BEA WebLogic Integration™

Samples

Version 8.1 Service Pack 3
Document Date: June 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

CHAPTER 1

Samples

This guide provides information and instructions for the WebLogic Integration samples.

WebLogic Integration also provides the following two tutorials:

- Tutorial: Building Your First Workflow
- Tutorial: Building Your First Data Transformation.

Topics Included in This Section

Chapter , "Non-XML Data Mapping Sample"
Provides information about a purchase order sample designed to illustrate the basic techniques of creating and testing message format definitions for non-XML data using Format Builder.

WebLogic Integration Use Case Scenarios

The Beta Release includes several use case samples that demonstrate various capabilities and features of WebLogic Integration. The use case documents are located in the following directory:

BEA_HOME\weblogic81\samples\integration\sampleApp

In the preceding path, *BEA_HOME* represents the directory in which you installed WebLogic Platform.

The following use case documents are available:

- *Asynchronous EJB Invoke Sample* (EJBInvocation.html)—This sample demonstrates the use of Client Request and Client Response operations, the use of an EJB control, the use of a Transformation control, and the For Each operation.
- *Parallel Workflow Sample* (ParallelWorkflow.html)—This sample consists of several tasks, including running a simple asynchronous workflow that receives an XML document over HTTP; validation of an XML document against a schema definition, publishing and Subscribing for messages using message broker control to different workflows, and collection of responses from the EJB control into single XML document.
- *InsertBasedEventDemo* (InsertBasedEventDemo.html)—This sample a number of features, including application view control, events and services, handling of service errors, and triggering of a workflow by an event.
- *Migration Use Case for WebLogic Integration Beta* (MigrationBeta.html)—This sample demonstrates migrating workflows from WebLogic Integration 7.0 to WebLogic Integration 8.1 Beta.
- *Complex Asynchronous Workflow Sample* (ComplexAsynch.html)—This sample demonstrates asynchronous interactions between two complex workflows.
- *Web Services Content Based Routing Sample* (WebServicesRouting)—This sample demonstrates content-base routing of XML documents between web services via workflow.

CHAPTER 2

Non-XML Data Mapping Sample

The WebLogic Integration software includes a purchase order sample designed to illustrate the basic techniques of creating and testing message format definitions for non-XML data using Format Builder. The purchase order sample consists of MFL and DATA files. This sample can be used to learn more about using non-XML data in WebLogic Integration and to test your installation of the non-XML mapping functionality of WebLogic Integration.

The following topics are discussed in this section:

Related Topics

To learn more about using non-XML data in WebLogic Integration and the using the Format Builder for creating and testing message format definitions, see Guide to Data Transformation.

For additional information about the Format Builder, see the help included with the Format Builder executable. (To access the Format Builder help, start the Format Builder as described in "Step 1. Start the Format Builder and Create a Message Format" on page -8 and then from the **Format Builder** menu bar, choose **Help→Help Topics**.)

What Is Included in the Purchase Order Sample

The following table describes the files provided with the purchase order sample application. All directory names are relative to the WebLogic Integration samples directory (*WLP_HOME*\integration\samples\di) where *WLP_HOME* is the top-level directory of your WebLogic Platform installation. (For example, if you installed WebLogic Platform in the c:\bea directory, the di directory is located at the following path: c:\bea\weblogic81\integration\samples\di.)

Table 2-1 List of Purchase Order Sample Application Files

Directory	File	Description
po	po_01.data	Purchase order data in non-XML format.
	po_02.data	Additional purchase order data in non-XML format.
	po.mfl	Prebuilt message format description of purchase order data.

Prerequisite Considerations

To understand how the Format Builder is used, it helps to understand the data formats used by the data integration tools provided by WebLogic Integration: non-XML data, XML, and MFL. If you have not already done so, please review *Supported Data Types* section of the Format Builder help. (To access the Format Builder help, start the Format Builder as described in "Step 1. Start the Format Builder and Create a Message Format" on page -8 and then from the **Format Builder** menu bar, choose **Help→Help Topics**.)

Performing Non-XML to XML Transformation

The following sections provide information about building a sample purchase order format definition and testing the transformation of non-XML data into XML format:

- Analyzing the Data to Be Transformed
- Creating the Format Definition and Testing the Transformation

You can build format definitions that provide the information required to transform non-XML data to or from XML. Format definitions are the metadata used to parse the content of a non-XML data file.

