IBM WebSphere Partner Gateway Enterprise and
Advanced Editions

PIP Sample for WebSphere Process Server

Version 6.0.0.1

IBM WebSphere Partner Gateway Enterprise and
Advanced Editions

PIP Sample for WebSphere Process Server

Version 6.0.0.1

Note!
Before using this information and the product it supports, read the information in "Notices" on page 23.

13September2005

This edition applies to WebSphere Partner Gateway Enterprise Edition (5724-L69), Version 6.0.0.1, and Advanced Edition (5724-L68), Version 6.0.0.1, and to all subsequent releases and modifications until otherwise indicated in new editions

To send us your comments about this documentation, e-mail doc-comments@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

PIP Sample

This document describes a PIP sample provided with WebSphere Partner Gateway.

Topics covered in this chapter include:

WebSphere Partner Gateway PIP sample

WebSphere Partner Gateway provides the PIP sample to demonstrate how to set up WebSphere Partner Gateway and WebSphere Process Server to exchange messages when you implement WebSphere Process Server as a back-end application. Additionally, you can see how WebSphere Partner Gateway behaves when sending and receiving messages from a community participant.

Note: For information on how to design and implement your business processes, components, and integration solutions using WebSphere Process Server, refer to the WebSphere Process Server documentation.

Although this PIP sample can be used with a WebSphere Partner Gateway installation that uses either WebSphere Application Server v6.0 or the embedded version of WebSphere Application Server Express, the instructions in this guide assume that the WebSphere Partner Gateway installation is using WebSphere Application Server v6.0.

The PIP sample supports two scenarios. The first scenario demonstrates how WebSphere Partner Gateway handles a two-action PIP. The second scenario is a continuation of the first scenario in which the PIP is cancelled.

For additional information on integrating WebSphere Partner Gateway with WebSphere Process Server, see the *Enterprise Integration Guide*.

Topology used by the sample

Both of the scenarios use the same topology. As shown in Figure 1 on page 2, System A has WebSphere Process Server and performs the roles of back-end application and community participant. One process, the buyer process, initiates the PIPs and another process, the seller process, receives the PIPs.

System B has WebSphere Partner Gateway Enterprise Edition performing the role of PIP requester. This system receives the PIP content from the buyer process in System A and sends the PIP message to System C. System C has WebSphere Partner Gateway Enterprise Edition performing the role of PIP responder. This system receives PIP messages from System B and passes the content on to the seller process on System A.

Messages between System A and the other systems are handled by WebSphere platform messaging, running in the instance of WebSphere Application Server that is installed with WebSphere Process Server on System A.

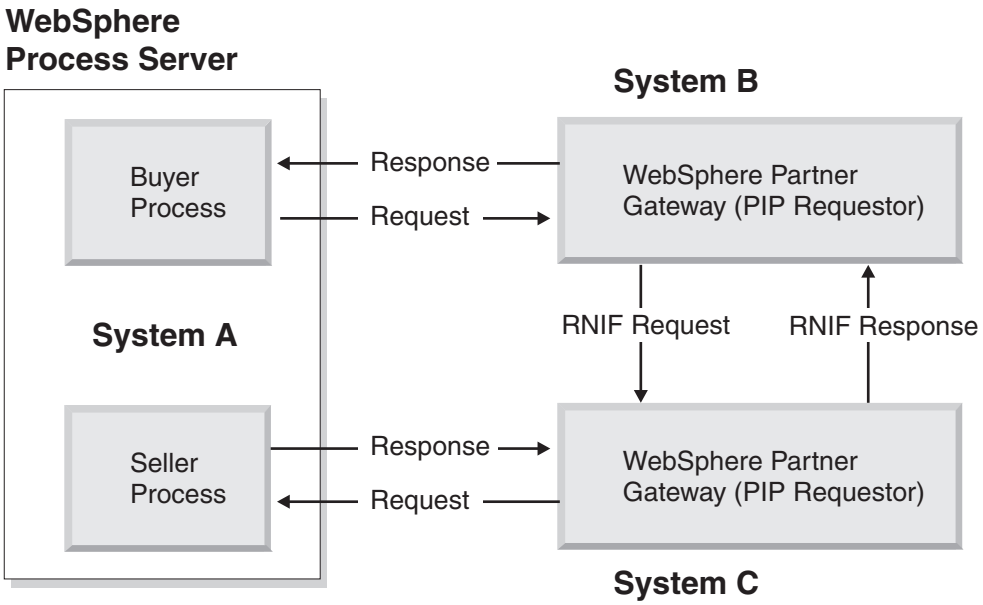

Figure 1. Topology used by the PIP sample.

Scenario 1: Processing a two-action PIP

Scenario 1 demonstrates how WebSphere Partner Gateway processes a two-action PIP as a sender and receiver. Figure 2 on page 3 shows the flow of PIP or PIP content messages among the systems in the scenario.

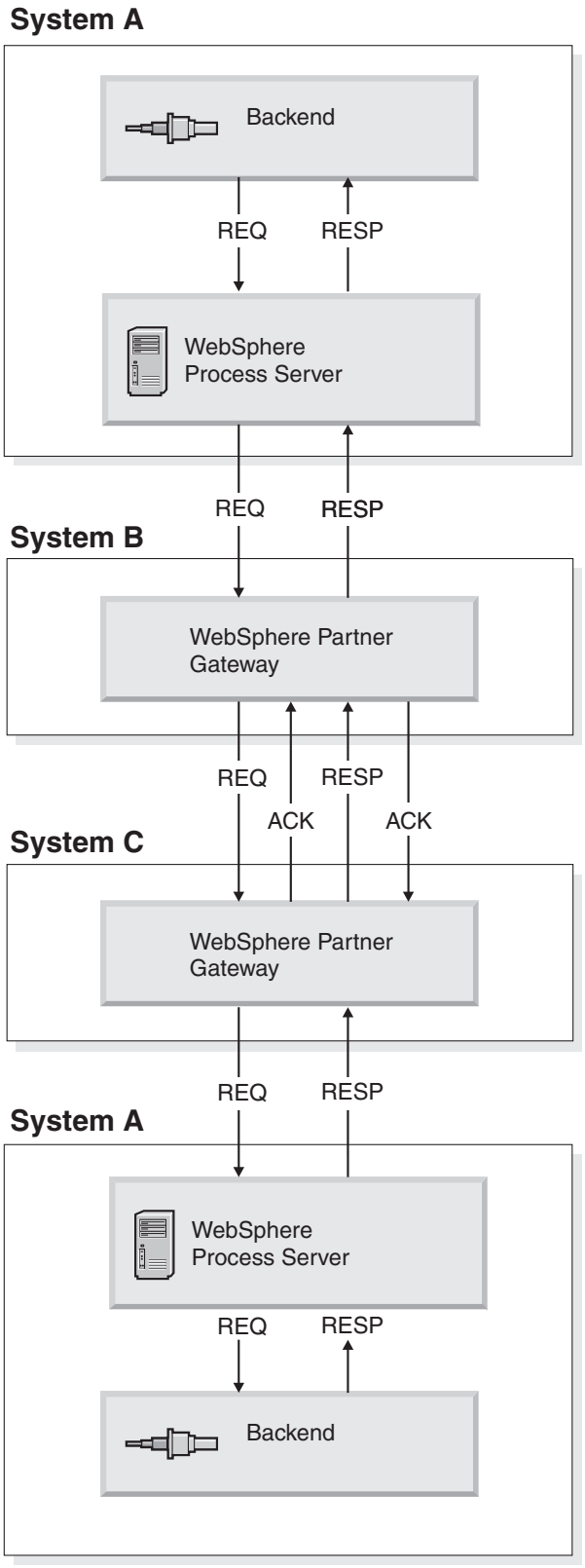

Figure 2. The flow of PIP or PIP content messages among the systems.

