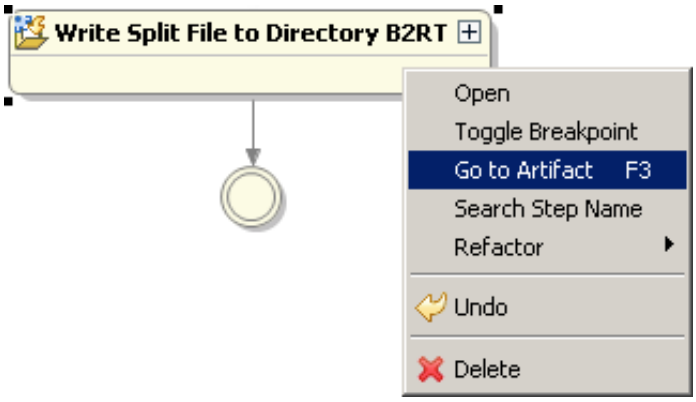
Modifying the configuration document

The **Write Split File to Directory B2RT_B** step is a File Drop service, which uses a configuration document for its runtime parameters. After viewing the properties of the file drop step, you are going to modify the configuration document:

1. Return to the Process page by clicking the Process tab on the lower edge of the page:

2. Select the **Write Split File to Directory B2RT_B** step, right-click, and select **Go to Artifact** (or simply click **F3**):

   The `DropFileinDirectory.drop` file opens in the XML editor.

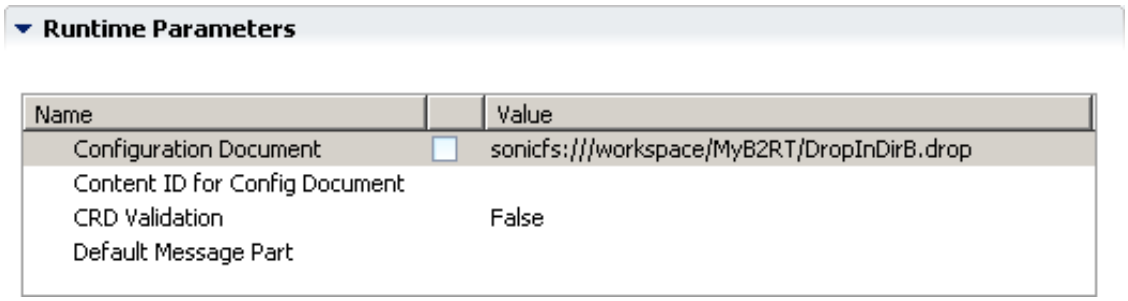
3. Specify the name of the Purchase order file. Change `FILENAME.xml` to `DroppedPO.xml`:

   ```
   <filename type="GENERATED">DroppedPO.xml</filename>
   ```

4. Specify the target directory for dropping the purchase order files. Change `FILE_DROP_DIRECTORY` to `FileDropOutputFiles\B2RT_B`:

   ```
   <dropDirectory>C:\Sonic\ESB7.6\samples\FileDropOutputFiles\B2RT_B</dropDirectory>
   ```

5. Save and close the `DropFileinDirectory.drop` file.
6. Go to the Navigator view and select the `DropFileInDirectory.drop` file. Right-click, select **Rename**, and change the name of the file to `DropInDirB.drop`.
7. Go back to the Service page in the ESB Process editor and change the name of the **Configuration Document** in the **Value** field to `DropInDirB.drop`:

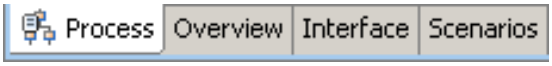
8. Save the FileTransfer process.

Next, create a scenario to test the FileTransfer process.

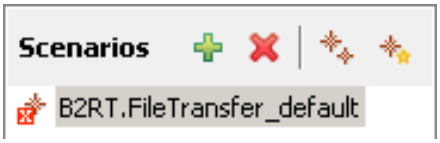
Creating a scenario

After modifying the configuration document for the File Drop service, you are going to create a scenario to test the FileTransfer process. The scenario sends the bulk purchase order file through the FileTransfer process:

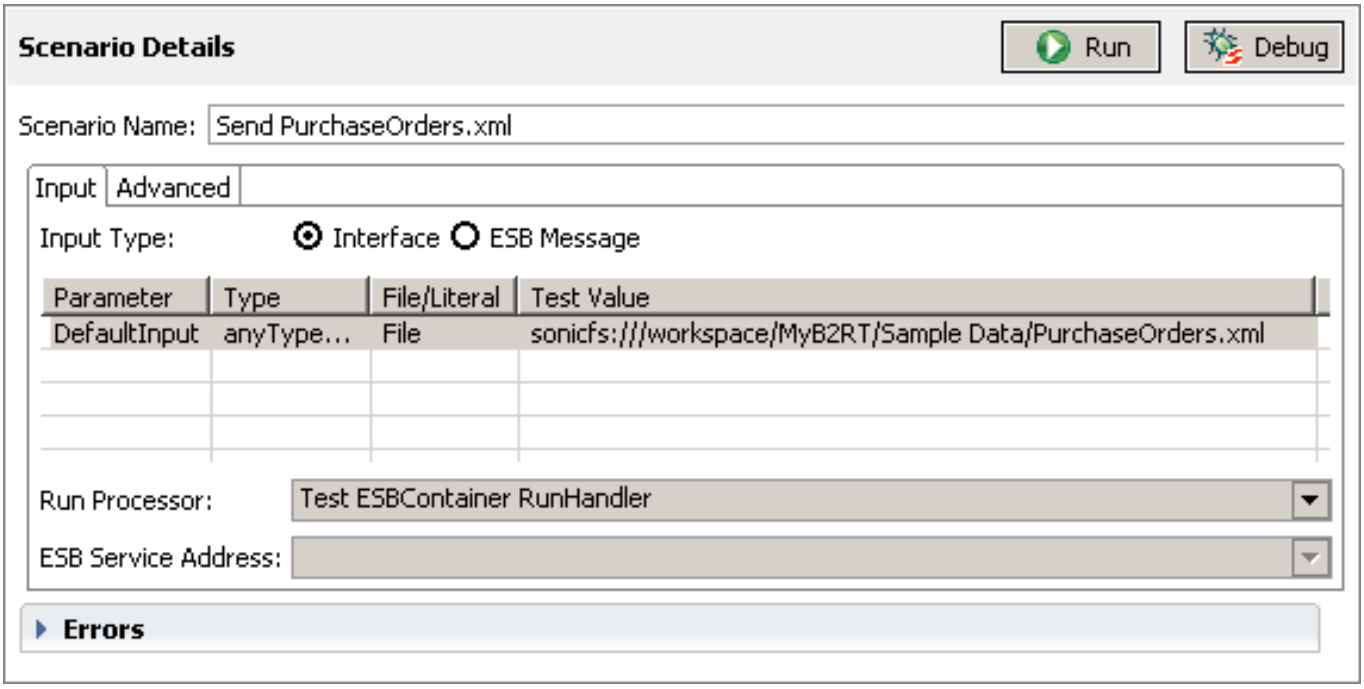
1. Go back to the Process page and click the **Scenarios** tab to open the Scenarios page:



2. In the **Scenarios** section, click **Add Scenario** to create a new scenario, automatically named MyB2RT.FileTransfer_default:



3. In the **Scenario Details** section, change the **Scenario Name** to *Send PurchaseOrders.xml*.
4. Drag `PurchaseOrders.xml` from the Sample Data folder in the Navigator view to the **Test Value** field:

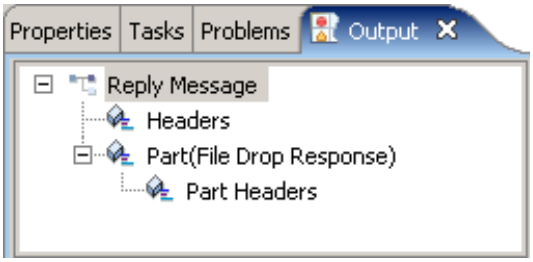


Next, use the scenario to test the FileTransfer process.

Testing the FileTransfer process

You can now run the scenario you created to test the FileTransfer process and confirm that the bulk purchase order file is dropped in the target directory:

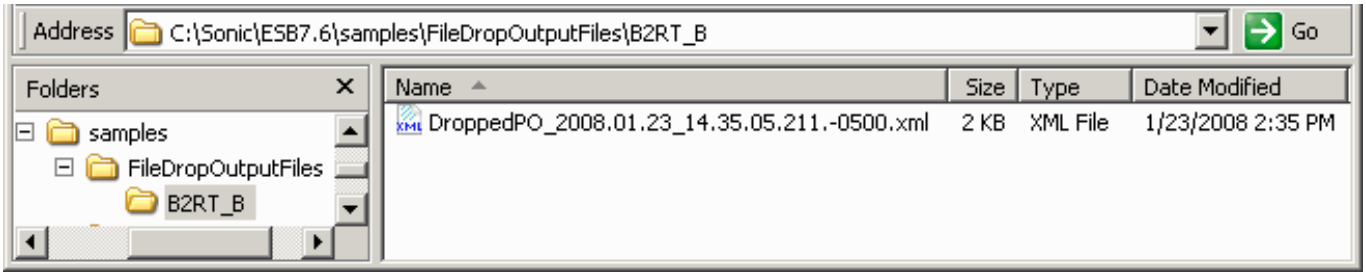
1. With the scenario open in the Scenarios page, click **Run** to run the process using the scenario you created.
2. Go to the Output view to view the single message (and its part and headers) in the left pane:



3. You can view the reply message, headers, part, and part headers in the right pane. Observe that the reply message contains details about the location where the file is dropped:

```
<?xml version="1.0"?>
<fileServicesEvent>
    <eventType>FileDrop</eventType>
    <timestamp>2008.01.23_14.35.05.227.-0500</timestamp>
    <fileName>DroppedPO_2008.01.23_14.35.05.211.-0500.xml</fileName>
    <dropLocation>C:\Sonic\ESB7.6\samples\FileDropOutputFiles\B2RT_B\</dropLocation>
</fileServicesEvent>
```

4. Go to the \B2RT_B folder to confirm that one file was dropped:



5. You can open the file and view its contents. Observe that the file contains the three purchase orders in `PurchaseOrders.xml`.

You have completed phase 1. Next, proceed with phase 2: Implementing validation and XML splitter operations.

Phase 2: Implementing validation and XML splitter operations

After completing phase 1, you are going to implement a validation operation and an XML splitter operation:

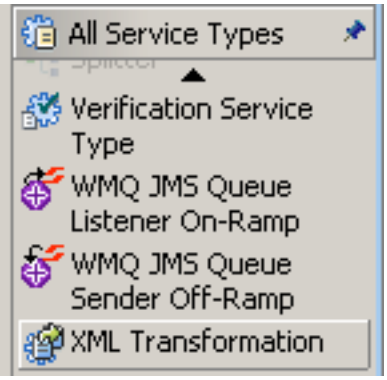
1. Implement the Validate step — Replace the Prototype service in the Validate step with an XML transformation service.
2. Specify validation — Complete the implementation of the validation operation.
3. Add an XML splitter step — Replace the Prototype service in the Transform step with an XML splitter.
4. View the properties of the XML splitter step — View the properties of the XML Transformation service, including the stylesheet.
5. Find an XPath node for the stylesheet — Use the XPath Helper to find an XPath expression to match every purchase order.
6. Modify the stylesheet — Replace a node in the stylesheet with the XPath expression you found.
7. Create a scenario to test the stylesheet — Use the sample purchase order file to create a scenario.
8. Test the stylesheet — Use the scenario to test the stylesheet.
9. Test the enhanced FileTransfer process — Confirm that three individual purchase order files are dropped in the target directory.

Start by implementing the Validate step.

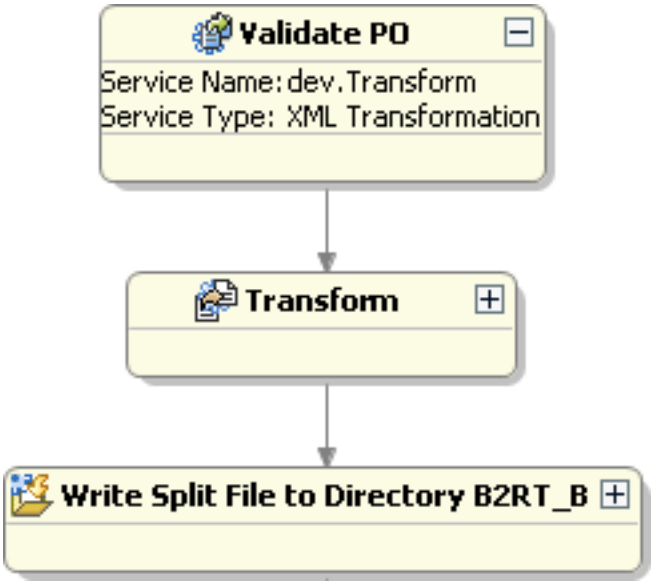
Implementing the Validate step

Validate is the first step in the VETO template you added to the FileTransfer process. Like all the steps in the VETO template, **Validate** is a Prototype service. You will replace this service with a an XML transformation service to implement the validation operation:

1. Go to the Process page and select the **XML Transformation** service type from the **All Service Types** section of the Palette:



2. Drag the **XML Transformation** service onto the **Validate** step.
3. Change the name of the step to **Validate PO**.
4. Expand the **Validate PO** step and observe that it is now an XML Transformation service:



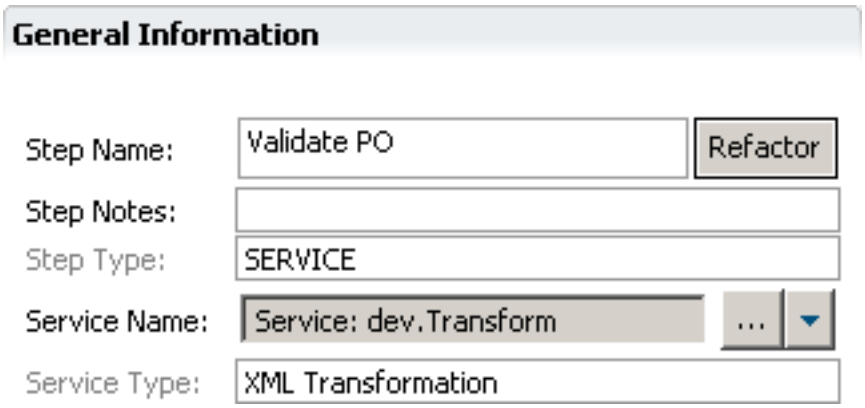
5. Save the FileTransfer process.