Analyzing the Data to Be Transformed

The key to translating non-XML data to and from XML is to create an accurate description of the non-XML data. For non-XML data (data that is not self-describing), you must identify the following elements:

- Data fields
- Data field attributes, such as name, data type, length/termination, optional, repeating
- Groups of related fields
- Group attributes, such as name, optional, repeating, delimited
- Hierarchical structure (groups of fields and/or other groups)

The following listing shows sample non-XML data that is included with WebLogic Integration in the *WLP_HOME*\integration\samples\di\po\po_01.data file, where *WLP_HOME* is the top-level directory of your WebLogic Platform installation. (For example, if you installed WebLogic Platform in the c:\bea directory, the po directory is located at the following path: c:\bea\weblogic81\integration\samples\di\po.)

In this sample, the data is taken from a fictitious purchase order on a proprietary system used by the XYZ Corporation. XYZ wants to exchange information with another system that accepts XML data.

```
1234;88844321;SUP:21Sprockley's Sprockets01/15/2000123 Main St.;
Austin;TX;75222;555 State St.;Austin;TX;75222;PO12345678;
666123;150;Red Sprocket;
```

To analyze the purchase order data:

1. Generate a definition of the data. To do so, you may need to use printed specifications or internal documentation. For this sample, we have described the purchase order format in Table 2-2.

Table 2-2 Purchase Order Master Record

Category	Field Name	Data Type	Length	Description
Basic Information	Purchase Request Number	Numeric	Delimited by semicolon	The Purchase Request number assigned by the Purchasing department. This number is used to track the status of an order from requisition through delivery and payment.
	Supplier ID	Numeric	Delimited by semicolon	The identification of the assigned supplier as defined in the corporate Supplier Data Base. Assignment of an approved supplier is made by the buyer when creating a Purchase Request from a requisition.
	Supplier Name	Character	Prefixed by a literal "SUP:". Following this literal is a two digit numeric length field.	The name of the assigned supplier as defined in the corporate Supplier Data Base. This field is prefixed with a literal to indicate that it is present.
	Requested Delivery Date	Date *MM/DD/YYYY*	10 characters	The delivery date specified by the requisitioner.
Shipping Address	Street	Character	Delimited by semicolon	The street address to be used in shipping the requested items.
	City	Character	Delimited by semicolon	The city to be used in shipping the requested items.
	State	Character	Delimited by semicolon	The state to be used in shipping the requested items.
	Zip	Numeric	Delimited by semicolon	The zip code to be used in shipping the requested items.

Table 2-2 Purchase Order Master Record (Continued)

Category	Field Name	Data Type	Length	Description
Billing Address	Street	Character	Delimited by semicolon	The street address to be used for billing.
	City	Character	Delimited by semicolon	The city to be used for billing.
	State	Character	Delimited by semicolon	The state to be used for billing.
	Zip	Numeric	Delimited by semicolon	The zip code to be used for billing.
Payment Terms	Supported payment terms may be either Purchase Order or Company Credit Card. A literal preceding the payment information identifies the type.			
	PO Type	Character	Literal “PO”	Indicates PO payment terms.
	PO Number	Numeric	Delimited by semicolon	Purchase Order number.
	Credit Card Type	Character	Literal “CC”	Indicates Credit Card payment terms.
	Credit Card Number	Numeric	Delimited by semicolon	Credit card number.
	Credit Card Expiration Month	Numeric	Delimited by semicolon	Expiration month for credit card.
	Credit Card Expiration Year	Numeric	Delimited by semicolon	Expiration year for credit card.

Table 2-2 Purchase Order Master Record (Continued)

Category	Field Name	Data Type	Length	Description
Purchase Items	The following fields identify the items to be purchased. This information may be repeated for each item that is part of this Purchase Request. At least one item must be present.			
	Part Number	Numeric	Delimited by semicolon	The supplier's part number of the requested item.
	Quantity	Numeric	Delimited by semicolon	The quantity requested. Must be greater then zero.
	Description	Character	Delimited by semicolon	Description of the requested item.

2. Identify fields.

   A field is a sequence of bytes that is meaningful to an application. For example, in Table 2-2, Purchase Request Number, Supplier ID, Supplier Name, and so forth, are all fields.

3. Identify field attributes.

   Field attributes include the name of the field, the type of data stored in the field, the length of the field, and the delimiter that indicates the end of the field. For example, the Supplier ID field is delimited by a semicolon (;) indicating the end of the field data, but the Requested Delivery Date has an implied length of 10 characters.