The scenario starts with the buyer process in WebSphere Process Server receiving a 3A4 request business object from the JSP file. This JSP file represents EIS for the buyer process. The buyer process creates a 3A4 request message and sets the

unique IDs (x_aux_process_instance_id and x_aux_system_msg_id) in the Backend Integration header of the message. The buyer process persists (stores in correlation sets and PIP3A4Buyer database) these IDs.

The buyer process sends the request on JMS to the WebSphere Partner Gateway instance configured as the buyer's gateway, which is the instance on System B. The buyer's gateway sends the RNIF request to the seller's gateway. The seller's gateway is the WebSphere Partner Gateway instance on System C. The seller's WebSphere Partner Gateway receives the RNIF request message, validates it and sends an acknowledgment signal to the buyer's gateway. The buyer's WebSphere Partner Gateway sends an EventNotification message with a statusCode of 100 to the buyer on the WebSphere Process Server. The buyer event dispatcher determines from the database table which buyer process needs to be invoked with this EventNotification message. The event dispatcher then invokes the PIP3A4Buyer process. Using correlation sets, the BPEL engine determines which PIP3A4Buyer process instance is waiting for this EventNotification message. This PIP3A4Buyer process instance receives the message and logs the message in the Systemout.log file.

The seller's WebSphere Partner Gateway packages the 3A4 request content in back-end integration packaging and sends this message to the seller process running in WebSphere Process Server. The seller process saves the IDs (in the correlation set and PIP3A4Seller database) contained in the Backend Integration packaging and logs the message in the Systemout.log file.

The JSP file on the sellers side represents EIS for the seller process. The JSP file (at the seller side) asynchronously sends the PIP 3A4 response message (confirmation) to the seller process. Using correlation sets, the BPEL engine determines which PIP3A4Seller process instance is waiting for this PIP3A4 response message. This PIP3A4Seller process instance receives the PIP3A4 response message. The seller sends the response message to the seller's WebSphere Partner Gateway, which packages the response in RNIF format and sends it to the buyer's gateway. The seller process update the database contents and stops the seller process.

The buyer's WebSphere Partner Gateway validates the response and sends an acknowledgment back to the seller's gateway, which in turn sends an Event Notification message with a status code of 100 to the seller event dispatcher in the WebSphere Process Server. The buyer's WebSphere Partner Gateway asynchronously sends the PIP 3A4 response message to the buyer process in WebSphere Process Server. Using correlation sets, the BPEL engine determines which PIP3A4Buyer process instance is waiting for this PIP3A4 response message. This PIP3A4Buyer process instance receives the PIP3A4 response message. This PIP3A4 process instance then updates the contents of database table, logs a message in the Systemout.log file, and stops the buyer process.

Scenario 2: Processing a 0A1PIP

Scenario 2 is a continuation of Scenario 1. Figure 3 on page 5 shows the messaging of the first scenario and the messaging used to cancel the PIP, which is Scenario 2.

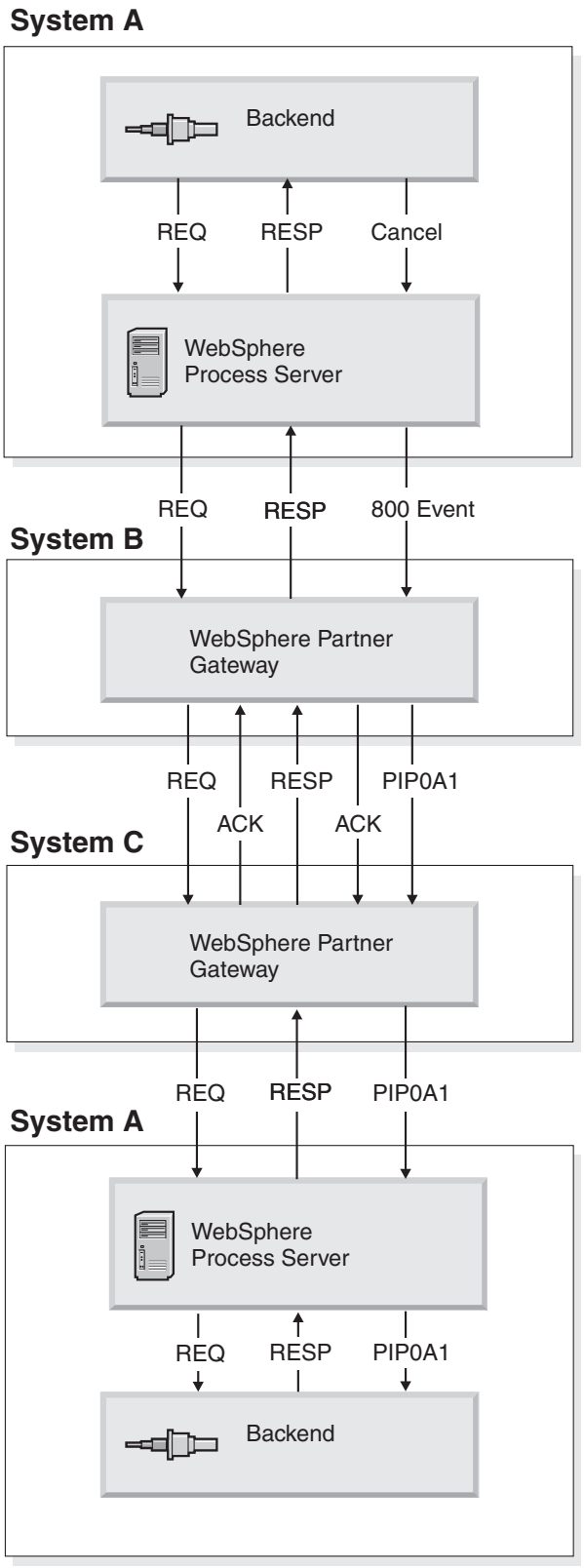

Figure 3. The flow of PIP or PIP content messages among the systems in scenario 2.

After the buyer process receives the response and successfully logs the message into the Systemout.log file, it stops the buyer process. The buyer event dispatcher receives a cancellation event from the JSP file of the buyer process. From the

cancellation event, the buyer event dispatcher determines which process this event should be dispatched. From the database table, the buyer event dispatcher determines if the process instance referenced in the cancellation event is still running. If the PIP3A4Buyer instance is still running, the event dispatcher invokes this process instance with the cancellation event. The JSP page at the buyer process then populates an Event Notification message with the following information:

Table 1. Scenario 2 Event notification field values

Field	Value
StatusMessage	Text that indicates that the application that sent the 3A4 PIP request has cancelled it
StatusCode	″800″ to indicate that the Event Notification message is to cancel a PIP
EventMessageID	Identifier for this Event Notification message
BusinessObjectID	Identifier of the PIP request to be cancelled. This is the value of DocumentId of the original PIP request.
GlobalMessageID	Identifier of the PIP request message. This is the value in the Msgid column in the database table used to store message metadata

The buyer event dispatcher then sends the event notification message to its gateway. The buyer's WebSphere Partner Gateway instance receives the event notification message and generates a 0A1 PIP based on the message. The instance sends the 0A1 PIP to the seller's gateway. The seller's WebSphere Partner Gateway instance receives the PIP 0A1 message and sends it to the seller to the seller event dispatcher.