Next, specify validation.

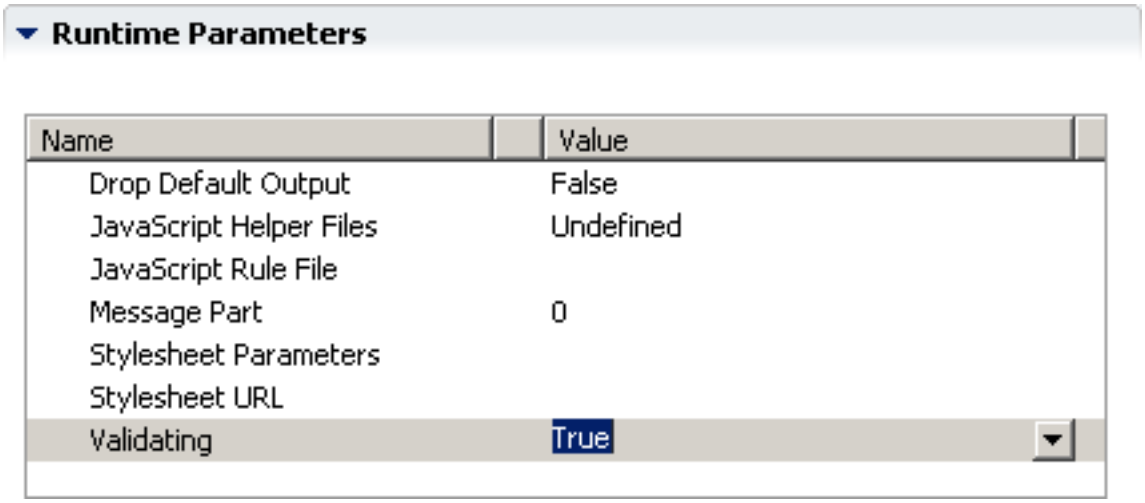
Specifying validation

After implementing the Validate step, you are going to specify XML schema validation. The Validate PO step will use an XML Transformation service to validate the incoming bulk purchase orders against an XML schema to be sure the purchase orders confirm to the correct XML document format:

1. Double-click the **Validate PO** step in the Process page to open the Service page and view the service properties.
2. In the **General Information** section, observe that this step is an XML Transformation service:



3. In the **Runtime Parameters** section, click the down arrow in the **Value** field for **Validating** to change the value to **True**:



4. Save the FileTransfer process.

Note: This is a simplified method of validation. In an enterprise application, you would implement more advanced validation.

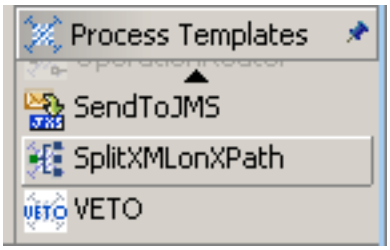
Next, add an XML splitter step.

Adding an XML splitter step

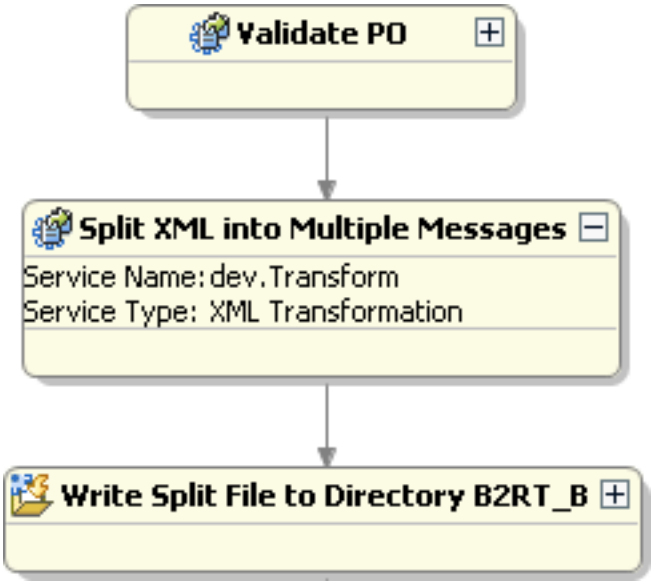
After specifying validation, you will use another enterprise integration pattern template to add an XML splitter step to replace the **Transform** step. The XML splitter step will split the bulk purchase order file into individual purchase order files based on an XPath expression. In an enterprise application, an XML splitter might split a file containing hundreds or even thousands of items.

To add the XML splitter step:

1. In the Process page, select the **SplitXMLonXPath** template from the **Process Templates** section of the Palette:



2. Drag the **SplitXMLonXPath** template below the **Transform** step. The ESB Process editor adds the **Split XML into Multiple Messages** step.
3. Since you added the **Split XML into Multiple Messages** step, you should now delete the **Transform** step.
4. Expand the **Split XML into Multiple Messages** step and observe that it is an XML Transformation service:



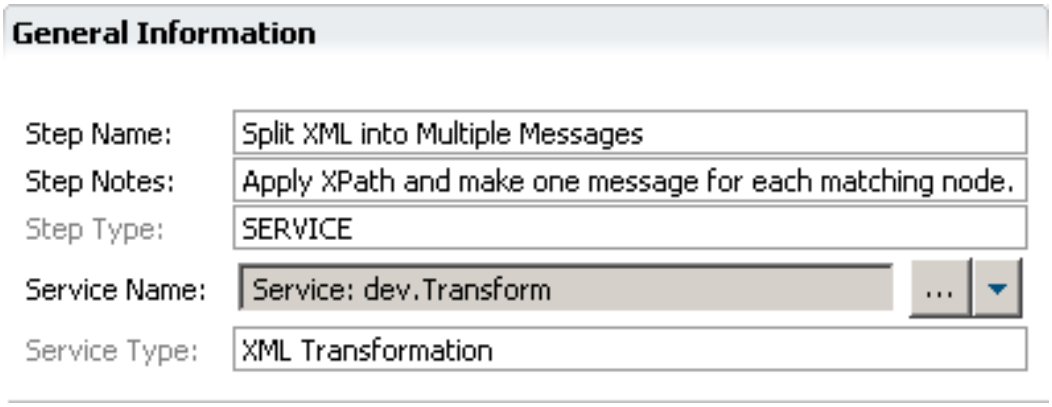
5. Save the FileTransfer process.

Next, view the properties of the XML splitter step.

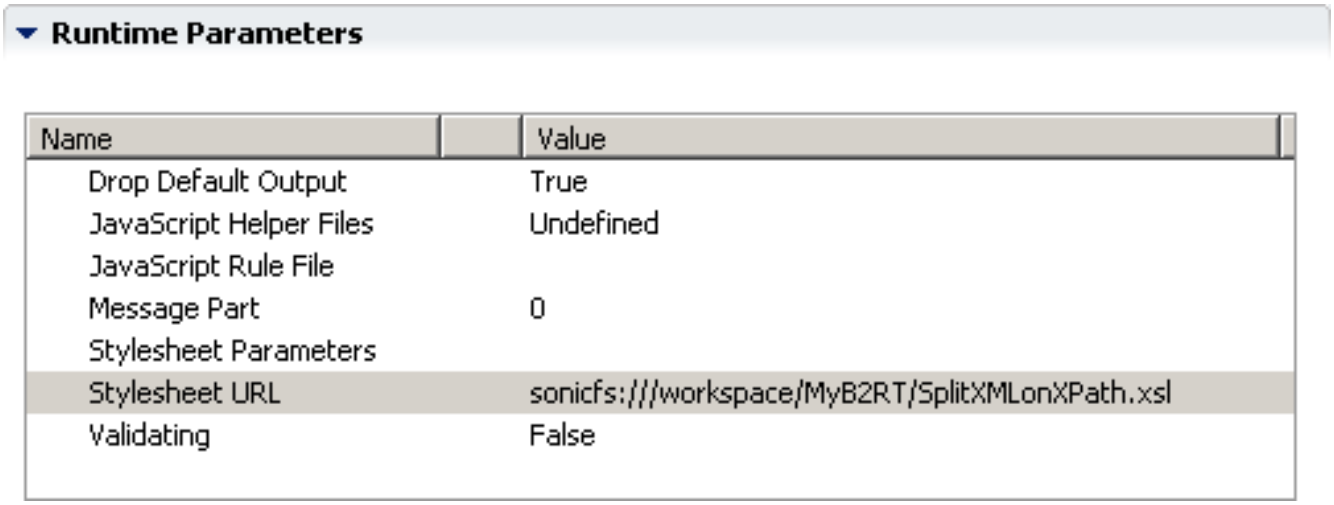
Viewing the properties of the XML splitter step

After adding the XML splitter step, you can view the properties of the **Split XML into Multiple Messages** step:

1. Double-click the **Split XML into Multiple Messages** step to open the Service page.
2. Observe that this step is an XML transformation service:



3. In the **Runtime Parameters** section, observe that the XML transformation service uses the `SplitXMLonXPath.xsl` stylesheet that Sonic Workbench added to your project when you used the SplitXMLonXPath template to add the XML splitter step:

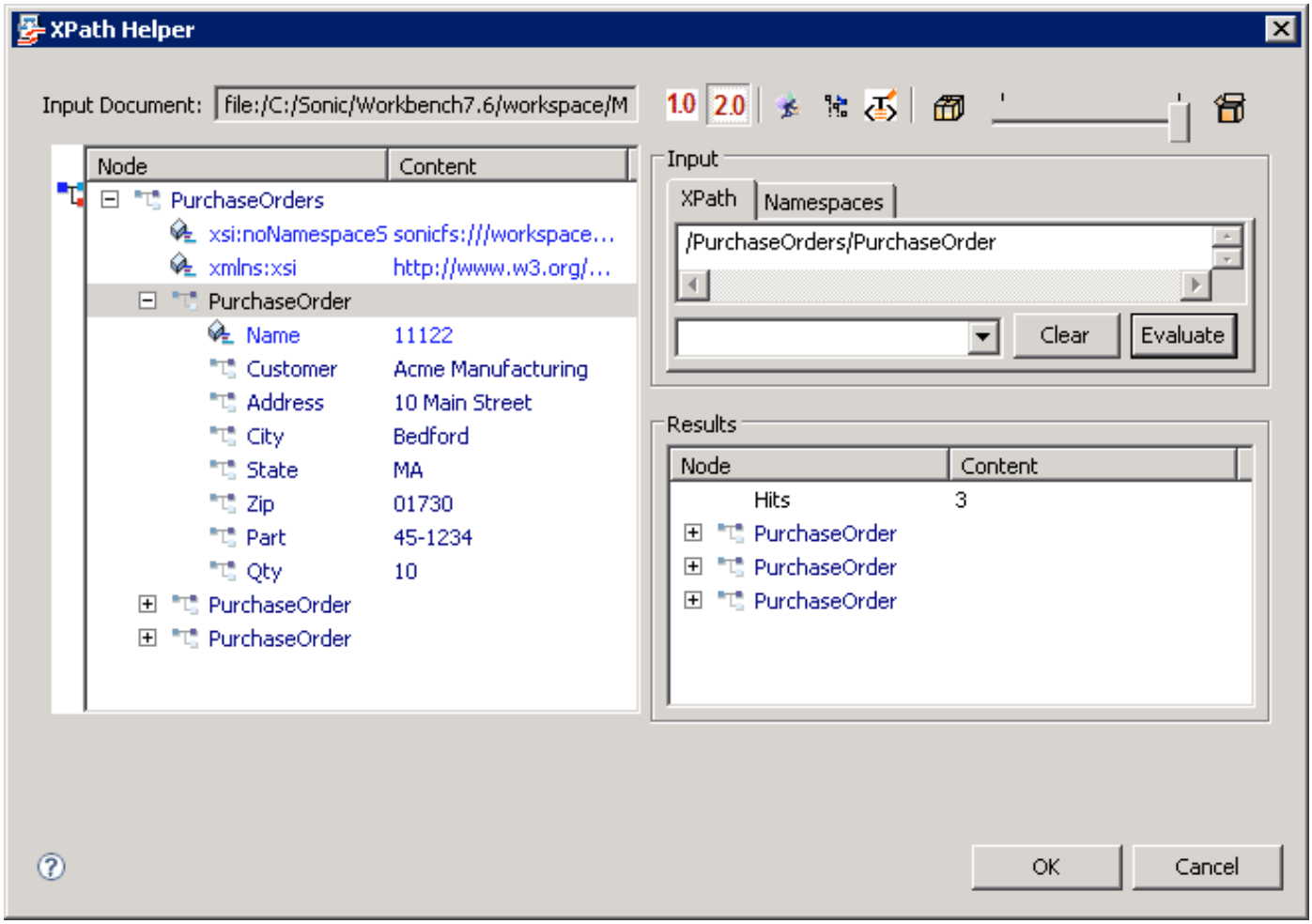


Next, find an XPath node for the stylesheet.