4. Identify hierarchical groups.

   Groups are collections of fields, comments, and other groups or references that are related in some way. In Table 2-1, notice that the sample data defines a number of distinct groups: Shipping Address, Billing Address, Payment Terms, and Purchase Items.

5. Identify group attributes.

   You must define the attributes of the hierarchical groups. Group attributes include the name of the group, whether the group is optional, repeating, or delimited, or whether it is defined as a reference to another group. For example, because the same fields are required to define both a Shipping Address and a Billing Address (`Street`, `City`, `State`, `Zip`), you can define an `Address` group within the `Shipping_Address` group and set up a reference to it from within the `Billing_Address` group.

After you complete the preceding procedure, you may want to put the data into a spreadsheet, as shown in the example in Figure 2-1. This spreadsheet can then serve as a guide when you create your purchase order message definition.

Figure 2-1 Analysis of Purchase Order Data

1	Description	Group	Field	Reference	Optional	Name / Refers To	Data Type	Occurrence	Delimited by
2	Purchase Request Number		X			PR_Number	Numeric	1	Semicolon
3	Supplier ID		X			Supplier_ID	Numeric	1	Semicolon
4	Supplier Name		X		X	Supplier_Name	String	1	Numeric field length 2
5	Requested Delivery Date		X			Requested_Delivery_Date	Date MM/DD/YYYY	1	Semicolon
6	Shipping Address	X				Shipping_Address		1	
7	Address	X				Address		1	
8	Street		X			Street	String	1	Semicolon
9	City		X			City	String	1	Semicolon
10	State		X			State	String	1	Semicolon
11	Zip		X			Zip	Numeric	1	Semicolon
12	Billing Address	X				Billing_Address		1	
13	Address			X		Address		1	
14	Street			X		Street	String	1	Semicolon
15	City			X		City	String	1	Semicolon
16	State			X		State	String	1	Semicolon
17	Zip			X		Zip	Numeric	1	Semicolon
18	Payment Terms	X				Payment Terms		1*	Payment Terms Can either be a Purchase Order or Credit Card.
19	Purchase Order	X						1	
20	Purchase Order Tag		X			Payment_Type_PO	Literal "PO"	1	
21	Purchase Order Number		X			PO_Number	Numeric	1	Semicolon
22	Credit Card	X						1	
23	Payment Type		X			Payment_Type_CC	Literal "CC"	1	
24	Credit Card Number		X			CC_Number	Numeric	1	Semicolon
25	Credit Card Expire Month		X			CC_Expire_Month	Numeric	1	Semicolon
26	Credit Card Expire Year		X			CC_Expire_Year	Numeric	1	Semicolon
27	Purchase Items	X				Purchase_Items		1	
28	Item	X				Item		1- n	
29	Part Number		X			Part_Number	Numeric	1	Semicolon
30	Quantity		X			Quantity	Numeric	1	Semicolon
31	Description		X			Description	String	1	Semicolon
32	Carriage Return		X		X	Lit_CR	Literal "\r"	1	
33	Line Feed		X		X	Lit_LF	Literal "\n"	1	

Creating the Format Definition and Testing the Transformation

This section walks you through the process of creating a message format for translating the non-XML data to XML. To make sure you create your purchase order message format correctly,

you can compare the file you create with the *WLP_HOME*\integration\samples\di\po\po.mfl file included with WebLogic Integration, where *WLP_HOME* is the top-level directory of your WebLogic Platform installation. (For example, if you installed WebLogic Platform in the c:\bea directory, the po directory is located at the following path: c:\bea\weblogic81\integration\samples\di\po.)

Step 1. Start the Format Builder and Create a Message Format

To start Format Builder and create a message format:

1. Choose **Start→Programs→BEA WebLogic Platform 8.1→Development Tools→Format Builder**.

   The Format Builder main window is displayed.

2. Choose **File→New**.

   A new message format root node is created and displayed in the navigation tree.

3. Enter PurchaseRequest in the **Name/XML Root** field.

4. Click **Apply**.

   The name of the message format root node is updated.

Step 2. Create the Basic Information Fields

To create the fields required to capture the basic identifying information for the purchase order:

1. To create the PR_Number field, choose **Insert→Field→As Child**.

   A new field is added to the navigation tree (left pane). The default field properties are displayed in the detail window (right pane).