WebSphere Process Server artifacts

The PIP sample consists of four business integration modules: PIP3A4Buyer, BuyerEventDispatcher, PIP3A4Seller, and SellerEventDispatcher. PIP3A4Buyer and BuyerEventDispatcher are for the buyer side, and PIP3A4Seller and SellerEventDispatcher are for the seller side. The PIP sample uses the index.jsp file, business objects, WSDL Interfaces, and BPEL processes listed in this section to support the scenarios.

Business objects

The PIP sample uses the following business objects:

- **PIP3A4Request** - The following business objects are required for PIP 3A4 request messages:
  - **BCG_PIP3A4PurchaseOrderRequest** - This is the top-level object for the PIP 3A4 request message. The same object is used for both scenarios 1 and 2. The same business object structure is used for the buyer and seller processes with different namespaces. For the buyer process, it uses the http://PIP3A4Buyer namespace, and for the seller process, it uses the http://Partner namespace. This business object contains the attachment business object (of type array), package headers business object, and payload wrapper business object as child business objects.
  - **BCG_PayloadContainer_Pip3A4PurchaseOrderRequest** - This is the wrapper business object for the PIP3A4 request. This business object contains the information about the payload, such as encoding and content type. The same business object structure is used for the buyer and seller processes with

different namespaces. For the buyer process, it uses the http://PIP3A4Buyer namespace, and for the seller, it uses the http://Partner namespace.

  - **Pip3A4PurchaseOrderRequest** - This is the payload business object of the PIP3A4 request. This represents service content of the PIP 3A4 request. Refer to the *Enterprise Integration Guide* for information about how this business object can be created.

- **PIP3A4Response** - The following business objects are required for the PIP 3A4 response message:
  - **BCG_PIP3A4PurchaseOrderConfirmation** - This is the top-level object for the PIP 3A4 confirmation message. The same object is used for both scenarios 1 and 2. The same business object structure is used for the buyer and seller processes with different name spaces. For the buyer process, it uses the http://Partner namespace, and for seller, it uses the http://PIP3A4Seller namespace. This business object contains the attachment business object (of type array), package headers business object, and payload wrapper business object as child business objects.
  - **BCG_PayloadContainer_Pip3A4PurchaseOrderConfirmation** - This is the wrapper business object for the PIP3A4 confirmation. This business object contains the information about the payload, such as encoding and content type. The same business object structure is used for the buyer and seller processes with different namespaces. For the buyer process, it uses the http://Partner namespace, and for seller process, it uses the http://PIP3A4Seller namespace.
  - **Pip3A4PurchaseOrderConfirmation** - This is the payload business object of the PIP3A4 confirmation. This represents the service content of PIP 3A4 response. Refer to the *Enterprise Integration Guide* for information about how this business object can be created.

- **EventNotification** - For EventNotification, the following business objects are required and reside on the BuyerEventDispatcher and Seller EventDispatcher modules:
  - **BCG_EventNotification** - This is the top-level object for the event notification message. The same object is used for both scenarios 1 and 2. The same business object structure is used for the buyer and seller event dispatchers with the same namespaces, http://EventNotification. This business object contains the attachment business object (of type array), package headers business object, and payload wrapper business object, as child business objects.
  - **BCG_PayloadContainer_EventNotification** - This is the wrapper business object for the event notification message. This business object contains the information about the payload, such as encoding and content type. The same business object structure is used for the buyer and seller event dispatchers with the same namespace, http://EventNotification.
  - **EventNotification** - This is the payload business object of the event notification message. This business object is in the namespace, which is http://www.ibm.com/websphere/bcg/2003/v1.0/xmleventnotification. The PIP sample already has these business objects. However, if you need to obtain this business object, you can follow following steps:
    1. Navigate to the Package_RNIFV02.02.zip file in the WebSphere Partner Gateway product directory.
    2. Extract the zip file to a temporary directory.
    3. XMLEvent_v1.0ns.xsd gives the business object.

- **Pip0A1FailureNotification** - For FailureNotification, the following business objects are required and reside on BuyerEventDispatcher and Seller EventDispatcher:
  - **BCG_Pip0A1FailureNotification** - This is the top-level object for the event notification message. This object is used for scenario 2. The same business object structure is used for the buyer and seller event dispatcher with the same namespace, http://PIP0A1. This business object contains the attachment business object (of type array), package headers business object, and payload wrapper business object, as child business objects.
  - **BCG_PayloadContainer_ Pip0A1FailureNotification** - This is the wrapper business object for the event notification message. This business object contains the information about the payload, such as encoding and content type. The same business object structure is used for the buyer and seller processes with the same name space, http://PIP0A1.
  - **Pip0A1FailureNotification** - This is the payload business object of the PIP 0A1 failure notification. This represents the service content of PIP 0A1 failure notification. Refer to the *Enterprise Integration Guide* for information about how this business object can be created.
- **BCG_AttachmentType** - The attachment container is the common business object for request, confirmation, event notification, and failure notification. The same business object structure is used for the buyer processes, buyer event dispatcher, seller processes, and seller event dispatcher with different namespaces, and each process and event dispatcher contains two different instances of the business object. For the buyer process, it uses the http://PIP3A4Buyer namespace for request and the http://Partner namespace for confirmation, and for the seller process, it uses the http://PIP3A4Seller namespace for confirmation and the http://Partner namespace for request. For the event dispatchers, it uses the http://EventNotification and http://PIP0A1 for failure notification message.
- **BCG_PackagingHeaders** - The packaging headers business object is the common business object for request, confirmation, event notification, and failure notification. This business object contains the information about the JMS headers, content type, and length. The same business object structure is used for the buyer and seller processes and event dispatchers with different namespaces, and each process or event dispatcher contains two different instances of the business object. For the buyer process, it uses the http://PIP3A4Buyer namespace for request, and the http://Partner namespace for confirmation, and for the seller process, it uses the http://PIP3A4Seller namespace for the confirmation, and the http://Partner namespace for request. For the event dispatchers, it uses the http://EventNotification and http://PIP0A1 for failure notification message.