Finding an XPath node for the stylesheet

As you observed, the XML Transformation service uses a stylesheet. The stylesheet uses an XPath expression to split the bulk purchase order file into individual purchase order files. You will use the XPath Helper to find an XPath node to match each purchase order in the bulk purchase order file:

1. Double-click `PurchaseOrders.xml` in the Sample Data folder in the Navigator view. The file opens in the XML editor.
2. Click  (in the upper right corner) to open the XPath Helper.
3. Double-click one of the PurchaseOrder nodes in the left panel.
4. Observe that the Xpath node, /PurchaseOrders/PurchaseOrder, is listed on the **XPath** tab in the **Input** section of the right panel.
5. Click **Evaluate**. Observe that all three XML nodes are listed in the Results section:



6. Copy the text, `PurchaseOrders/PurchaseOrder`, from the **Input** section.
7. Click **OK** to close the XPath Helper.
8. Close `PurchaseOrders.xml` (and the XML editor).

Next, modify the stylesheet.

Modifying the stylesheet

After finding the XPath node, you are ready to specify this node in the generic XPath expression in the stylesheet:

1. Go to the Process page of the ESB editor.
2. Select the **Split XML into Multiple Messages** step and click **F3**. The `SplitXMLonXPath.xslt` stylesheet opens in the XSLT editor.
3. Specify the node in the following XPath expression by replacing `/NODE` with `PurchaseOrders/PurchaseOrder` that you copied from the XPath Helper:

   ```
   <xsl:for-each select="/PurchaseOrders/PurchaseOrder">
   ```

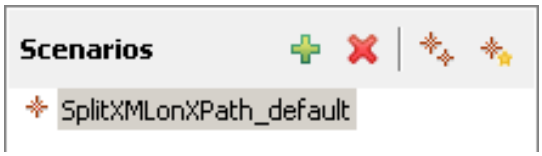
   Your stylesheet should now resemble the stylesheet in the Sample.B2RT project.
4. Save the stylesheet.

Next, create a scenario to test the stylesheet.

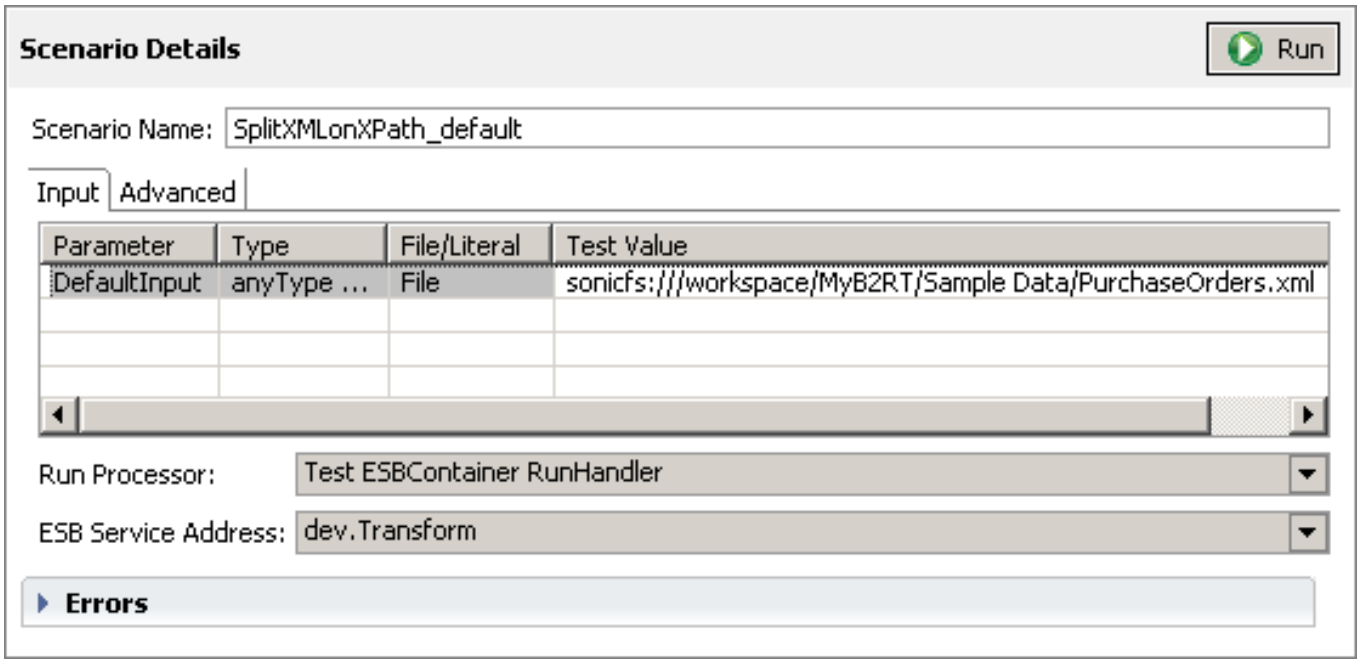
Creating a scenario to test the stylesheet

After modifying the stylesheet, you can create a scenario to test the stylesheet:

1. With the `SplitXMLonXPath.xsl` stylesheet open in the XSLT editor, click the **Scenarios** tab to open the Scenarios page.
2. In the **Scenarios** section, click **Add Scenario** to create a new scenario. By default, the new scenario is named SplitXMLonXPath_default:



3. In the **Scenario Details** section, enter a **Scenario Test Value** by dragging the sample XML file, `PurchaseOrders.xml`, from the Sample Data folder in the Navigator view. The **Scenario Details** section now looks like this:

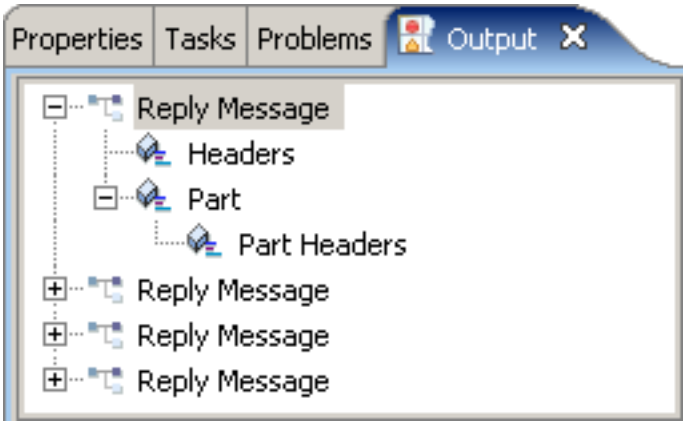


Next, you can run the scenario to test the stylesheet.

Testing the stylesheet

You can run the scenario you created to test the stylesheet:

1. In the Scenarios page of the XSLT editor, click **Run** to run the stylesheet using this scenario.
2. Go to the Output view and observe the four reply messages, one for each of the three purchase orders and one for the default message (the original message):



3. View each message in the right pane, observing that each one contains one purchase order from the bulk purchase order file:

```
<PurchaseOrder Name="11122"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Customer>Acme Manufacturing</Customer>
   <Address>10 Main Street</Address>
   <City>Bedford</City>
   <State>MA</State>
   <Zip>01730</Zip>
   <Part>45-1234</Part>
   <Qty>10</Qty>
</PurchaseOrder>
```

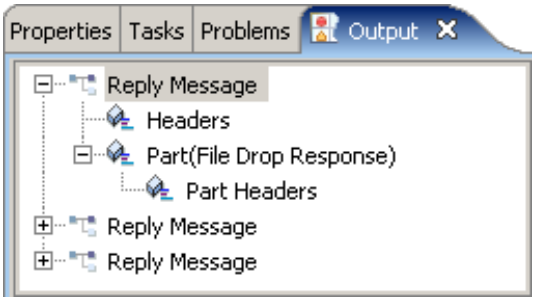
4. Now that you modified and tested the stylesheet, you can save and close it.

Next, test the enhanced FileTransfer process

Testing the enhanced FileTransfer process

Now that you tested the stylesheet, you can test the modified FileTransfer process:

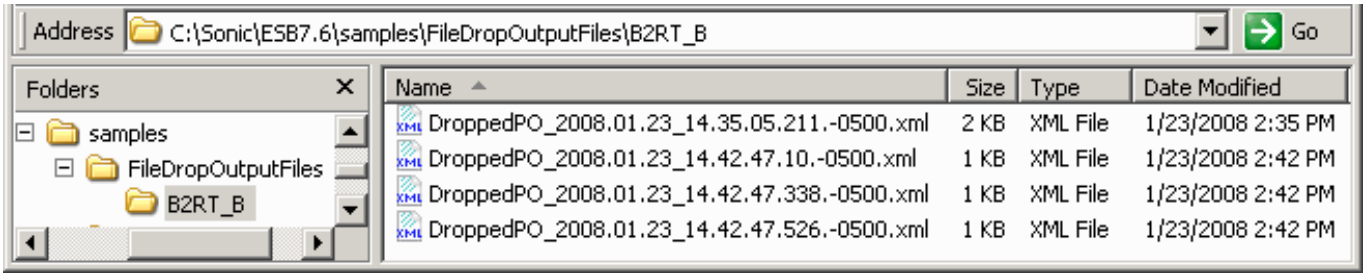
1. Return to the Process page of the FileTransfer process and save the FileTransfer process.
2. Click the **Scenarios** tab to open the Scenarios page.
3. Click **Run** to run the same scenario you created.
4. Go to the Output view to view the three messages (and parts and headers) in the left pane. There are three messages, one for each purchase order in the bulk purchase order file:



5. View each message in the right pane. Observe that the reply message contains details about the location where the file is dropped, for example:

```
<?xml version="1.0"?>
<fileServicesEvent>
    <eventType>FileDrop</eventType>
    <timestamp>2008.01.23_14.42.47.10.-0500</timestamp>
    <fileName>DroppedPO_2008.01.23_14.42.47.10.-0500.xml</fileName>
    <dropLocation>C:\Sonic\ESB7.6\samples\FileDropOutputFiles\B2RT_B\</dropLocation>
</fileServicesEvent>
```

6. Go to the \B2RT_B folder to confirm that three new files were dropped:



7. You can open the new files and observe that each new file contains one of the three purchase orders in `PurchaseOrders.xml`.

You have completed phase 2. Next, proceed with phase 3: Implementing the second file drop operation.

Phase 3: Implementing the second file drop operation

After completing phase 2, you will develop the final phase of the FileTransfer process by adding a branch and implementing a second File Drop operation:

1. Add a fanout — Create two branches in the FileTransfer process.
2. Modify the fanout — Move two existing steps onto the right branch of the fanout.
3. Add the second file drop step — Add a file drop step to the left branch of the fanout.
4. Modify the second configuration document — Specify the name of the purchase order file and the target drop directory in the configuration document for the second file drop step.
5. Test the completed FileTransfer process — Confirm that the purchase order files are dropped correctly in the target directories.

At the end of this phase, the FileTransfer process:

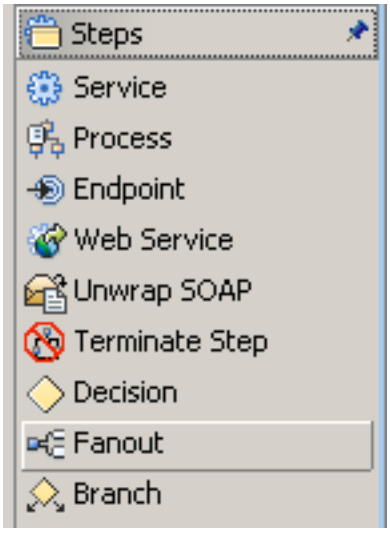
- Writes the original purchase order file to directory B2RT_A
- Writes the split purchase order files to directory B2RT_B

Start by adding a fanout.