2. Define the properties for the field as described in the following table.

In this section . . .	Do the following . . .
Field Description	Enter PR_Number in the **Name** field.
	Select Numeric from the **Type** drop down list.
Field Occurrence	Verify that the **Once** check-box is selected.

In this section . . .	Do the following . . .
Field Attributes: Termination	Select the **Delimiter** check-box **Note:** You must select the **Delimiter** option and not the **Delimiter Field** option.
	Enter a semi-colon (`;`) in the **Value** field on the **Delimiter** tab.
Field Attributes: Code Page	Select the desired code page from the **Code Page** drop down list. A code page specifies the character encoding of the non-XML data in the field. For example, accept the default setting: `windows-1252 - Windows Latin-1`

Note: To learn more about the check-boxes and fields in the detail window (right pane), select the F1 key. A browser is displayed with a description of the check-boxes and fields.

Note: These values are determined by the analysis of the raw purchase order data shown in Figure 2-1.

3. Click **Apply**.

   The properties of the field are updated.

   Note: Because the only difference between the `PR_Number` field and the `Supplier_ID` field is the name you can use the Format Builder Duplicate feature to create the `Supplier_ID` field.

4. In the navigation tree (left pane), right-click the `PR_Number` field and select **Duplicate** from the drop-down menu.

   A duplicate field (`NewPR_Number`) is added as a sibling and becomes the current selection in the navigation tree.

5. Enter `Supplier_ID` in the **Name** field and click **Apply** to update.

6. To save your changes to the message format document:

   a. Choose **File→Save As.**

      The **Save As** dialog box is displayed.

b. Navigate to the *WLP_HOME*\integration\samples\di\po directory, where *WLP_HOME* is the top-level directory of your WebLogic Platform installation. (For example, if you installed WebLogic Platform in the c:\bea directory, the po directory is located at the following path: c:\bea\weblogic81\integration\samples\di\po.)

c. Enter the desired filename, for example, enter: my_po.mfl.

Note: The Format Builder automatically assigns the .mfl extension to message format files if no extension is specified.

d. Click **Save As**.

7. To add a the Supplier_Name field, choose **Insert→Field→As Sibling**.

   A new field is added to the navigation tree (left pane). The default field properties are displayed in the detail window.

8. Define the properties for the Supplier_Name field as described in the following table.

In this section . . .	Do the following . . .
Field Description	Enter Supplier_Name in the **Name** field.
	Select the **Optional** checkbox.
	In the **Type** drop-down menu. select String.
Field Occurrence	Verify that the **Once** check-box is selected.
Field Attributes	Select the **Field is Tagged** check-box.
	Enter the following in the Field is Tagged field: SUP:
Field Attributes: Termination	Select the **Imbedded Length** check-box.
	Select Numeric from the **Type** drop-down menu on the **Description** tab.
	Verify that the **Length** check-box is selected, and then enter 2 in the **Length** text box.

Note: To learn more about the check-boxes and fields in the detail window (right pane), select the F1 key. A browser is displayed with a description of the check-boxes and fields.

9. Click **Apply**.

   The properties of the field are updated.

   Note: The dotted-line box around the field icon in the navigation tree indicates that this field is optional.

10. To add the `Requested_Delivery_Date` field, choose **Insert→Field→As Sibling**.

    A new field is added to the navigation tree. The default field properties are displayed in the detail window.

11. Define the properties for the `Requested_Delivery_Date` field as described in the following table.

In this section . . .	Do the following . . .
Field Description	Enter `Requested_Delivery_Date` in the **Name** field.
	In the **Type** drop-down menu select `Date: MM/DD/YYYY` .
Field Occurrence	Verify that the **Once** check-box is selected.
Field Attributes	In the **Data Base Type** drop-down menu, verify that **String** is selected.

    Note: The contents of the detail window for a field is determined by the Type setting. When you select a date type from the drop down list, the length is implicitly determined. Therefore, the **Field Attributes: Termination** properties are no longer displayed.

    Note: To learn more about the check-boxes and fields in the detail window (right pane), select the F1 key. A browser is displayed with a description of the check-boxes and fields.

12. Click **Apply**.

    The properties of the field are updated.

13. Choose **File→Save** to save your changes.

Step 3. Create the Shipping Address and Billing Address Groups

To create the shipping address and billing address groups:

1. Select any field in the navigation tree and choose **Insert→Group→As Sibling**.

A new group is added to the navigation tree. The default group properties are displayed in the detail window.