WSDL Interfaces

The PIP sample uses the following interfaces:

- **ReceiveRequest** - This interface is used for receiving the request from the JSP file.
  Operation Name: receiveRequestMessage
  Input variable: recvReq
  Input parameter type: BCG_Pip3A4PurchaseOrderRequest
  Operation type: One-way
- **SendRequest** - This interface is used for sending the request to WebSphere Partner Gateway.
  Operation Name: sendRequestMessage

Input variable: sendReq
Input parameter type: BCG_Pip3A4PurchaseOrderRequest
Operation type: One-way

- **ReceiveConfirmation** - This interface is used for receiving the confirmation message from WebSphere Partner Gateway (for the buyer) or from the JSP file (for the seller).
  Operation Name: receiveConfirmationMessage
  Input variable: recvConfirm
  Input parameter type: BCG_Pip3A4PurchaseOrderConfirmation
  Operation type: One-way
- **SendConfirmation** - This interface is used for sending the confirmation message to WebSphere Partner Gateway.
  Operation Name: sendConfirmationMessage
  Input variable: sendConfirm
  Input parameter type: BCG_Pip3A4PurchaseOrderConfirmation
  Operation type: One-way
- **BCGToEventDispatcher** - This interface is used for receiving the event/failurenotification from WebSphere Partner Gateway. This interface has the following two operations:
  1
  Operation Name: processEvent
  Input variable: eventBO
  Input parameter type: BCG_EventNotification
  Operation type: One-way
  2
  Operation Name: processPIP0A1
  Input variable: pipBO
  Input parameter type: BCG_Pip0A1FailureNotification
  Operation type: One-way
- **BackendToEventDispatcher** - This interface is used for receiving the event notification from the JSP file.
  Operation Name: processBackendEvent
  Input variable: eventBO
  Input parameter type: BCG_EventNotification
  Operation type: One-way
- **BackendEventNotifier** - This interface is used for sending the event notification to the PIP process.
  Operation Name: processBackendEvent
  Input variable: backendEventBO
  Input parameter type: BCG_EventNotification
  Operation type: One-way
- **BCGEventNotifier** - This interface is used for sending the event notification to WebSpher Partner Gateway.
  Operation Name: processBCGEvent
  Input variable: eventBO
  Input parameter type: BCG_EventNotification
  Operation type: One-way
- **PIP0A1Notifier** - This interface is used for sending the failure notification to WebSpher Partner Gateway.
  Operation Name: processPIP0A1
  Input variable: pip0A1BO
  Input parameter type: BCG_Pip0A1FailureNotification
  Operation type: One-way

BPEL processes

The PIP sample uses the following processes:

- **PIP3A4BuyerBPEL** - This buyer BPEL process processes the incoming and outgoing messages at the buyer side and logs the appropriate messages in the Systemout.log file. See Figure 4 for an example.
- **IP3A4SellerPBPEL** - This seller BPEL process processes the incoming and outgoing messages at the seller side and logs the appropriate messages in the Systemout.log file. See Figure 5 on page 11 for an example