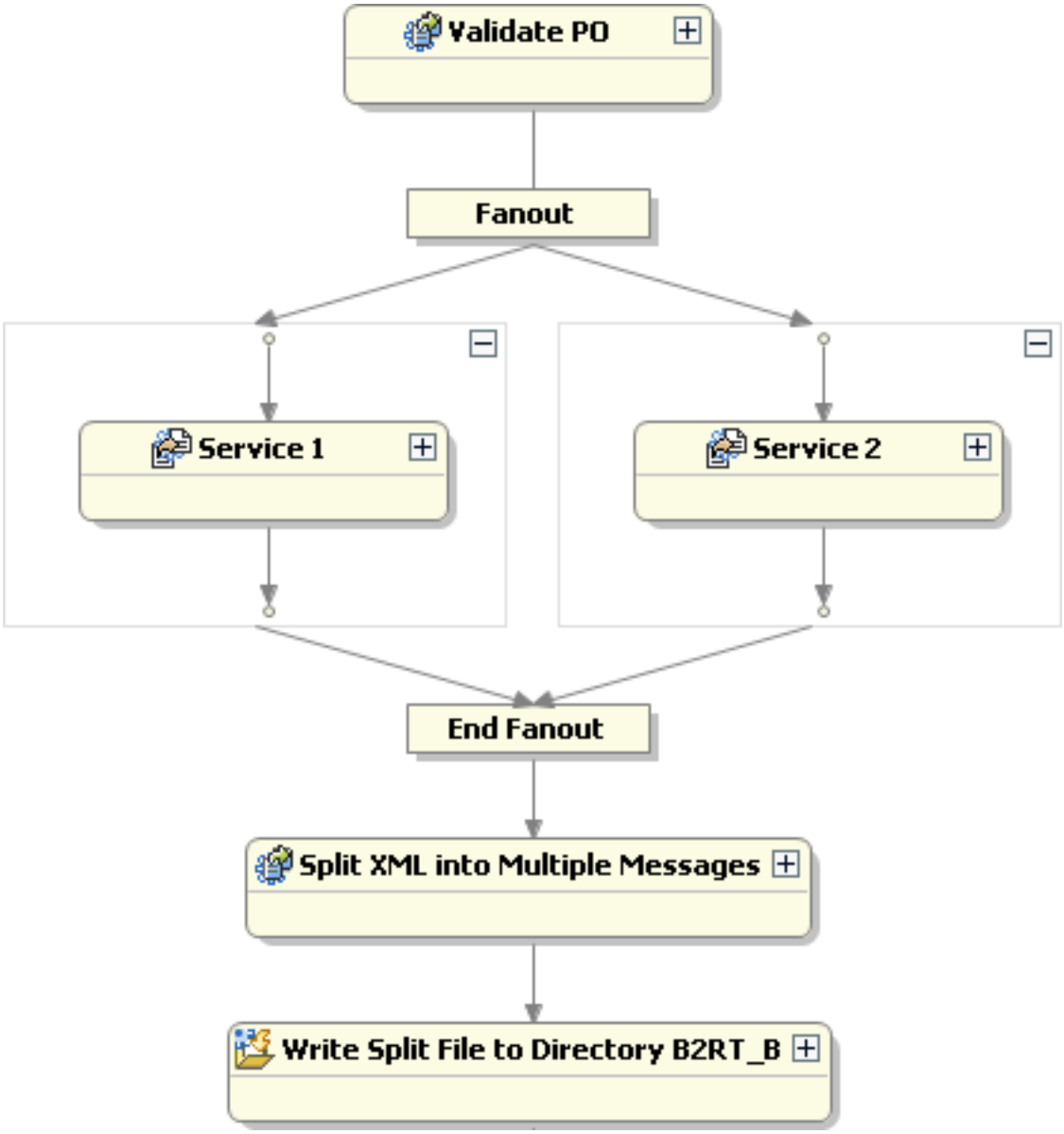
Adding a fanout

After testing the enhanced FileTransfer process, you add a fanout:

1. Go back to the Process page and select the **Fanout** step from the **Steps** section of the Palette:



2. Drag the **Fanout** step onto the FileTransfer process just below the **Validate PO** step:



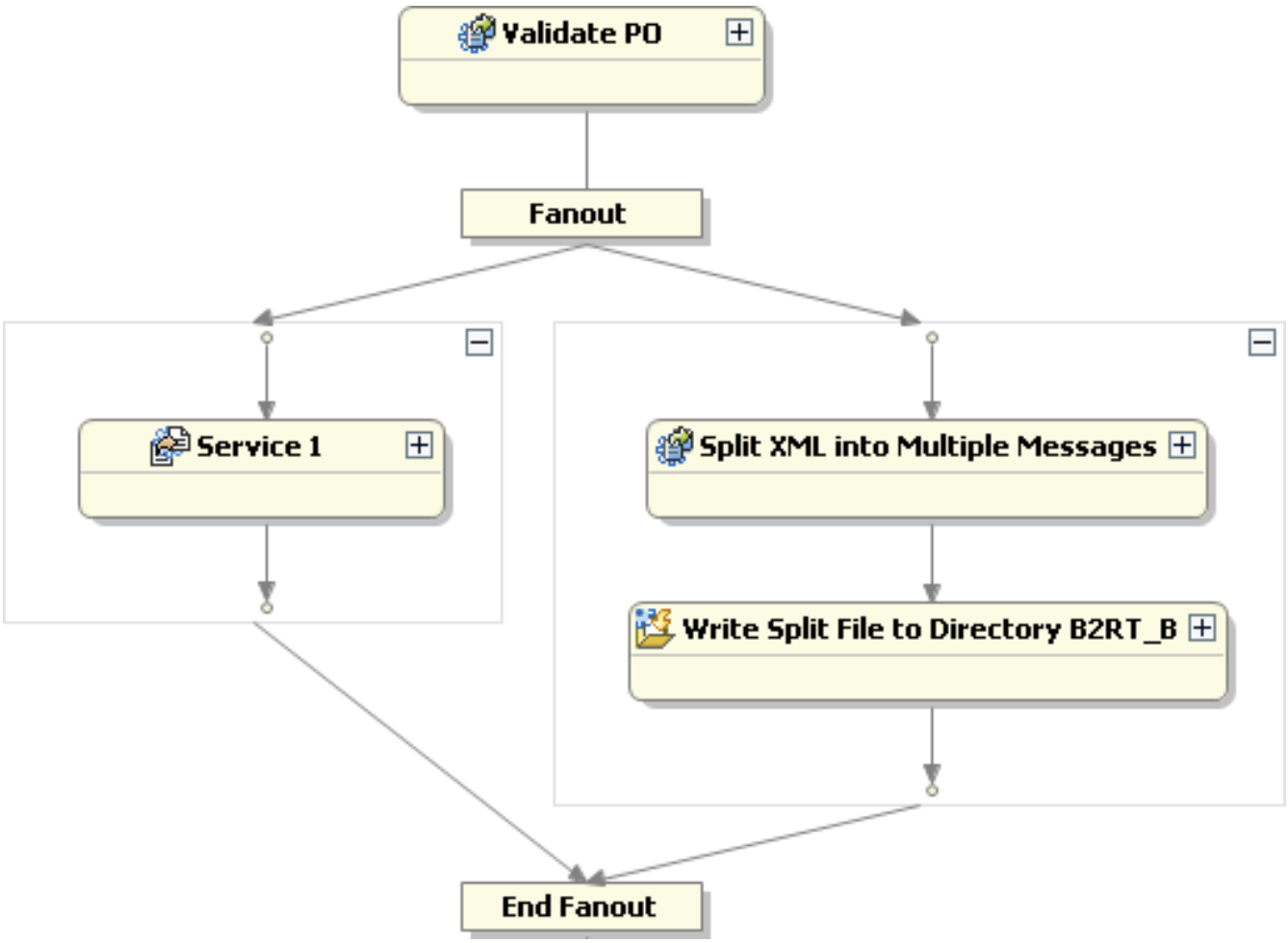
3. Save the FileTransfer process.

Next, modify the fanout.

Modifying the fanout

After adding the fanout, you move some of the existing steps onto the fanout:

1. Move the **Split XML into Multiple Messages** step up onto the right branch of the fanout.
2. Move the **Write Split File to Directory B2RT_B** step up into the fanout below the **Split XML into Multiple Messages** step.
3. Delete the unnecessary **Service 2** step:



4. Save the FileTransfer process.

Next, add the second File Drop step.

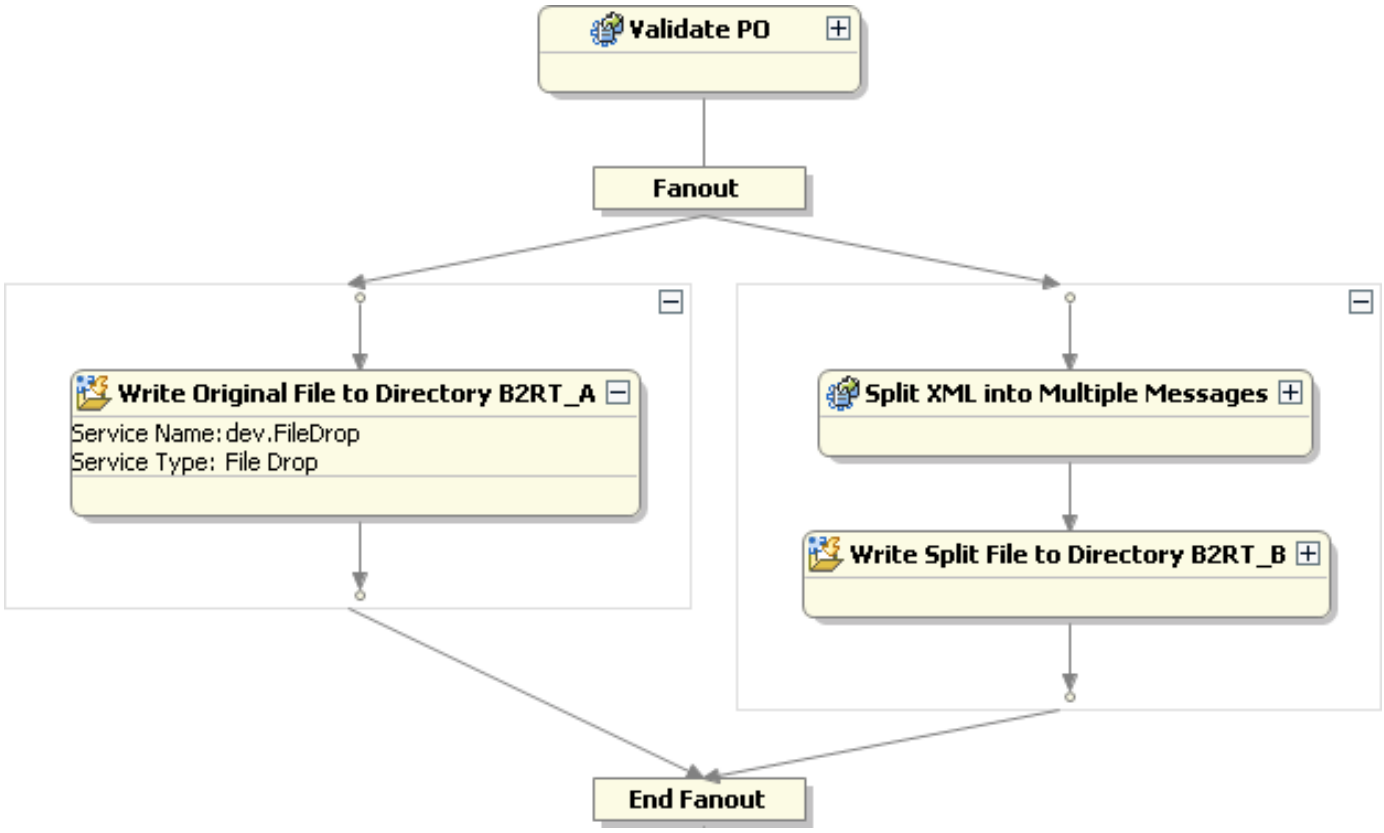
Adding the second file drop step

After modifying the fanout, **Service 1** is still a Prototype service. You will replace it with a second file drop step to drop the bulk purchase order in a different directory.

To add the file drop step:

1. Select the **DropFileInDirectory** template from the **Process Templates** section of the Palette:

2. Drag the **DropFileInDirectory** template onto your process below the **Service 1** step. The ESB Process editor adds the **Drop into Directory** step.
3. Change the name of the step from **Drop into Directory** to **Write Original File to Directory B2RT_A**.
4. Delete the **Service 1** step.
5. Expand the **Write Original File to Directory B2RT_A** step and observe that it is a File Drop service:



6. Save the FileTransfer process.

Next, modify the configuration document.

Modifying the second configuration document

After adding the second file drop step, you have to modify the configuration document that the **Write Original File to Directory B2RT_A** step uses for its runtime parameters:

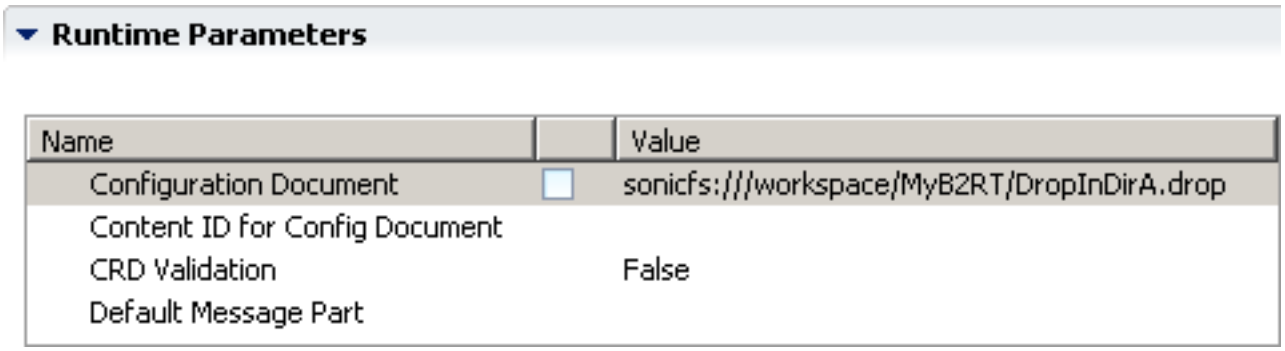
1. In the Process page, select the **Write Original File to Directory B2RT_A** step and click **F3**. The `DropFileinDirectory.drop` file opens in the XML editor.
2. Specify the name of the Purchase order file by changing `FILENAME.xml` to `OriginalPOList.xml`:

   ```
   <filename type="GENERATED">OriginalPOList.xml</filename>
   ```

3. Specify the target directory for dropping the purchase order files. Change `FILE_DROP_DIRECTORY` to `FileDropOutputFiles\B2RT_A`:

   ```
   <dropDirectory>C:\Sonic\ESB7.6\samples\FileDropOutputFiles\B2RT_A</dropDirectory>
   ```

4. Save and close the `DropFileinDirectory.drop` file.
5. Go to the Navigator view and select the `DropFileinDirectory.drop` file. Right-click, select **Rename**, and change the name of the file to `DropInDirA.drop`.
6. Go back to the Process page and double-click on the **Write Original File to Directory B2RT_A** step to open the Service page.
7. In the **Runtime parameters** section, change the name of the **Configuration Document** in the **Value** field to `DropInDirA.drop`:

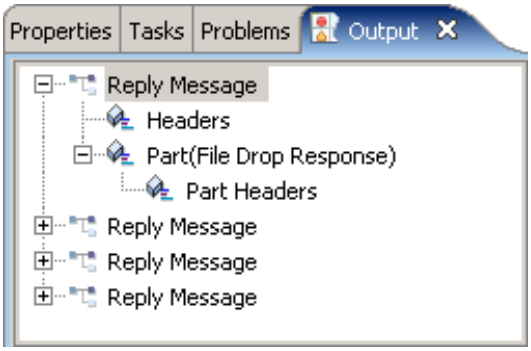
8. Save the completed FileTransfer process.