2. Define the properties for the `Shipping_Address` group as described in the following table.

In this section . . .	Do the following . . .
Group Description	In the **Name** field, enter: `Shipping_Address` .
Group Occurrence	Verify that the **Once** check-box is selected.
Group Attributes	Verify that the **None** check-box is selected for **Group Delimiter**.

Note: These values are determined by the analysis of the raw purchase order data shown in Figure 2-1.

Note: To learn more about the check-boxes and fields in the detail window (right pane), select the F1 key. A browser is displayed with a description of the check-boxes and fields.

3. Click **Apply**.

   The properties of the group are updated.

   Note: Because the only difference between the `Shipping_Address` group and the `Billing_Address` group is the name, you can use the Format Builder Duplicate feature to create the `Billing_Address` field.

4. In the navigation tree (left pane), right-click the `Shipping_Address` group and select **Duplicate** from the drop-down menu.

   A duplicate group (`NewShipping_Address`) is added as a sibling and becomes the current selection in the navigation tree.

5. In the **Name** field, enter `Billing_Address` and click **Apply** to update.

   Because the `Shipping_Address` and `Billing_Address` groups contain the same fields with the same attributes, we can define an `Address` group within the `Shipping_Address` group and set up the `Address` group within the `Billing_Address` group as a reference.

6. In the navigation tree (left pane), select the `Shipping_Address` group and choose **Insert→Group→As Child**.

7. Use the data in Figure 2-1 to create the `Address` group and click **Apply** to update the group properties.
8. Follow the steps outlined in Step 2. Create the Basic Information Fields, and the data provided in Figure 2-1, to create the `Street`, `City`, `State`, and `Zip` fields as children of the `Address` group.

   Note: Once the `Street` field is created, you can use the **Duplicate** button to create the `City` and `State` fields.
9. To create a reference to the `Address` group, right-click the `Address` group under `Shipping_Address`, and select **Copy** from the drop-down menu.

   The `Address` group properties (including the child fields) are copied and placed on the clipboard.
10. Right-click the `Billing_Address` group and select **Paste As Reference** from the drop-down menu.

    The `Address` reference, is pasted immediately after the `Billing_Address` group. The arrow in the icon is identifies the group as a reference group:

    
11. To make the `Address` group reference a child of the `Billing_Address` group, select the reference, and then choose **Edit→Demote**.

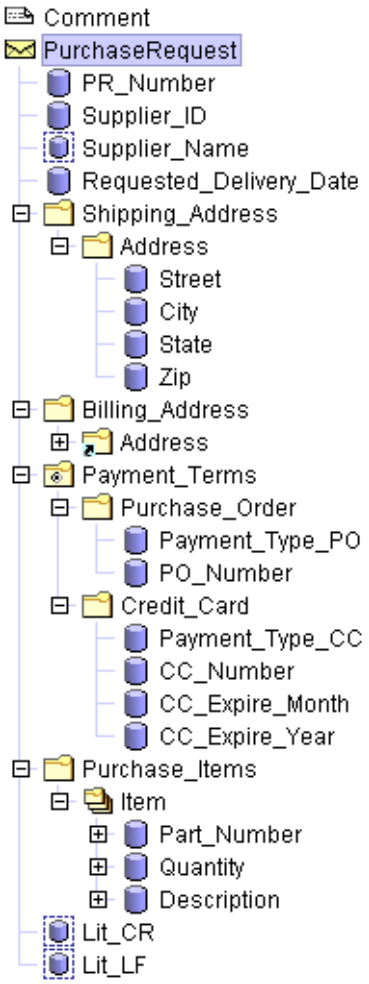
    The `Address` reference becomes a child of the `Billing_Address` group.
12. Choose **File→Save** to save your changes.

Step 4. Create the Remaining Items

Repeat the process of creating or duplicating fields and groups as required to complete the purchase order message format document. Use the analysis of the raw purchase order data presented in Figure 2-1 to determine the values you need to enter for each item. For assistance, refer to the *`WLP_HOME`*`\integration\samples\di\po\po.mfl` file, where *`WLP_HOME`* is the top-level directory of your WebLogic Platform installation. (For example, if you installed WebLogic Platform in the `c:\bea` directory, the `po` directory is located at the following path: `c:\bea\weblogic81\integration\samples\di\po`.)

When you finish entering the required items, your navigation tree (left pane) should look similar to the one shown in Figure 2-2.