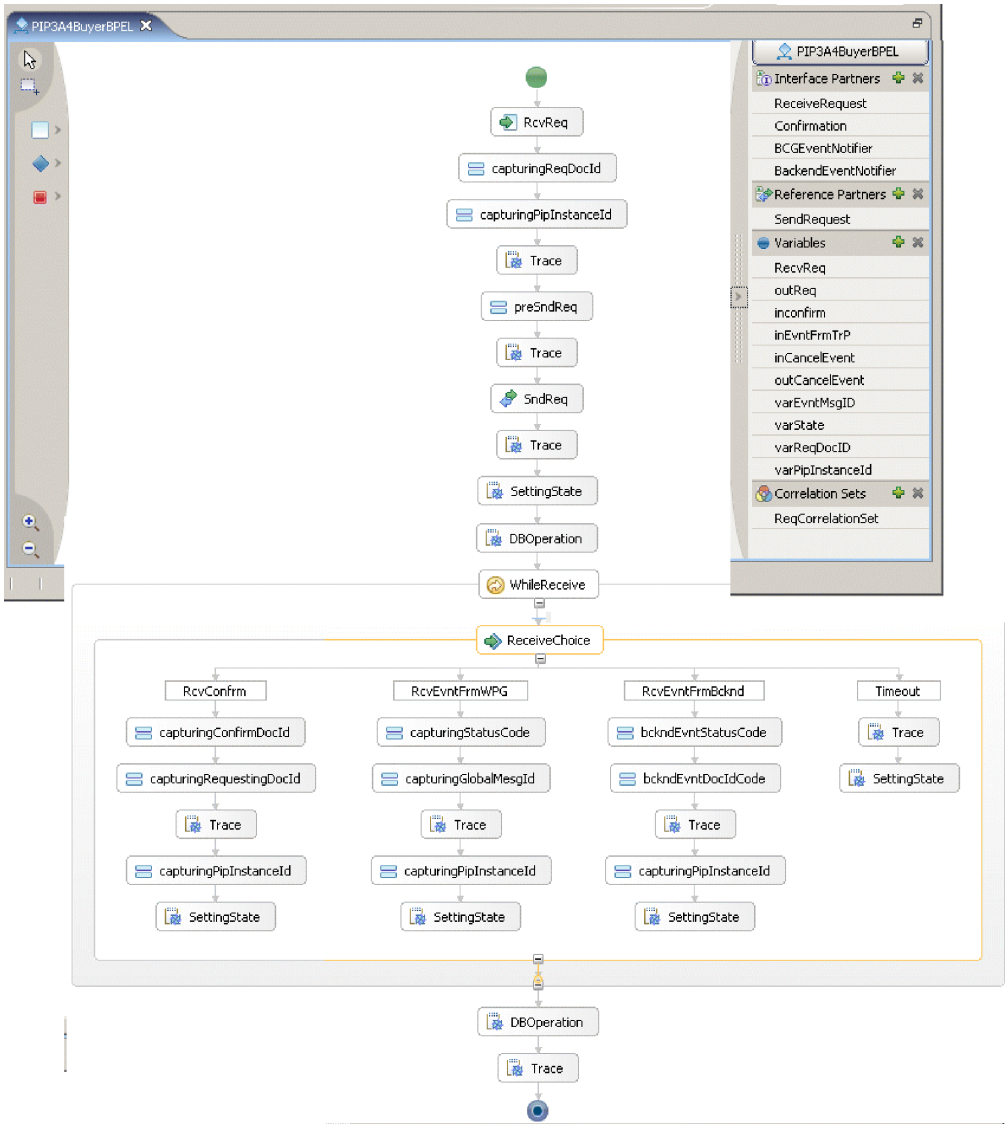

Figure 4. Example of PIP3ABuyerBPEL process

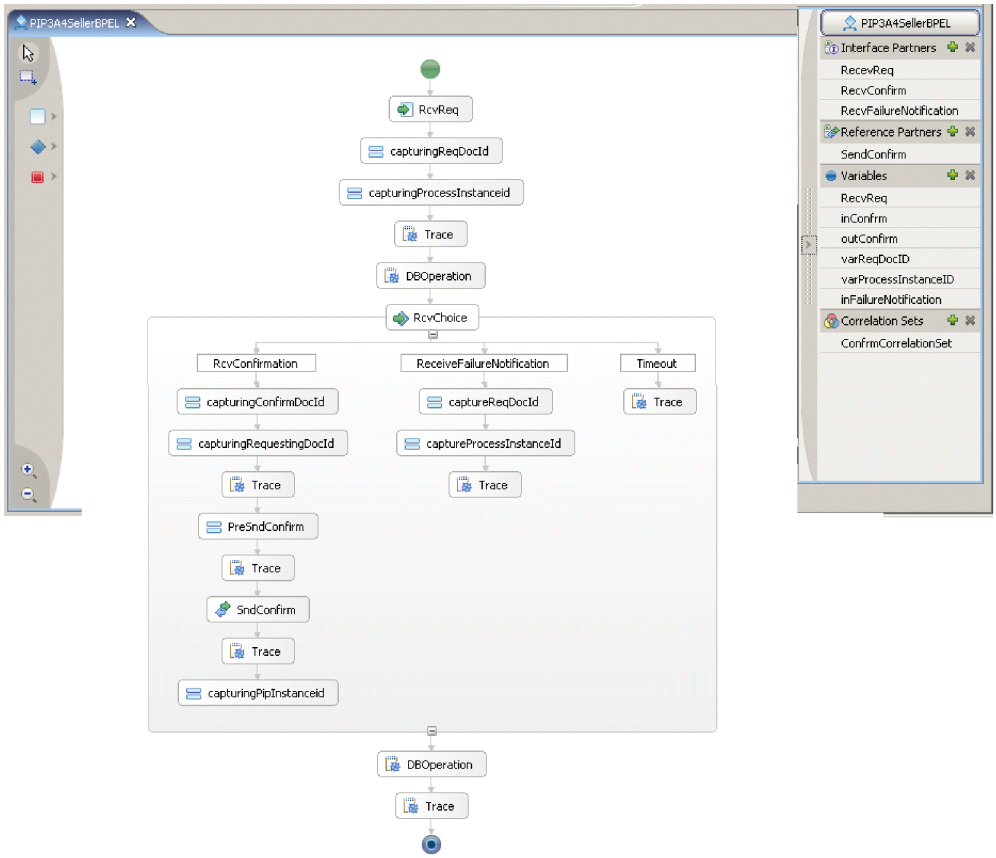

Figure 5. Example of PIP3ASellerBPEL process

Setting up the sample

Setting up the sample involves setting up WebSphere Partner Gateway, WebSphere Admin Console, and WebSphere Process Server. As explained in the section "Topology used by the sample" on page 1, system A is running WebSphere Process Server, System B is running WebSphere Partner Gateway for buyer, and system C is running WebSphere Partner Gateway for seller. Messages in the sample flow in the following manner:

- **Buyer side**: The index.jsp on the buyer side initiates the flow by sending the PIP3A4 request message. This invokes the buyer process running on system A. The buyer process sends request message to WebSphere Partner Gateway on system B over the WPM JMS queue with JNDI name RequestQ/PIP3A4Buyer. WebSphere Partner Gateway on system B sends the response message to the buyer process running on WebSphere Process Server on system A over the WPM JMS queue using the JNDI name ResponseQ/PIP3A4BuyerGW.
  The buyer event dispatcher sends event messages to WebSphere Partner Gateway on system B over the WPM JMS queue with JNDI name EventToBCG/PIP3A4Buyer. WebSphere Partner Gateway on system B sends the event or failure notification messages to the buyer process running on WebSphere Process Server on system A over the WPM JMS queue using the JNDI name EventFromBCG/PIP3A4BuyerGW.
- **Seller side**: The index.jsp on the seller side initiates the flow for the seller side. It invokes the seller process running on system A. The seller process sends the

response message (PIP message) to WebSphere Partner Gateway on system C over the WPM JMS queue using the JNDI name RequestQ/PIP3A4Seller. WebSphere Partner Gateway on system C sends the request message (PIP message) to the seller process running on WebSphere Process Server on system A over the WPM JMS queue ResponseQ/PIP3A4SellerGW.
The seller event dispatcher sends the event messages to WebSphere Partner Gateway on system C over the WPM JMS queue using the JNDI name EventToBCG/PIP3A4Seller. WebSphere Partner Gateway on system C sends the event or failure notification messages to the seller process running on WebSphere Process Server on system A over the WPM JMS queue EventFromBCG/PIP3A4SellerGW.

The following sections describe how to do this.

Setting up WebSphere Partner Gateway

The following procedure describes how to set up WebSphere Partner Gateway so that it has the settings and resources it needs to run the scenarios of the PIP sample. The following steps describe the setup for both System B and System C.

1. Start WebSphere Partner Gateway and log in to the Community Console as Hub Admin.
2. Create a Community Manager profile to represent WebSphere Partner Gateway and a Community Participant profile for the other system, as follows:
   - **System B**: On System B, create a Community Manager profile to represent WebSphere Process Server on the buyer side and a Community Participant profile to represent the Community Participant on System C. Create following business IDs:
     - Community manager: 987654321
     - Community participant: 123456789
   - **System C**: On System C, create a Community Manager profile to represent WebSphere Process Server on the seller side and a Community Participant profile to represent the Community Participant on System B. Create following business IDs:
     - Community manager: 123456789
     - Community participant: 987654321

   For information on creating profiles, see the *Administrator Guide*.
3. Create the gateways for the profiles.

   See the *Administrator Guide* for more information on creating gateways.
   a. Click **Account Admin > Profiles > Community Participant**.
   b. Search for the Community Manager profile you created.
   c. Select the profile and click **Gateways**.
   d. Click **Create**.
   e. Create two gateways on each side, one or events and the other for request or response messages.
   f. In the Gateway Detail section, type or select the values shown in Table 2 on page 13:

Table 2. Community Manager gateway values

Parameter	Value to type or select
Gateway Name	Type the name for the gateway, as follows: System B: PIP message: **PIP3A4BuyerGW** Event messages: **EventGW** System C: PIP message: **PIP3A4SellerGW** Event messages: **EventGW**
Transport	JMS
Target URI	iiop://<System_A_IP_addresss:2089>/ If you have installed WebSphere Partner Gateway using the embedded WebSphere option, you must specify the target URI as follows: File:///<user_defined_MQ_JNDI_bindings_path> For more information, refer to the *Enterprise Integration Guide*.
JMS Factory Name	Type the name of the JMS factory, as follows: System B: **PIP3ABuyer/PIP3A4BuyerQCF** System C: **PIP3A4Seller/PIP3ASellerQCF**
JMS Queue Name	Type the name of the JMS queue, as follows: System B: PIP messages: **ResponseQ/PIP3A4BuyerGW** Event messages: **EventFromBCG/PIP3A4BuyerGW** System C: PIP messages: **ResponseQ/PIP3A4SellerGW** Event messages: **EventFromBCG/PIP3A4SellerGW**
JMS JNDI Factory Name	com.ibm.websphere.naming.WsnInitialContextFactory If you have installed WebSphere Partner Gateway using embedded WebSphere option, please refer to the *Enterprise Integration Guide*.
JMS Message Class	TextMessage

Note: Refer to the *Enterprise Integration Guide* for more information on back-end configuration using JMS transport protocol through WPM.

g. For other parameters, use the default values.

h. Click **Save**.

i. Create the gateway for the community participant in the same way but use the following values for the Gateway Detail section:

Table 3. Community participant gateway values

Parameter	Value to type or select
Gateway name	Type any name for the gateway
Transport	HTTP/1.1

Table 3. Community participant gateway values (continued)

Parameter	Value to type or select
Target URI	Type the URL for the other WebSphere Partner Gateway system, as follows: **System B**: http://<IP_Address_of_System_C:57080>/bcgreceiver/submit/test **System C**: http://<IP_Address_of_System_B:57080>/bcgreceiver/submit/test

For the other parameters, use the default values.

j. Click **Save**.