Next, test the completed FileTransfer process

Testing the completed FileTransfer process

After you modify the second configuration document, you can run the completed FileTransfer process:

1. Return to the Process page of the FileTransfer process.
2. Click the **Scenarios** tab to open the Scenarios page.
3. Click **Run** to run the same scenario you created.
4. Go to the Output view to view the reply messages in the left pane. There are four messages, one for the bulk purchase order and three for the split purchase orders:



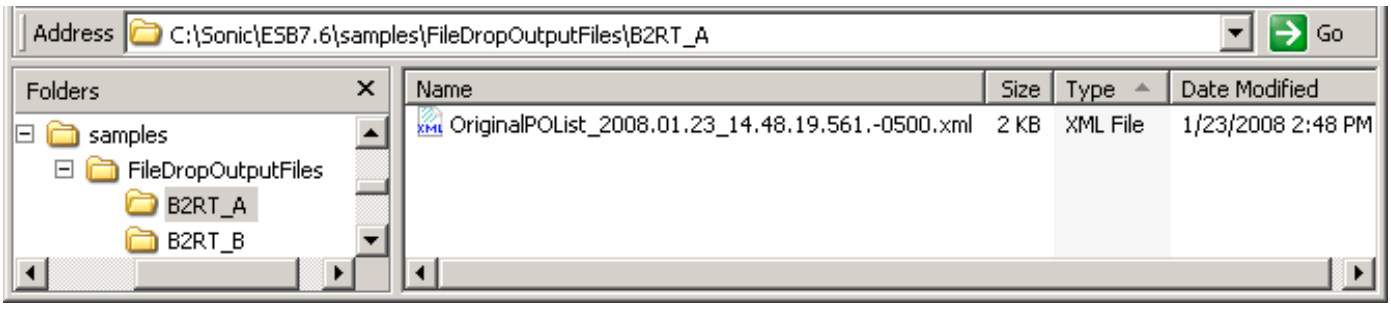
5. View each message in the right pane, for example the first message indicates that the bulk purchase order was dropped in directory B2RT_A:

```
<?xml version="1.0"?>
<fileServicesEvent>
    <eventType>FileDrop</eventType>
    <timestamp>2008.01.23_14.48.19.577.-0500</timestamp>
    <fileName>OriginalPOList_2008.01.23_14.48.19.561.-0500.xml</fileName>
    <dropLocation>C:\Sonic\ESB7.6\samples\FileDropOutputfiles\B2RT_A\</dropLocation>
</fileServicesEvent>
```

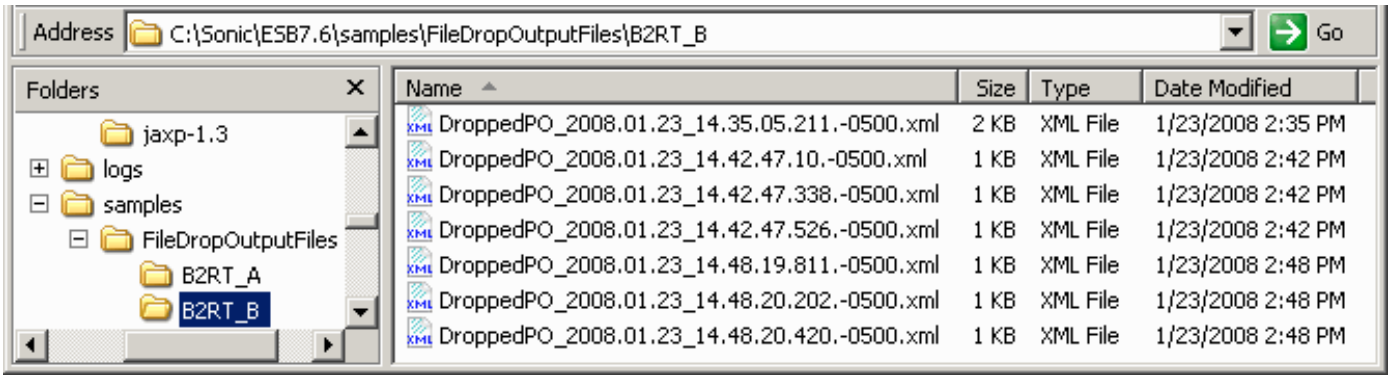
6. Observe that the other messages indicate that the split purchase orders were dropped in directory B2RT_B, for example:

```
<?xml version="1.0"?>
<fileServicesEvent>
    <eventType>FileDrop</eventType>
    <timestamp>2008.01.23_14.48.19.827.-0500</timestamp>
    <fileName>DroppedPO_2008.01.23_14.48.19.811.-0500.xml</fileName>
    <dropLocation>C:\Sonic\ESB7.6\samples\FileDropOutputFiles\B2RT_B\</dropLocation>
</fileServicesEvent>
```

7. Go to the \B2RT_A folder to confirm that one file was dropped in directory B2RT_A:



8. Confirm that three new files were dropped in directory B2RT_B:



9. You can open the files and look at the purchase orders.

You have successfully completed phase 3. You can optionally experience the distributed debugging features in sonic workbench while you track and debug the FileTransfer process.

Tracking and debugging the Batch to Real-time sample application

Sonic Workbench includes integrated distributed testing and tracking tools to increase SOA development productivity. You can immediately test ESB processes and services locally before deploying them across the network.

You already tested the completed FileTransfer process. You can also track messages, set breakpoints, and debug the FileTransfer process during the development process:

1. Configure ESB process tracking — Configure the FileTransfer process to specify the level of process tracking.
2. Add an ESB process tracker — Configure the Tracking Message Viewer so you can view message and process flows during execution
3. Set breakpoints — Set breakpoints in the FileTransfer process so you can visually step through and debug the process as it runs.
4. Start the debugger — Observe the debugging information in the Breakpoints, Debug, ESB Variables, and ESB Process Tracking views at the first breakpoint.
5. Run to the next breakpoint — Observe the debugging information in the ESB Variables and ESB Process Tracking views at the second breakpoint.
6. Run to the last breakpoint — Observe the debugging information in the ESB Variables and ESB Process Tracking views at the third breakpoint.
7. View the results of tracking and debugging — Observe the information in the Output and ESB Process Tracking views at the conclusion of the debugging session.

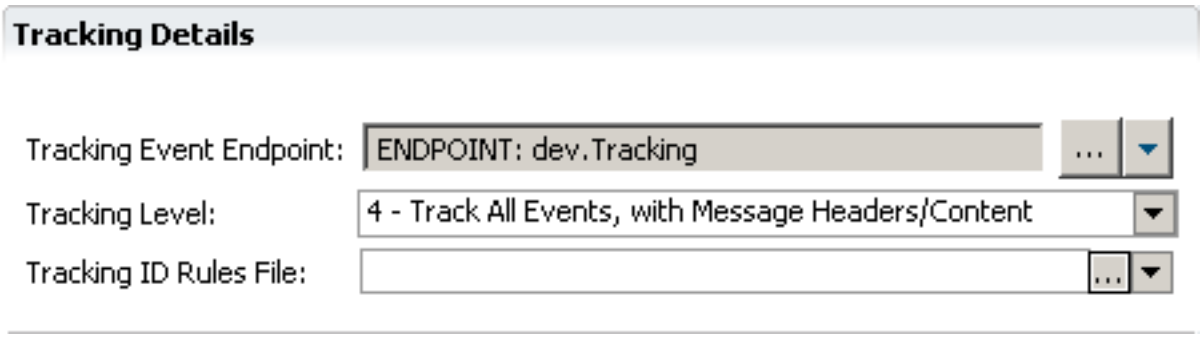
Start by configuring ESB process tracking.

Configuring ESB process tracking

You already tested the completed FileTransfer process. You can also track the messages through the FileTransfer process.

Currently the FileTransfer process is not configured for tracking. You are now going to change the ESB process configuration to specify process tracking and the level of tracking:

1. Click the **Overview** tab to go to the **Overview** page in the ESB Process editor.
2. In the **Tracking Details** section, change the **Tracking Level** to **Level 4 - Track All Events, with Message Headers/Content**:



3. Save the FileTransfer Process.

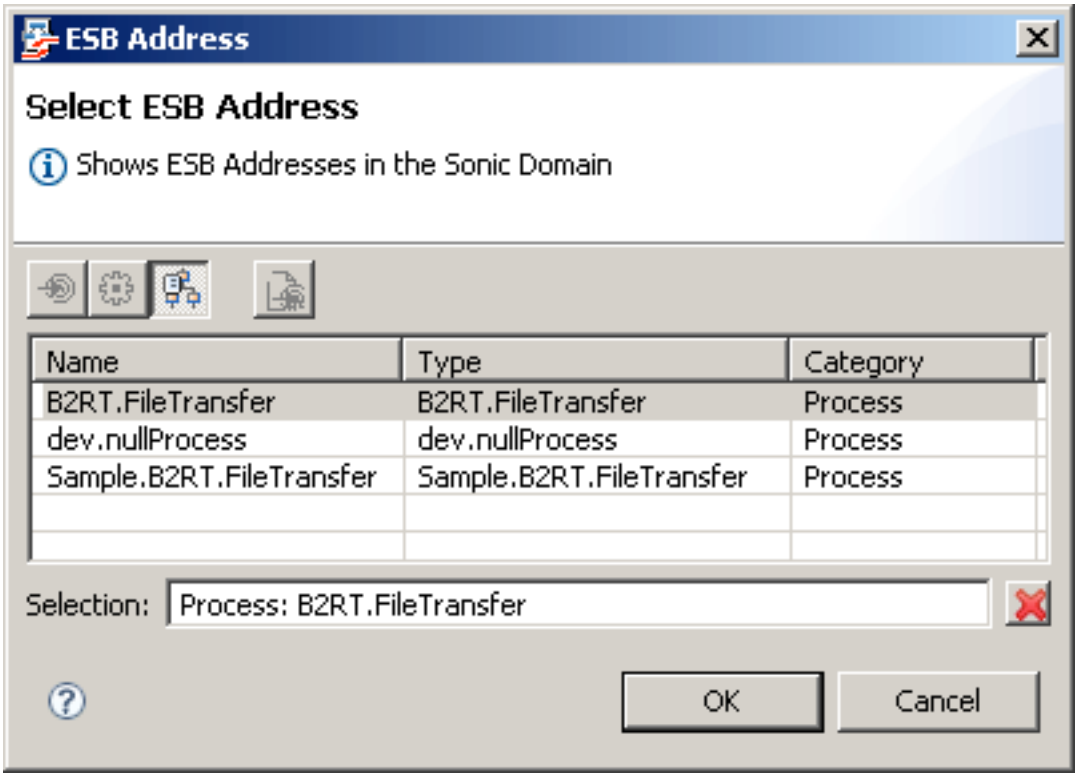
Next, add an ESB Process tracker.