Figure 2-2 Completed Navigation Tree for Purchase Order Sample



Step 5. Save the Completed Message Format

Choose **File→Save** to save your completed message format.

Step 6. Test the Message Format

To test the message format to identify any errors before using it to transform data:

1. Choose **Tools→Test** to display the **Format Tester** dialog box.
2. Choose **File→Open Non-XML** to display the **Open** dialog box.
3. Navigate to the *WLP_HOME*\integration\samples\di\po directory, where *WLP_HOME* is the top-level directory of your WebLogic Platform installation. (For example, if you installed WebLogic Platform in the c:\bea directory, the po directory is located at the following path: c:\bea\weblogic81\integration\samples\di\po.)
4. Select the PO_01.DATA file and click **Open**.

   The non-XML data is displayed in the **Non-XML** window.
5. Click **Transform→Non-XML To XML**.

   The content of the PO_01.DATA file is transformed to XML based on the active MFL document. The XML output is displayed in the XML window.

   Note: To view a description of each transformation step, choose **Display→ Debug** to open the **Debug** window, and then choose **Transform→Non-XML To XML**. A message is displayed for each step of the process.
6. If the transformed data appears to be correct, choose **File→Save XML**.
7. Type the name po.xml in the File name field, and then click **Save** to save the XML output.

Performing XML-to-Non-XML Transformation

You can also use the Format Builder to create message definitions and test the transformation of XML data to non-XML. The steps required to do this are essentially the same as those you follow to transform non-XML data to XML. To transform XML data to non-XML, first create an MFL description for the non-XML format. The purchase order sample files can be used to test the process as described in the following procedure.

1. Choose **File→Open** from the **Format Builder** menu.
2. Select the purchase order message format document.
3. Click **Open**.

   The message format document is displayed in the navigation tree.
4. Choose **Tools→Test** to display the **Format Tester** dialog box.
5. Choose **File→Open XML** from the **Format Tester** menu.

6. Navigate to the *WLP_HOME*\integration\samples\di\po directory, where *WLP_HOME* is the top-level directory of your WebLogic Platform installation. (For example, if you installed WebLogic Platform in the c:\bea directory, the po directory is located at the following path: c:\bea\weblogic81\integration\samples\di\po.)

7. Select the po.xml file and click **Open**.

   The XML data is displayed in the right pane.

8. Choose **Transform→XML to Non-XML**.

   The XML data is transformed, and the purchase order data is displayed, in non-XML format, in the right pane.

   Note: To view a description of each transformation step, choose **Display→ Debug** to open the **Debug** window, and then choose **Transform→XML to Non-XML**. A message is displayed for each step of the process.

9. If the transformed data appears to be correct, choose **File→ Save Non-XML**.

10. Type the name (for example test_po.data) in the **File name** field, and then click **Save** to save the non-XML output.

Using the Suppressible Attribute for a Static Subscription

A static subscription to a Message Broker channel can be specified as *suppressible* by setting the **suppressible** attribute for the subscription in the business process. Valid values for **suppressible** are **true** and **false** (the default). Setting **suppressible** to **true** specifies that the static subscription is suppressed in favor of dynamic subscriptions. In other words, you can prevent specific messages on a Message Broker channel from starting a new business process; instead the messages can be received, using a dynamic subscription, by a business process that is already running.

A typical use for the suppressible attribute is to design the routing of messages that have the same *group ID* to the same business process. An ideal scenario is to use a JMS property of messages handled by a JMS event generator as the group ID.

Designing such a scenario includes setting up the business process and the event generators, as described in the following sections:

Setting up the Business Process

In this example, we use a JMS property of messages handled by a JMS event generator as the group ID. We design the routing of messages with the same group ID to the same business process.

The following code example shows the property in the JMS event generator:

```
<JmsEventGenerator>
```

```
        <property>
                <Name>GROUPID</Name>
                <Value>100</Value>
        </property>
        ...
</JmsEventGenerator>
```

To Create a Business Process that Receives all Messages With the Same GROUPID Property Sent on a Specific Message Broker Channel

1. Design your business process to be started when it receives a message from a Message Broker channel to which it is subscribed. To learn how, see Subscription Start (Synchronous).

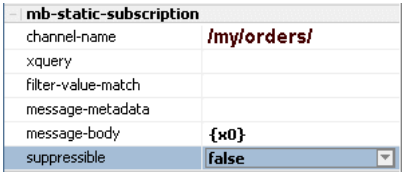
   Note: In this example, we use a channel named `/my/orders`.