4. Set the gateways as default gateways:
   a. Click **Account Admin > Profiles > Community Participant**.
   b. Search for the Community Manager profile you created.
   c. Select the profile and click **Gateways**.
   d. In the Gateway List section, click **View Default Gateways**.
   e. For all of the gateway types, select the gateway you created.
   f. Set the default gateways for the Community Participant profile in the same way.
5. Upload the following PIP document flow packages:
   - Package_RNIF_V02.00.zip
   - BCG_Package_RNIFV02.00_3A3V02.02.zip
   - BCG_Package_RNIFV02.00_0A1V02.00.zip
   - BCG_Package_RNSC1.0_RNIFV02.00_3A4V02.02.zip
   - BCG_Package_RNSC1.0_RNIFV02.00_0A1V02.00.zip

   Refer to the "Uploading packages" section in the *Administrator Guide* for information on uploading packages. If packages for the other RNIF version or another version of the PIP have already been loaded, set the Overwrite Data parameter to Yes.

   You can verify that the packages have been uploaded by clicking **Hub Admin > Hub Configuration > Document Flow Definition**. Click **All** and look for the following in the RNIF (V02.00) and Backend Integration packages:
   - Document Flow: 3A4 (V02.02)
   - Document Flow: 0A1 (V02.00)
6. Create interactions for the PIPs:
   a. Click **Hub Admin > Hub Configuration > Document Flow Definition**.
   b. In the Manage Document Flow Definitions window, click **Manage Interactions**.
   c. In the Manage Interactions screen, click **Create Interaction**.
   d. Expand the Document Flow Definition trees by clicking **All** in the Source tree and in the Target tree.
   e. In the Source tree, select the radio button for **Action: Purchase Order Request Action** in the following context:

```
Package: RNIF (V02.00)
    Protocol: RosettaNet (V02.00)
        Document Flow: 3A4 (V02.02) "Request Purchase Order"
            Activity: Request Purchase Order
```

   f. In the target tree, select the radio button for **Action: Purchase Order Request Action** in the following context:

```
Package: Backend Integration (1.0)
    Protocol: RNSC (1.0)
        Document Flow: 3A4 (V02.02) "Request Purchase Order"
            Activity: Request Purchase Order
```

g. In the Action field, select **Bi-directional Translation of RosettaNet and RosettaNet Service Content with Validation**.

h. Click **Save**.

i. Repeat steps a-h to create an interaction in the other direction. That is, the RNIF Package is the target and the Backend Integration package is the source.

j. Repeat steps a-i to create interactions for the following actions:

- 3A4 Purchase Order Confirmation Action.
- 0A1 Failure Notification Action

7. Create an interaction for XMLEvent.

a. Click **Hub Admin > Hub Configuration > Document Flow Definition**.

b. In the Manage Document Flow Definitions window, click **Manage Interactions**.

c. In the Manage Interactions window, click **Create Interaction**.

d. Expand the Document Flow Definition trees by clicking **All** in the Source tree and in the Target tree.

e. In the Source tree, select the radio button for **Document Flow: XMLEvent (1.0)** in the following context:

```
Package: Backend Integration (1.0)
    Protocol: XMLEvent (1.0)
```

f. In the Target tree, select the radio button for **Document Flow: XMLEvent (1.0)** in the following context:

```
Package: Backend Integration (1.0)
    Protocol: XMLEvent (1.0)
```

g. In the Action field, select **Pass Through**.

h. Click **Save**.

8. Create an interaction for XMLEvent to 0A1 RNSC.

a. Click **Hub Admin > Hub Configuration > Document Flow Definition.**

b. In the Manage Document Flow Definitions window, click **Manage Interactions.**

c. In the Valid Document Flow Interactions window, click **Create Interaction.**

d. Expand the Document Flow Definition trees by clicking **All** in the Source tree and in the Target tree.

e. In the Source tree, select the radio button for **Document Flow: XMLEvent (1.0)** in the following context:

```
Package: Backend Integration (1.0)
    Protocol: XMLEvent (1.0)
```

f. In the target tree, select the radio button for **Action: Failure Notification Action** in the following context:

```
Package: Backend Integration (1.0)
    Protocol: RNSC (1.0)
       Document Flow: 0A1 (V02.00) "Notification of Failure"
          Activity: Distribute Notification of Failure
```

g. In the Action field, select **Bi-directional Translation of RosettaNet and xml with validation.**

h. Click **Save.**

9. Create targets for the transport protocols:

a. Click **Hub Admin > Hub configuration > Targets**.
b. Click **Create**.
c. In the Target Name field, type a name.
d. In the Transport field, select HTTP/S.
e. In the Target Configuration section, type the URI for the Receiver that handles HTTP messages such as /bcgreceiver/Receiver.
f. Select the appropriate Gateway Type. (Example: Production).
g. Click **Save**.
h. Click **Hub Admin > Hub configuration > Targets**.
i. Click **Create**.
j. Create two targets on each side, one for PIP messages and another for event messages.
k. In the Target Name field, type a name.
l. In the Transport field, select JMS.
m. In the Target Configuration section, type the appropriate values for the following fields:
   - JMS Provider URL: iiop://<System_A_IP_address:2089>/
     If you have installed WebSphere Partner Gateway using the embedded WebSphere option, you must specify the JMS Provider URL as follows: File:///<user_defined_MQ_JNDI_bindings_path> For more information, refer to the *Enterprise Integration Guide*.
   - JMS Queue Name:
     System B:
     Pip messages: **RequestQ/PIP3A4Buyer**
     Event messages: **EventToBCG/PIP3A4Buyer**
     System C:
     Pip messages: **RequestQ/PIP3A4Seller**
     Event messages: **EventToBCG/PIP3A4Seller**
   - JMS Factory Name:
     System B: PIP3A4Buyer/PIP3A4BuyerQCF
     System C: PIP3A4Seller/PIP3A4SellerQCF
   - JNDI Factory Name:
     com.ibm.websphere.naming.WsnInitialContextFactory

   Note: Refer to the *Enterprise Integration Guide* for more information on back-end configuration using JMS transport protocol through WPM.
n. Select the appropriate Gateway Type. Example: Production
o. Click **Save.**

10. Refer to the *Administrator Guide* for information on enabling security.
11. Enable the B2B capabilities for the profiles.
    a. Click **Account Admin > Profiles > Community Participant**.
    b. Search for the Community Manager profile you created.
    c. Select the profile and click **B2B Capabilities**.
    d. Expand the Document Flow Definition tree by clicking **All**.
    e. Ensure that the Community Manager has the B2B capabilities for the RNIF (V02.00) and Backend Integration (1.0) packages enabled. If the packages are inactive (neither enabled or disabled), activate them by clicking the icon in the Set Source and Set Target columns.

f. Repeat the previous step for the RosettaNet (V02.00) protocol under the RNIF (V02.00) package and the XMLEvent (1.0) and RNSC (1.0) protocols under the Backend Integration (1.0) package. Do the same for the following Document Flows:
   - Document Flow: XMLEvent (1.0) under Protocol: XMLEvent (1.0)
   - Document Flow: 3A4 (V02.02) under Protocol: RNSC (1.0)
   - Document Flow: 0A1 (V02.00) under Protocol: RNSC (1.0)
   - Document Flow: 3A4 (V02.02) under Protocol: RosettaNet (V02.00)
   - Document Flow: 0A1 (V02.00) under Protocol: RosettaNet (V02.00)

g. Repeat a-f for the Community Participant profile.