Adding an ESB Process tracker

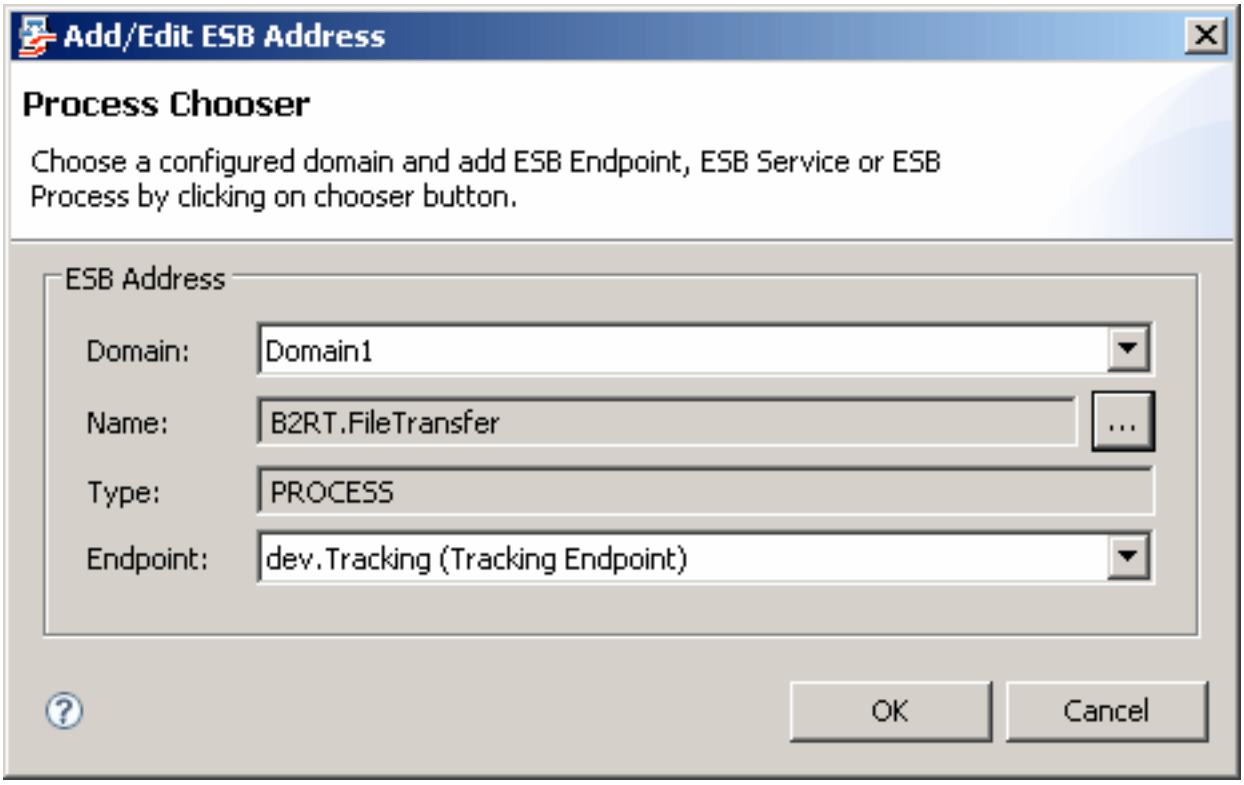
You can use the ESB Process Tracking view to trace step-by-step execution of a running process, making it easier to understand message and process flows even as they fan out into independent flows of execution. You will use the ESB Process Tracking view to view tracking messages when you debug the scenario.

After configuring ESB process tracking in the FileTransfer process configuration, you set up a listener in the Tracking Message Viewer:

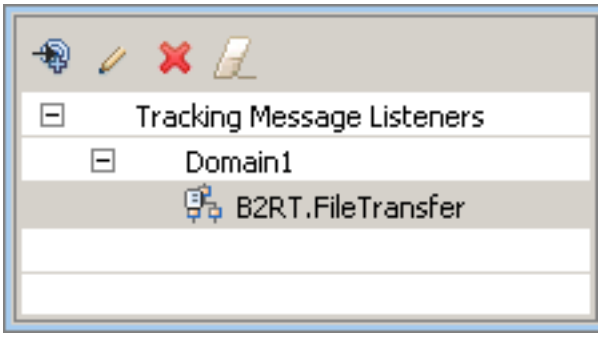
1. Go to the ESB Process Tracking view and click **Add**  . The **Process Chooser** opens.
2. Click **...** in the **Name** field to open the **ESB Address** chooser and select **B2RT.FileTransfer**:



3. Click **OK** to return to the **Process Chooser**.



4. Click **OK**.
5. Observe that **Sample.B2RT.FileTransfer** is listed as a **Tracking Message Listener**:

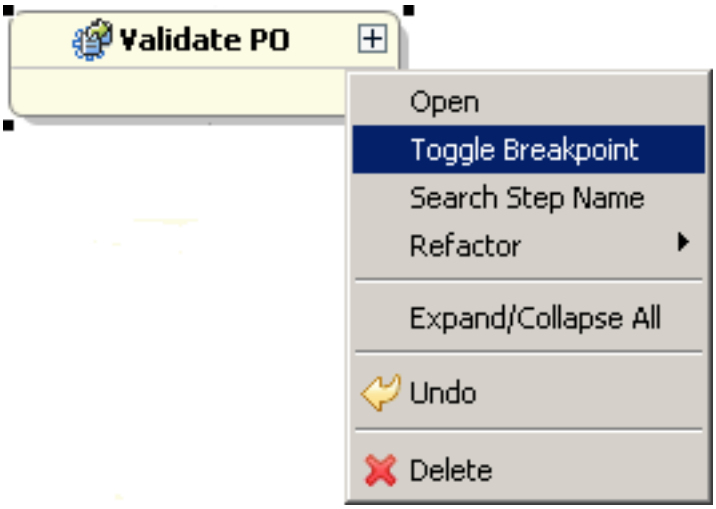


Next, set breakpoints.

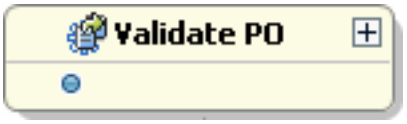
Setting breakpoints

You added an ESB Process tracker. You can also set breakpoints to debug the scenario. By setting breakpoints, you can visually step through and debug processes as they run:

1. Return to the Process page of the ESB Process editor. Select the **Validate PO** step, right-click, and select **Toggle Breakpoint**:



2. Observe the small circle on the step denoting that there is a breakpoint on that step:



3. Set breakpoints on the **Split XML into Multiple Messages** and the **Write Split File to Directory B2RT_B** steps also.

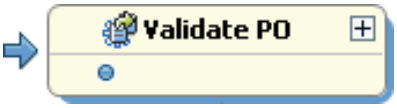
Next, start the debugger.

Starting the debugger

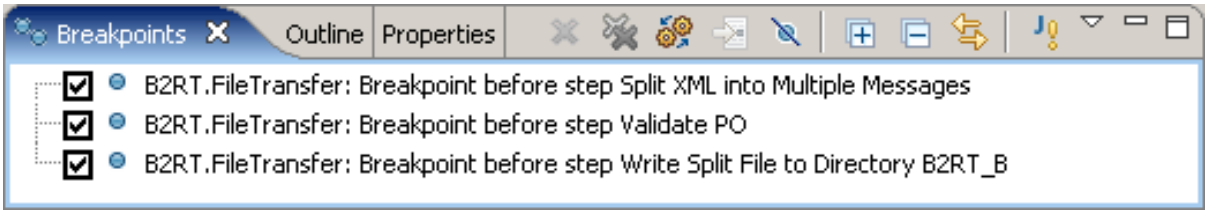
After setting breakpoints, you can start debugging the FileTransfer process:

1. Select the **Scenarios** tab to open the Scenarios page.
2. Click **Debug** to start debugging the ESB process using the same scenario you created.
3. Observe that the perspective changes to the Sonic Debug perspective:

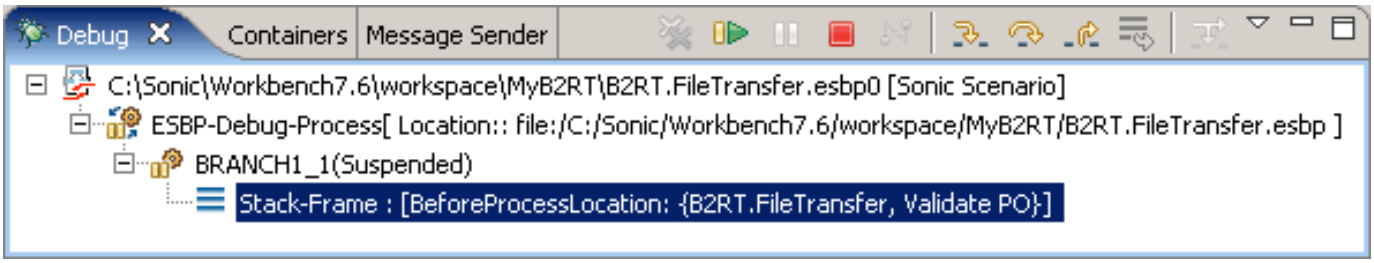
4. Observe that there is now an arrow to the left of the first step with a breakpoint, showing the current step in debugging:



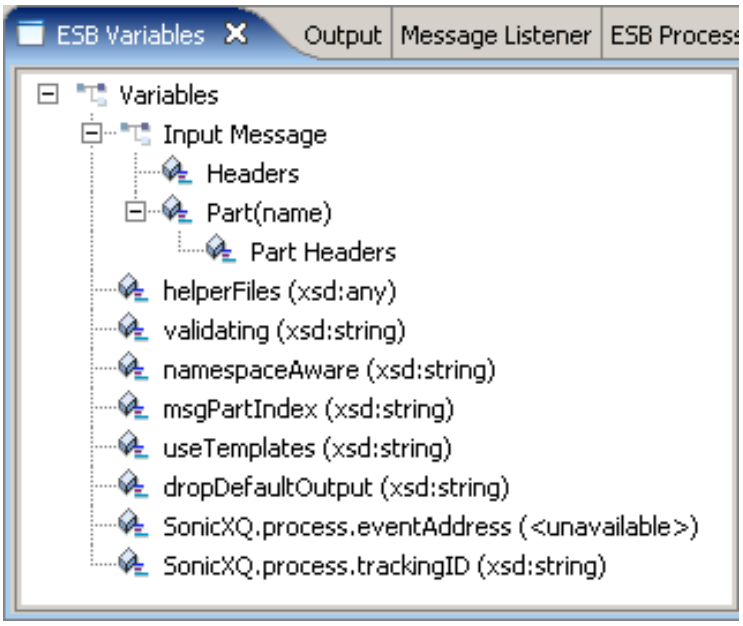
5. Go to the Breakpoints view to see the breakpoints you set:



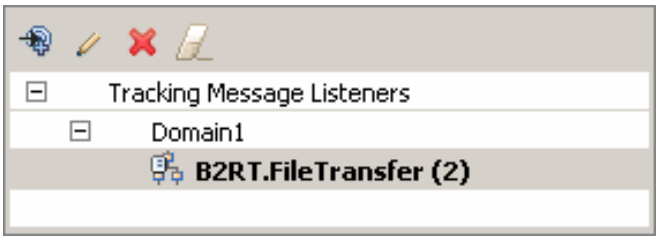
6. Go to the Debug view to view the stack frame location at the first breakpoint:



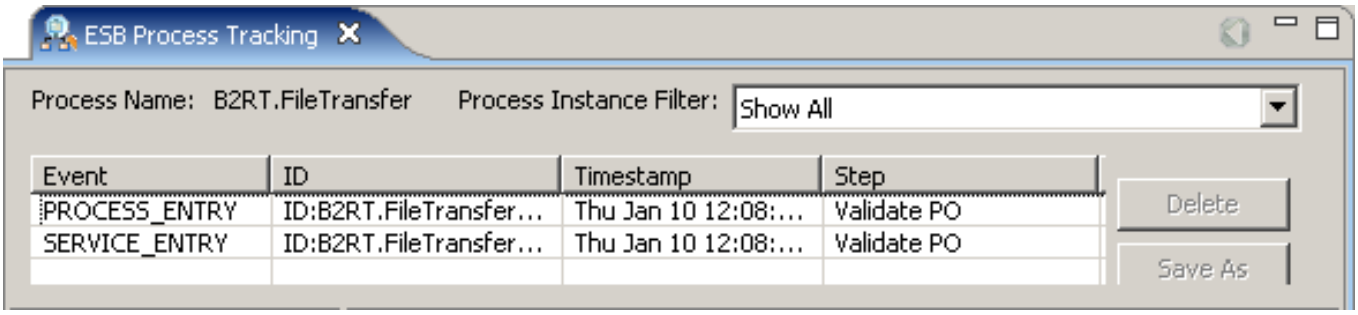
7. Select the stack frame and go to the ESB Variables view to view the message and variables:



8. Select the Input Message in the left pane and observe that the right pane shows the same data as in `PurchaseOrders.xml`.
9. Go to the ESB Process Tracking view and observe that there are two tracking messages:



10. You can view details about the messages in the right pane:



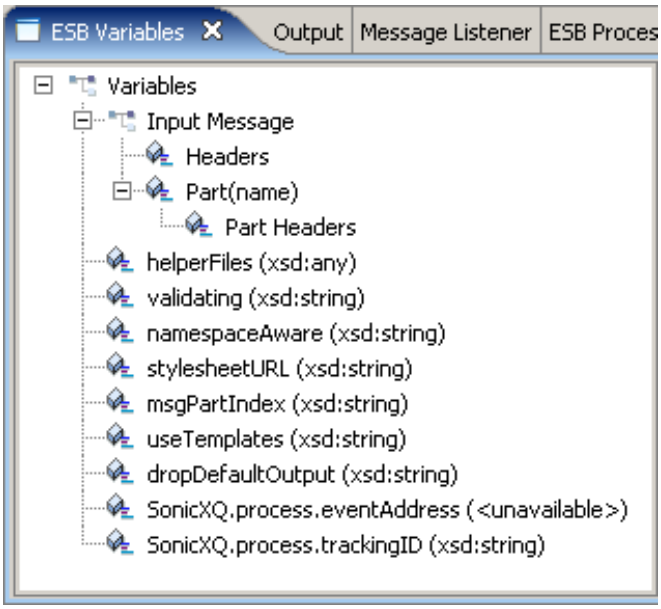
Next, run to the next breakpoint.