2. Set the **suppressible** attribute to **true**. To do so:

   a. With the business process opened in the WebLogic Workshop graphical design environment, click the **Source View** tab, and locate the static subscription method in the source code. It is preceded by the following JPD annotation:

      ```
      @jpd:mb-static-subscription
      ```

   b. Click anywhere inside the static subscription method. The **Property Editor** displays a set of attributes (including **suppressible**) grouped under **mb-static-subscription**.



   c. In the **Property Editor**, select **true** for the **suppressible** attribute. The JPD annotation in the source code is updated to include the suppressible attribute:

      ```
      @jpd:mb-static-subscription suppressible="true"
      ```

3. Design the business process to receive a message.

4. Extract `$metadata/property[name='GROUPID']` from the message into a variable (say, the `myOrderNumber` variable).

5. Subscribe (dynamically) to the `/my/orders` channel using the `myOrderNumber` variable as a filter value. That is, use a filter expression with the same XPath expression as you used to extract the data into the variable in the preceding step:

   ```
   $metadata/property[name='GROUPID']
   ```

   To learn how to design a dynamic subscription, see Message Broker Subscription Control.

6. Design a loop to receive the remainder of the messages with the same `GROUPID` using the subscription control.

   To learn how to design looping logic, see Looping Through Items in a List.

Note: To avoid race conditions, steps 1-5 must be performed while in a synchronous message broker subscription. To learn about creating a synchronous subscription, see Subscription Start (Synchronous).

Setting up the Event Generators

For the suppressible pattern in this example to work correctly, there must be a single publisher for a given group ID. Having a single publisher for the group ID means that the publisher is *paused* when a new subscriber is subscribing dynamically (see steps 1-5 in the preceding section: Setting up the Business Process), thereby avoiding race conditions with the subscriptions. When you use JMS event generators, there are two ways to ensure that a single publisher publishes a given group ID:

- Set the JMS event generator pool size to 1. The JMS event generator is packaged as a message-driven bean pool.

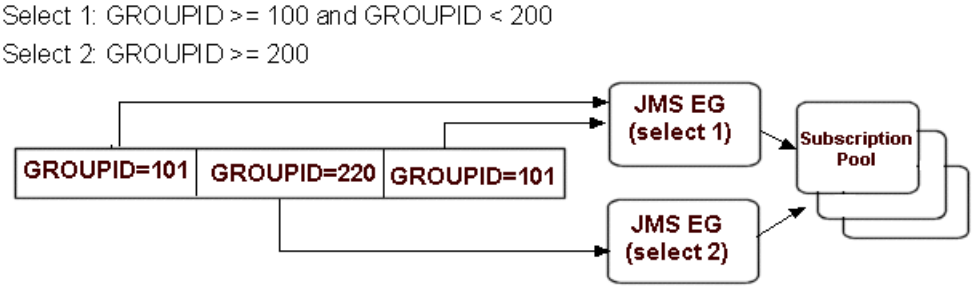
  You must manually edit the deployment descriptor to set the JMS event generator pool size. To do so, use the WebLogic Builder tool, which is available from the WebLogic Workshop design console menu bar:

  Tools→WebLogic Server→WebLogic Builder

  Note: You do not need to restrict the pool size of the subscribers.

- Create several JMS event generators each with a different message selector, and each with a pool size of 1. The message selector uniquely maps a group ID to a JMS event generator. In other words, one event generator is responsible for publishing all the events with a given group ID. The following figure illustrates how the `GROUPID=101` messages are published by a single JMS event generator (JMS EG (select1)).

  In the scenario illustrated in the following figure, a queue, which has `GROUPIDs` in the range 100-299, is defined. Two JMS event generators are defined, one to handle `GROUPIDs` in the range 100-199, and one to handle `GROUPIDs` in the range 200-299.



Because the static and dynamic subscriptions described in the preceding section (Setting up the Business Process) are within the Subscription Start (Synchronous) block, the JMS event generator that publishes the `GROUPID=101` messages is blocked after delivery of the first message until the dynamic subscription and any other logic within the synchronous subscription block is executed. Then the next `GROUPID=101` message is published. With **suppressible** set to **true** for the business process' static subscription, the second message does not start a new business process. Instead, it is received by the dynamic subscription on the business process that was started by receiving the first `GROUPID=101` message.