12. Create participant connections.
    a. Click **Account Admin > Participant Connections**.
    b. In the Source, select the Community Manager profile.
    c. In the Target, select the Community Participant profile.
    d. Click **Search**.
    e. Click the **Activate** button for the following interaction:

Table 4.

Source	Target
Package: Backend Integration (1.0) Protocol: XMLEvent (1.0) Document Flow: XMLEvent (1.0)	Package: Backend Integration (1.0) Protocol: RNSC (1.0) Document Flow: 0A1 (V02.00) Activity: Distribute Notification of Failure (N/A)

    f. After step d, click the **Activate** button for all the other interactions you see on the screen.
    g. In the Source, select the Community Participant profile.
    h. In the Target, select the Community Manager profile.
    i. Click **Search**.
    j. Click the **Activate** for the following interaction:

Table 5.

Source	Target
Package: Backend Integration (1.0) Protocol: XMLEvent (1.0) Document Flow: XMLEvent (1.0)	Package: Backend Integration (1.0) Protocol: XMLEvent (1.0) Document Flow: XMLEvent (1.0)

    k. After step d, click the **Activate** button for all the other interactions you seen on the screen.
    l. Select EventGW of community manager for all XML Event and PIP0A1 interactions.

Setting up WebSphere Application Server Admin Console

The following procedure describes how to create JMS resources required on System A to run all the scenarios of the PIP sample.

1. Log on to WebSphere Application Server Admin Console for System A.

2. Create the following System integration buses:
   - PIP3A4Buyer
   - PIP3A4Seller

   For more information on creating system integration buses, refer to the *Enterprise Integration Guide*.
3. Create the following queues:

   The following queues are required on PIP3A4Buyer service integration bus:

Table 6. Required queues on PIP3A4Buyer service integration bus

Queue name	JNDI name
RequestQ	RequestQ/PIP3A4Buyer
ResponseQ	ResponseQ/PIP3A4BuyerGW
EventForBCG	EventFromBCG/PIP3A4BuyerGW
EventToBCG	EventToBCG/PIP3A4Buyer

   The following queues are required on PIP3A4Seller service integration bus:

Table 7. Required queues on PIP3A4Seller service integration bus

Queue name	JNDI name
RequestQ	RequestQ/PIP3A4Seller
ResponseQ	ResponseQ/PIP3A4SellerGW
EventForBCG	EventFromBCG/PIP3A4SellerGW
EventToBCG	EventToBCG/PIP3A4Seller

   For more information on creating queue destinations on system integration buses, refer to the *Enterprise Integration Guide*.
4. Deploy the generated .ear files.
   For information on generating .ear files, refer "Generating .ear files for PIP processes" on page 19.
   For information on deploying generated .ear files, refer to the WebSphere Application Server documentation.
5. Create two databases with embedded cloudscape with the names **PIP3A4Buyer** and **PIP3A4Seller**. For each of the databases, following these steps:
   a. Using Windows Explorer, navigate to the following directory:`<Websphere process server installed directory>\runtimes\bi_v6\cloudscape\bin\embedded>`
   b. Execute **cview.bat**.
   c. Create two databases with the names **PIP3A4Buyer** and **PIP3A4Seller**.
   d. Create the PROCStatus table with the following structure in each of the databases. The buyer process uses the PROCStatus table in PIP3A4Buyer database, and the seller process uses PROCStatus table in PIP3A4Seller database.

Table 8. Structure for PROCStatus table

Column	Type	Description
`PIPINSTANCEID`	`VARCHAR(80)`	PIP instance identifier. Uniquely identifies a PIP process.
`REQDOCID`	`VARCHAR(80)`	Requesting document identifier
`PROCNAME`	`VARCHAR(20)`	Name of the process

Table 8. Structure for PROCStatus table (continued)

Column	Type	Description
STATUS	VARCHAR(10)	It can take two values: START: if given process instance is still running. STOP: if given process instance is not running.

e. Close the cloudscape databases.

Important:
If you forget to close the databases, your business processes will not be able to access the databases.

f. Configure the JDBC provider information using the WAS admin console, as follows:

1) Select resources > jdbc provider.
2) Click the cloudscape jdbc provider (xa).
3) If one does not exist, create it.
4) Add datasource to this provider.
5) Make sure the JNDI names are **jdbc/PIP3A4Buyer** for the buyer database and **jdbc/PIP3A4Seller** for the seller database.

Generating .ear files for PIP processes

The following procedure describes how to set up WebSphere Process Server so that it has the settings and resources it needs to run all the scenarios of the PIP sample.

1. Open WebSphere Integration Developer, and provide a new name for Workspace, for example: **PIP3A4**.
2. Right-click ″Business integration explorer,″ and select Import from the menu.
3. Select Project Interchange from the Wizard SELECT page.
4. Provide the BCG_PIP3A4Sample.zip file location in the ″From zip file text″ field.
5. Click the Select All and Finish buttons.
6. Clean your project by selecting Project > Clean project. Ensure that the ″Start a build immediatly″ option is selected.
7. Export PIP3A4Buyer as BCG_PIP3A4BuyerApp.ear and PIP3A4Seller as BCG_PIP3A4SellerApp.ear, then close WebSphere Integration Developer.
8. Start the WebSphere Process Server, and deploy the .ear files using the WebSphere Application Server Admin Console.
9. Start the following applications: PIP3A4BuyerApp and PIP3A4SellerApp.

Note: At the time of cleaning and building your project, you may get the following error in the default.xsd file. Ignore this type of error:

```
The element may not have duplicate name and target namespace.
```

Running Scenario 1

The following steps describe how to run Scenario 1.

1. Send PIP3A4 purchase order request to Buyer Process. The following steps describe how to do this

a. Open a Web browser, and go to the following URL: http://<System_A_IP_address>:9080/PIP3A4BuyerWeb/index.jsp.
b. Locate the BCG_3A4PurchaseOrderRequest_V02.02.xml message by clicking the Browse button, then click the ″Create Business Object″ button.

c. Send the business object in asynchronous mode by clicking the Send Request button. The PIP3A4BuyerBPEL process is invoked with the business object sent from the index.jsp.

2. Open the Systemout.log file (located in <WPS installation>\profiles\ProcSrv01\logs\server1), then search for the following text:

```
*** Buyer Side ::Received Request with doc id($$$$$)from Backend ***
*** Buyer Side ::Assigned Received Request with doc id($$$$$)to
outbound port ***
*** Buyer Side ::Sent Request with doc id($$$$$) to Trading partner ***
```

Important: The symbol $$$$$ refers to the dynamic values that are being sent from or received by the partner.

This text indicates that the 3A4 request has been successfully posted to the JMS JNDI queue (RequestQ/PIP3A4Buyer).

3. Use the Document Viewer of the Console to verify that the WebSphere Partner Gateway instance on the buyer's side has received the 3A4 request and sent it to the WebSphere Partner Gateway instance configured as the seller's gateway. Do this for both System B and System C. The following steps describe how to do this:

   a. Open your favorite browser.

   b. Type the URL of the console in the Address text field.
   System B: http://<Address of System B>:<WebSphere Partner Gateway console port of System B>/console
   System C: http://<Address of System C>:<WebSphere Partner Gateway console port of System C>/console

   c. Enter the user name, password and Company Login Name

   d. Click the "Viewers" tab.

   e. Click the "Document viewer" tab.

   f. Click Search. The display shows all the documents that are sent or received by the system.

   g. To get details about the documents received and sent from System C, follow steps 3a through 3f replacing System B with System C.