Running to the next breakpoint

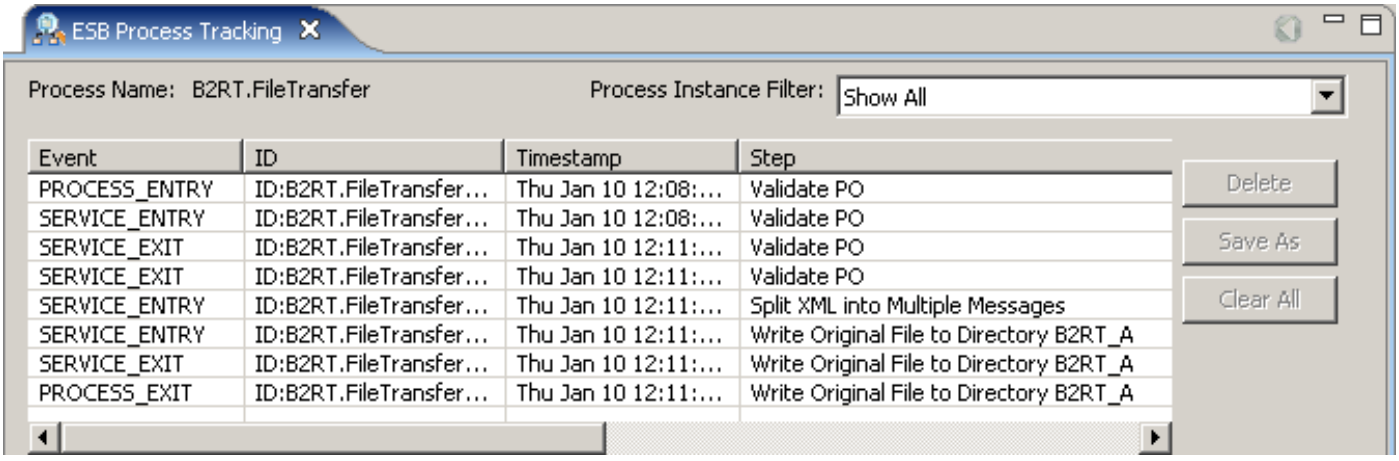
After viewing the debugging information at the first breakpoint, you can run to the next breakpoint in the FileTransfer process:

1. In the Debug view, click **Resume** to run to the next breakpoint. Observe that the arrow is now on the **Split XML into Multiple messages** step:

2. Select the stack frame and go to the ESB Variables view:



3. Go to the ESB Process Tracking view and observe that there are now eight tracking messages:

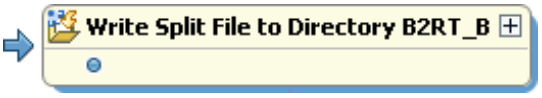


Next, run to the last breakpoint.

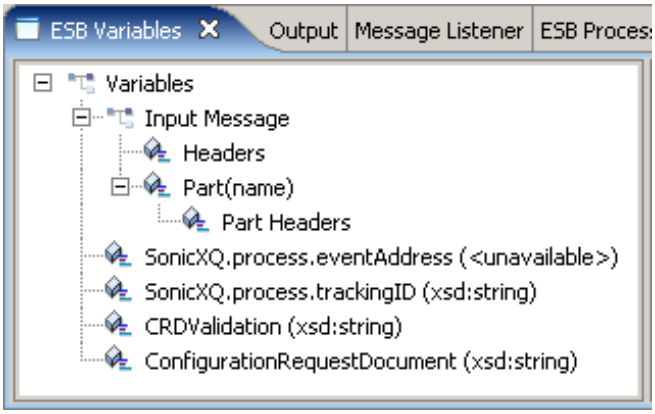
Running to the last breakpoint

After stepping to the second breakpoint, you can step to the last breakpoint:

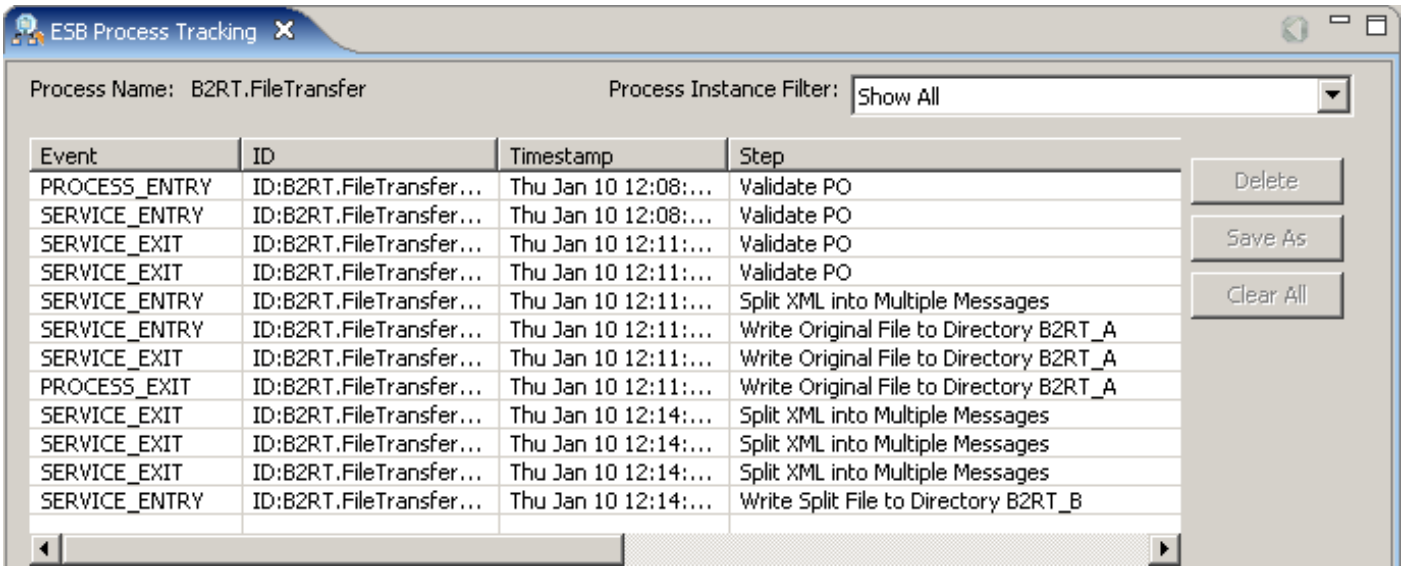
1. In the Debug view, click **Resume** to run to the next breakpoint. Observe that the arrow is now on the **Write Split File to Directory B2RT_B** step:



2. Select the stack frame and go to the ESB Variables view:



3. Go to the ESB Process Tracking view and observe that there are now 12 tracking messages:

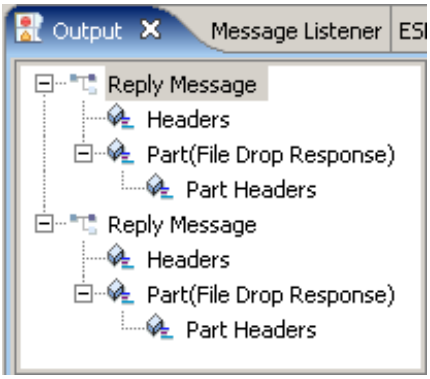


Next, view the results of tracking and debugging.

Viewing the results of tracking and debugging

After running to the last breakpoint, you can run to the end of the process and view the results of tracking and debugging:

1. In the Debug view, click **Resume** again. The debugger runs to the end of the process.
2. Observe that the Output view moved to the front and there are two reply messages in the left pane:



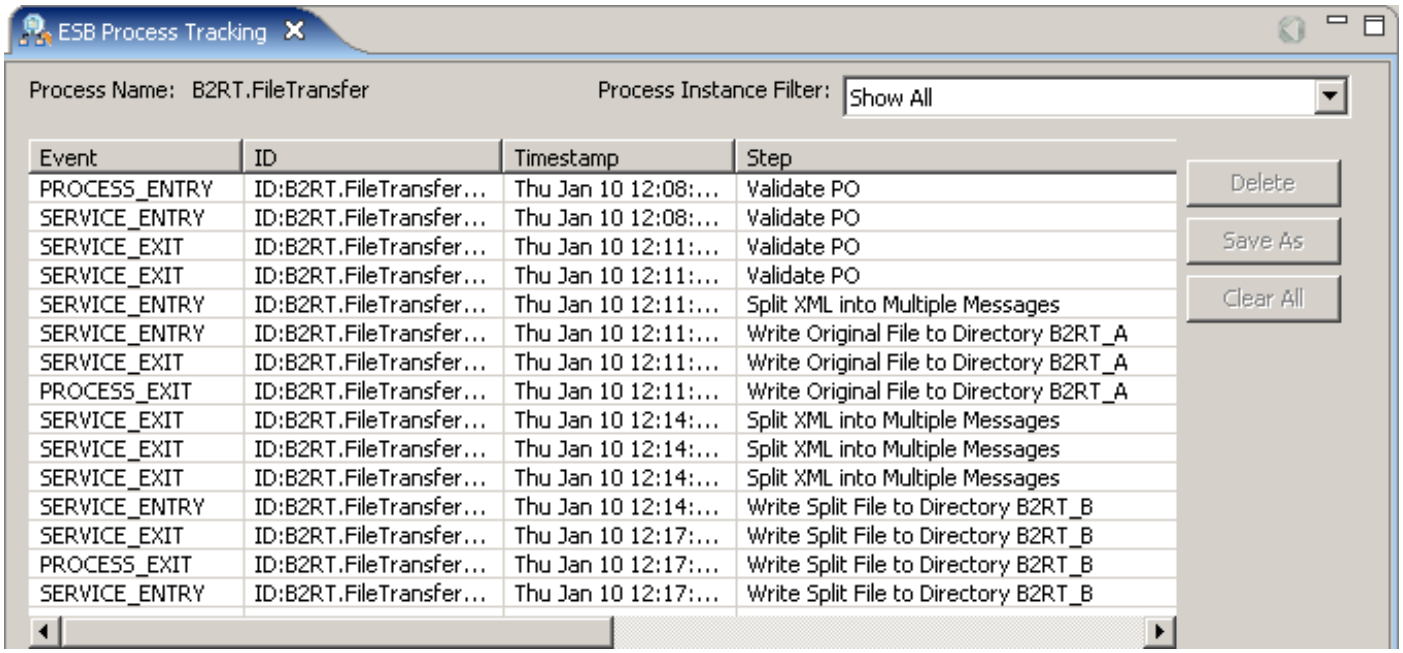
3. Observe the output for the first reply message in the right pane, indicating that the FileTransfer process dropped files named `OriginalPOList*.xml` in the `B2RT_A` directory:

```
<?xml version="1.0"?>
<fileServicesEvent>
   <eventType>FileDrop</eventType>
   <timestamp>2008.01.09_10.10.20.347.-0500</timestamp>
   <fileName>OriginalPOList_2008.01.09_10.10.20.331.-0500.xml</fileName>
   <dropLocation>C:\Sonic\ESB7.6\samples\FileDropOutputFiles\B2RT_A\</dropLocation>
</fileServicesEvent>
```

4. Observe the output for the second reply message in the right pane, indicating that the FileTransfer process dropped files named `DroppedPO*.xml` in the `B2RT_B` directory:

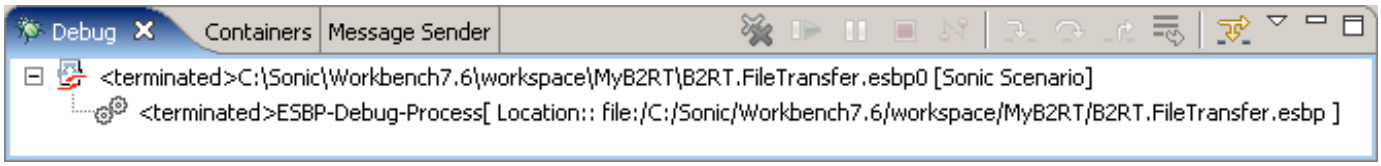
```
<?xml version="1.0"?>
<fileServicesEvent>
   <eventType>FileDrop</eventType>
   <timestamp>2008.01.09_10.12.57.196.-0500</timestamp>
   <fileName>DroppedPO_2008.01.09_10.12.57.196.-0500.xml</fileName>
   <dropLocation>C:\Sonic\ESB7.6\samples\FileDropOutputFiles\B2RT_B\</dropLocation>
</fileServicesEvent>
```

5. Go to the ESB Process Tracking view and observe that there are now 15 tracking messages:



Event	ID	Timestamp	Step
PROCESS_ENTRY	ID:B2RT.FileTransfer...	Thu Jan 10 12:08:...	Validate PO
SERVICE_ENTRY	ID:B2RT.FileTransfer...	Thu Jan 10 12:08:...	Validate PO
SERVICE_EXIT	ID:B2RT.FileTransfer...	Thu Jan 10 12:11:...	Validate PO
SERVICE_EXIT	ID:B2RT.FileTransfer...	Thu Jan 10 12:11:...	Validate PO
SERVICE_ENTRY	ID:B2RT.FileTransfer...	Thu Jan 10 12:11:...	Split XML into Multiple Messages
SERVICE_ENTRY	ID:B2RT.FileTransfer...	Thu Jan 10 12:11:...	Write Original File to Directory B2RT_A
SERVICE_EXIT	ID:B2RT.FileTransfer...	Thu Jan 10 12:11:...	Write Original File to Directory B2RT_A
PROCESS_EXIT	ID:B2RT.FileTransfer...	Thu Jan 10 12:11:...	Write Original File to Directory B2RT_A
SERVICE_EXIT	ID:B2RT.FileTransfer...	Thu Jan 10 12:14:...	Split XML into Multiple Messages
SERVICE_EXIT	ID:B2RT.FileTransfer...	Thu Jan 10 12:14:...	Split XML into Multiple Messages
SERVICE_EXIT	ID:B2RT.FileTransfer...	Thu Jan 10 12:14:...	Split XML into Multiple Messages
SERVICE_ENTRY	ID:B2RT.FileTransfer...	Thu Jan 10 12:14:...	Write Split File to Directory B2RT_B
SERVICE_EXIT	ID:B2RT.FileTransfer...	Thu Jan 10 12:17:...	Write Split File to Directory B2RT_B
PROCESS_EXIT	ID:B2RT.FileTransfer...	Thu Jan 10 12:17:...	Write Split File to Directory B2RT_B
SERVICE_ENTRY	ID:B2RT.FileTransfer...	Thu Jan 10 12:17:...	Write Split File to Directory B2RT_B

6. After you finish debugging, click **Terminate** to end the debug session:



7. Select **File > Exit** to close Sonic Workbench.

To explore this sample application further, you can set different breakpoints and debug other scenarios. In addition, you can modify other aspects of the sample and view the results.