4. The seller's WebSphere Partner Gateway instance (System C) receives the 3A4 Request RNIF message and converts it to a RNSC message. It then sends the message (3A4 Request RNSC as backend integration packaging message over JMS) to the seller process (PIP3A4SellerBPEL) running on WebSphere Process Server (System A). The sellers' WebSphere Partner Gateway instance on System C also sends RNIF acknowledgement to the buyers's WebSphere Partner Gateway instance on System B.

5. The buyer's WebSphere Partner Gateway instance (System B) receives this RNIF acknowledgment from the seller's WebSphere Partner Gateway instance on System C. It then converts it to an XML event with status code 100, and sends this XML event (as backend integration packaging message over JMS) to the buyer's Event Dispatcher(BuyerEventDispatcher) running on WebSphere Process Server (System A).

   - (Buyer side) The BuyerEventDispatcher receives the XML event from the buyer's WebSphere Partner Gateway instance (System B). This is the acknowledgment for the 3A4 request which was sent by BuyerBPELProcess earlier. To verify that BuyerBPELProcess received the XML event, open the Systemout.log file (located in <WPS installation>\profiles\ProcSrv01\logs\server1) in a text editor, and search for the following text:

```
*** Buyer Side ::Received Event with status code($$$$$) from Trading
partner for the Request with doc id($$$$$) ***
```

Important: The symbol $$$$$ refers to the dynamic values that are being sent from or received by the partner.

- (Seller side) The SellerBPELProcess receives the 3A4 request message from seller's WebSphere Partner Gateway instance (System C). This is the 3A4 request which was sent by the buyer process earlier. To verify that SellerBPELProcess received the 3A4 request message, open the Systemout.log file (located in <WPS installation>\profiles\ProcSrv01\logs\server1) in a text editor, and search for the following text:

  ```
  *** Seller Process :: Request with doc id($$$$$) Received from Trading
  Partner***
  ```

  Important: The symbol $$$$$ refers to the dynamic values that are being sent from or received by the partner.

6. Send PIP 3A4 purchase order confirmation to Seller Process. The following steps describe how to do this:
   a. Open another instance of your browser, and go to the following URL: http://<System_A_IP_address>:9080/PIP3A4SellerWeb/index.jsp.
   b. Locate the BCG_3A4PurchaseOrderConfirmation_V02.02.xml message by clicking the Browse button.
   c. Click the Create Business Object button.
   d. Provide the Requesting document identifier and x-aux-process-instance-id values, as follows:
      1) Provide the value of the Requesting Document Identifier in the **RequestingDocumentIdentifier** text field. This value is the same as the value of the x-aux-system-msg-id header of the received request message. You can obtain this value from the index.jsp file of the buyer.
      2) Provide the value of the process instance identifier value in the **x-aux-process-instance-id** text field. This value is the same as the value of the x-aux-process-instance-id header of the received request message. You can obtain this value from the index.jsp file of the seller.
7. Send the business object in asynchronous mode by clicking the Send Request button. The PIP3A4SellerBPEL process is invoked with the business object sent from the index.jsp.
8. The PIP3A4SellerBPEL receives a 3A4 purchase order confirmation message. It then sends this message to the JMS queue configured in WebSphere Partner Gateway (System C) JMS gateway. To verify 3 that the A4 purchase order confirmation message was successful sent, open the Systemout.log file (located in <WPS installation>\profiles\ProcSrv01\logs\server1) in a text editor, and search for the following text:

   ```
   *** Seller Process :: Confirmation with DocId($$$$$)Received from
   Backend for the request with DocID($$$$$) ***
   ```

   ```
   *** Seller Process :: Received Confirmation with DocID($$$$$) assigned
   to outbound port for the request with DocId($$$$$) ***
   ```

   ```
   *** Seller Process :: Confirmation with DocID($$$$$) Sent to Trading
   partner for the request with DocId($$$$$) ***
   ```

   Important: The symbol $$$$$ refers to the dynamic values that are being sent from or received by the partner.

The seller's WebSphere Partner Gateway (System C) receives a 3A4 confirmation message. It then sends the 3A4 confirmation RNIF message to the buyer's WebSphere Partner Gateway (System B). The buyer's WebSphere Partner Gateway (System B) receives the RNIF message and converts it to RNSC. It then sends a 3A4 RNSC (as backend integration packaging message over JMS) to the JMS queue configured in WebSphere Partner Gateway (System B) JMS gateway.

9. The BuyerBPELProcess receives the 3A4 confirmation message from the JMS queue. This is the 3A4 confirmation at the buyer's backend process. To verify that the BuyerBPELProcess received the 3A4 confirmation message, open the Systemout.log file (located in <WPS installation>\profiles\ProcSrv01\logs\server1) in a text editor, and search for the following text:

   ```
   *** Buyer Side ::Received Confirmation from Trading partner ***
   ```

Running Scenario 2

The following steps describe how to run Scenario 2.

1. Select the Initiated request ID from the drop-down list from the Scenario 2 section at the buyer side.
2. Send the event by clicking the Send Event button.
3. Open the Systemout.log file (located in <WPS installation>\profiles\ProcSrv01\logs\server1) in a text editor, then search for the following text:

   ```
   ##############################################
   Associated Process for the Event having PIPInstanceID ==> $$$$$ is not
   active
   ##############################################
   ```

   Important: The symbol $$$$$ refers to the dynamic values that are being sent from or received by the partner.

   This indicates that the XML Event has been successfully posted to the EventToBCG/PIP3A4Seller queue.
4. The buyer's WebSphere Partner Gateway instance receives the event message. The instance sends a PIP 0A1 message to the seller's gateway.
5. The seller's WEbSphere Partner Gateway instance receives the PIP 0A1 message and converts it into a PIP 0A1 RNSC message. The seller's WebSphere Partner Gateway instance sends this 0A1 RNSC message to the backend.
6. The SellerEventDispatcher receives the PIP 0A1 RNSC message. This is the cancellation event at the seller's backend process.
7. Open the Systemout.log file (located in <WPS installation>\profiles\ProcSrv01\logs\server1) in a text editor, and search for the following text:

   ```
   ##############################################
   Associated Process for the PIP0A1 having failed PIPInstanceID ==> $$$$
   is not active
   ##############################################
   ```

   Important: The symbol $$$$$ refers to the dynamic values that are being sent from or received by the partner.

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Burlingame Laboratory Director
IBM Burlingame Laboratory
577 Airport Blvd., Suite 800

Burlingame, CA 94010
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not necessarily tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

WebSphere Partner Gateway contains code named ICU4J which is licensed to you by IBM under the terms of the International Program License Agreement, subject to its Excluded Components terms. However, IBM is required to provide the following language to you as a notice:

COPYRIGHT AND PERMISSION NOTICE

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.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program. General-use programming interfaces allow you to write application software that obtain the services of this program's tools. However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

i5/OS
IBM
the IBM logo
AIX
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
IMS
Informix
iSeries
Lotus
Lotus Notes
MQIntegrator

MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

WebSphere Partner Gateway Enterprise and Advanced Editions, version 6.0.0.1

Printed in USA