You have now completed the Batch to Real-time tutorial. See the next steps after the Batch to Real-time tutorial.

Next steps after developing the Batch to Real-time sample application

If you have not developed the sample application in the Remote Information Access tutorial, you can do so now. Batch to real time and remote information access are two of the most common enterprise integration scenarios that benefit from a Sonic ESB SOA solution.

Two other common scenarios that benefit from a Sonic ESB SOA solution include:

- Remote data distribution
- Respond to real-time business events

You can go to the Progress Software Developers Network (PSDN) at `http://www.psdn.com` to learn more about these and other enterprise integration scenarios.

The Sonic Workbench online help contains information on other sample applications included in your Sonic ESB installation. (Access the sample documentation from **Help > Help Contents > Progress Sonic ESB Product Family: Developer's Guide > Progress Sonic ESB Samples and Tutorials**). These sample applications illustrate different features of Sonic ESB and provide a useful next step in getting acquainted with this product:

- Integration pattern samples
- Split and Join service samples
- File Handling samples
- Audit service sample
- File polling sample
- Resubmit sample

Sonic BPEL Server also provides:

- Sonic BPEL Server tutorial
- Sonic BPEL Server samples

You can also learn more about the concepts in the Batch to Real-time tutorial.

Reference: Files in the Batch to Real-time sample project

When you double-click the files in the Sample.B2RT project in the Navigator view, they open in the appropriate editor in Sonic Workbench. You use the following files in this tutorial:

- `\resources` — Folder containing `PurchaseOrders.xsd`, the XML schema that defines the XML elements used in the purchase order files
- `\Sample Data` — Folder containing the purchase order files you can use to test the FileTransfer process:
  - `PurchaseOrder.xml` — XML file containing one purchase order
  - `PurchaseOrders.xml` — XML file containing three purchase orders
- `DropinDirA.drop` — Configuration document that specifies how to drop the bulk purchase order files
- `DropinDirB.drop` — Configuration document that specifies how to drop the split purchase order files
- `SampleB2RT.FileTransfer.esbp` — FileTransfer ESB process that you develop in the tutorial
- `SplitXMLonXPath.xsl` — Stylesheet that uses an XPath expression to split the bulk purchase order files into individual purchase order files

PurchaseOrders.xsd

`PurchaseOrders.xsd` is an XML schema, which opens in the XML Schema editor. The **Validate PO** step uses this file to perform XML schema validation on the incoming bulk purchase order files. This XML schema defines two elements used in the XML files containing the purchase orders:

- `PurchaseOrder` — Purchase order element, used in PurchaseOrder.xml and PurchaseOrders.xml
- `PurchaseOrders` — List of purchase order elements, used in PurchaseOrders.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
 <xs:element name="PurchaseOrders">
  <xs:complexType>
   <xs:sequence>
    <xs:element maxOccurs="unbounded" ref="PurchaseOrder"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="PurchaseOrder">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="Customer"/>
    <xs:element ref="Address"/>
    <xs:element ref="City"/>
    <xs:element ref="State"/>
    <xs:element ref="Zip"/>
    <xs:element ref="Part"/>
    <xs:element ref="Qty"/>
   </xs:sequence>
   <xs:attribute name="Name" use="required" type="xs:integer"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="Customer" type="xs:string"/>
 <xs:element name="Address" type="xs:string"/>
 <xs:element name="City" type="xs:NCName"/>
 <xs:element name="State" type="xs:NCName"/>
 <xs:element name="Zip" type="xs:integer"/>
 <xs:element name="Part" type="xs:NMTOKEN"/>
 <xs:element name="Qty" type="xs:integer"/>
</xs:schema>
```

PurchaseOrder.xml

`PurchaseOrder.xml` is an XML file, which opens in the XML editor. This file uses the `PurchaseOrder` element defined by the XML schema.and contains only one purchase order. (This file is not used in the tutorial; however, you could use it for further exploration.)

```
<PurchaseOrder Name="11122"
      xsi:noNamespaceSchemaLocation="sonicfs:///workspace/Sample.B2RT/resources/PurchaseOrders.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Customer>Acme Manufacturing</Customer>
   <Address>10 Main Street</Address>
   <City>Bedford</City>
   <State>MA</State>
   <Zip>01730</Zip>
   <Part>45-1234</Part>
   <Qty>10</Qty>
</PurchaseOrder>
```

PurchaseOrders.xml

`PurchaseOrders.xml` is an XML file, which opens in the XML editor. This file uses the `PurchaseOrder` and `PurchaseOrders` elements defined by the XML schema. `PurchaseOrders.xml` is the bulk purchase order file that you use to create a scenario to test the FileTransfer process.

```
<PurchaseOrders
      xsi:noNamespaceSchemaLocation="sonicfs:///workspace/Sample.B2RT/resources/PurchaseOrders.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <PurchaseOrder Name="11122">
      <Customer>Acme Manufacturing</Customer>
      <Address>10 Main Street</Address>
      <City>Bedford</City>
      <State>MA</State>
      <Zip>01730</Zip>
      <Part>45-1234</Part>
      <Qty>10</Qty>
   </PurchaseOrder>
   <PurchaseOrder Name="11133">
      <Customer>Fox Printing</Customer>
      <Address>10 Main Street</Address>
      <City>Arlington</City>
      <State>MA</State>
      <Zip>02476</Zip>
      <Part>45-1268</Part>
      <Qty>20</Qty>
   </PurchaseOrder>
   <PurchaseOrder Name="11144">
      <Customer>Bill's Supplies</Customer>
      <Address>10 Main Street</Address>
      <City>Reading</City>
      <State>PA</State>
      <Zip>19601</Zip>
      <Part>45-1498</Part>
      <Qty>25</Qty>
   </PurchaseOrder>
</PurchaseOrders>
```

DropinDirA.drop

`DropinDirA.drop` is a configuration document (`.drop` file), which opens in the XML editor. Every File Drop service uses a configuration document to load its runtime parameters.

`DropinDirA.drop` is the configuration document that you modify for the **Write Original File to Directory B2RT_A** step. This configuration document specifies that the File Drop service will drop files named `OriginalPOList.xml` in directory `B2RT_A`:

```
<?xml version="1.0" encoding="UTF-8"?>
<fileDropRequest>
   <statusDestinations>
      <destinationAddress>
         <type>exit</type>
      </destinationAddress>
   </statusDestinations>
   <filename type="GENERATED">OriginalPOList.xml</filename>
   <!-- TODO: Change the following line to the Target Directory (usually an explicit path) -->
   <dropDirectory>C:\Sonic\ESB7.6\samples\FileDropOutputFiles/B2RT_A</dropDirectory>
   <components>
      <fileDrop id="DEFAULT_DROP"/>
   </components>
</fileDropRequest>
```

DropinDirB.drop

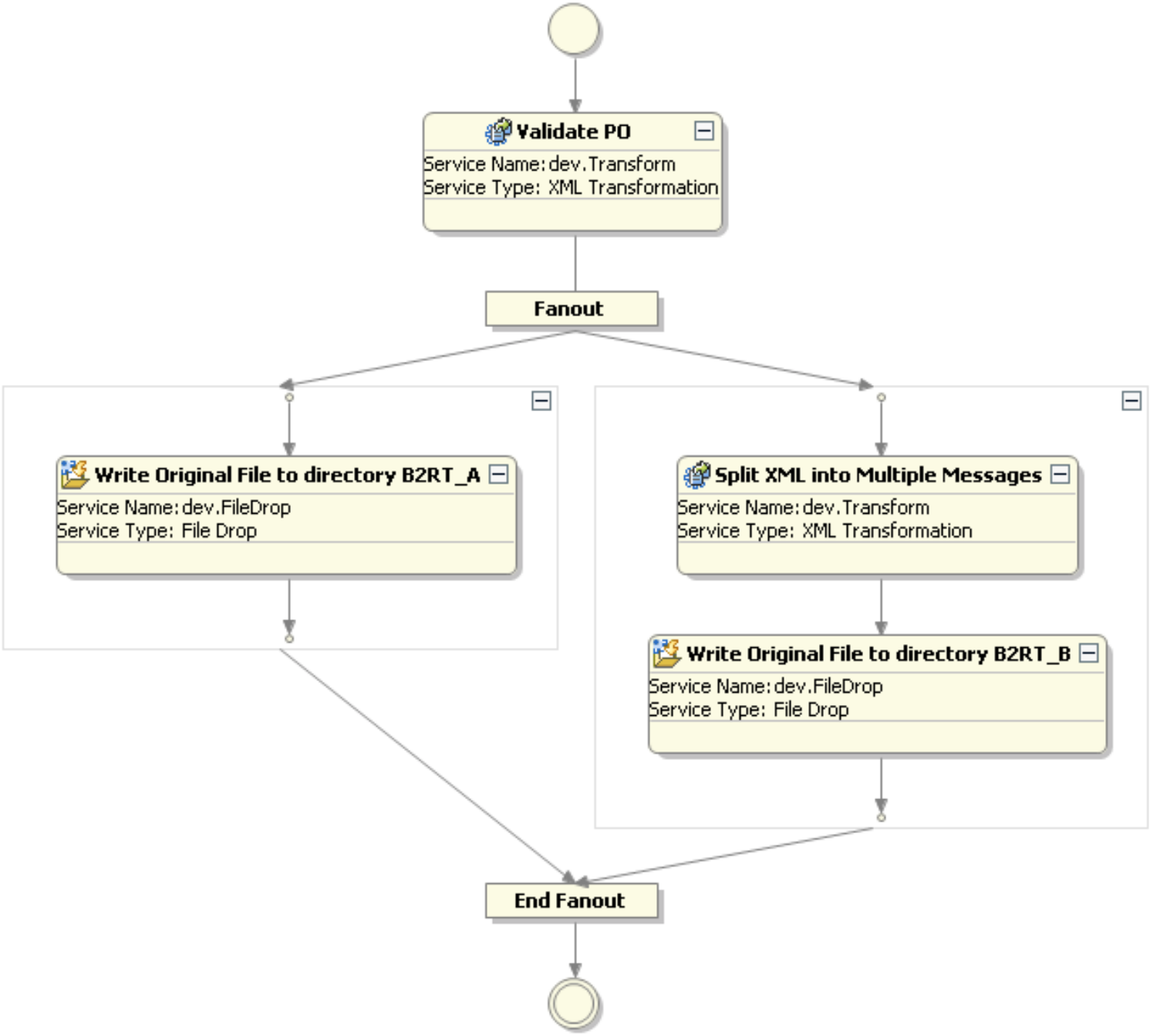
`DropinDirB.drop` is a configuration document (`.drop` file), which opens in the XML editor. Every File Drop service uses a configuration document to load its runtime parameters.

`DropinDirB.drop` is the configuration document that you modify for the **Write Split File to Directory B2RT_B** step. This configuration document specifies that the File Drop service will drop files named `DroppedPO.xml` in directory `B2RT_B`:

```
<?xml version="1.0" encoding="UTF-8"?>
<fileDropRequest>
   <statusDestinations>
      <destinationAddress>
         <type>exit</type>
      </destinationAddress>
   </statusDestinations>
   <filename type="GENERATED">DroppedPO.xml</filename>
   <dropDirectory>C:\Sonic\ESB7.6\samples\FileDropOutputFiles/B2RT_B</dropDirectory>
   <components>
      <fileDrop id="DEFAULT_DROP"/>
   </components>
</fileDropRequest>
```

SampleB2RT.FileTransfer.esbp

`SampleB2RT.FileTransfer.esbp` is the FileTransfer ESB process, which opens in the ESB Process editor. The FileTransfer process contains the services used in the Batch to Real-time sample application. You can expand each step to show more detail:



SplitXMLonXPath.xsl

`SplitXMLonXPath.xsl` is a stylesheet, which opens in the XSLT editor. The **Split XML into Multiple Messages** step uses and XPath expression in this stylesheet to split the bulk purchase order files into individual purchase order files. It splits the bulk purchase order files according to the XPath node, `PurchaseOrders/PurchaseOrder`that you find:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:XQMessageElem="http://www.sonicsw.com/sonicxq/com.sonicsw.xq.service.xform.TransformationElementFactory"
    extension-element-prefixes="XQMessageElem">
  <xsl:template match="/">
      <xsl:for-each select="/PurchaseOrders/PurchaseOrder">
      <XQMessageElem:XQMessage>
         <XQMessage>
           <part>
             <content><xsl:copy-of select="."/></content>
           </part>
         </XQMessage>
       </XQMessageElem:XQMessage>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```