IBM WebSphere Business Integration Connect
Enterprise and Advanced Editions

PIP Sample

Version 4.2.1

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices and Trademarks," on page 33.

20February2004

This edition applies to Version 4, Release 2, Modification 1 of IBM® WebSphere® Business Integration Connect Enterprise Edition (5724-E87) and Advanced Edition (5724-E75), and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send to the following address:

IBM Burlingame Laboratory
Information Development
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A

Include the title and order number of this book, and the page number or topic related to your comment.

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 Business Integration Connect.

Business Integration Connect PIP sample

Business Integration Connect provides the PIP sample to demonstrate how to set up Business Integration Connect and WebSphere Interchange Server 4.2.1 to exchange messages when you implement WebSphere Interchange Server as a backend application. Additionally, you can see how Business Integration Connect behaves when sending and receiving messages from a Community Participant and how it handles attachments.

The PIP sample supports three scenarios. The first scenario demonstrates how Business Integration Connect handles a two action PIP. The second scenario is a continuation of the first scenario in which the PIP is cancelled. The third scenario demonstrates how Business Integration Connect handles a one action PIP that has an attachment.

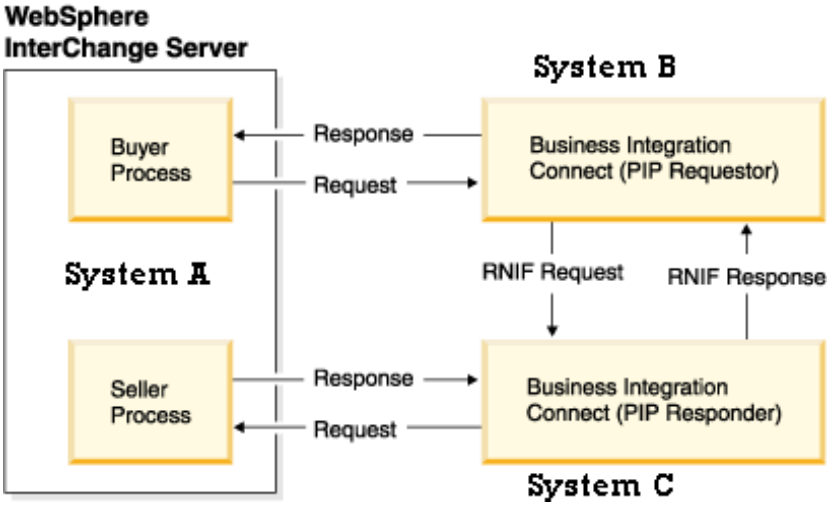
For information additional information on integrating WebSphere Interchange Server, see the Integration Overview.

Topology used by the sample

All of the scenarios use the same topology. System A has WebSphere Interchange Server 4.2.1 and performs the roles of backend application and Community Participant. One process, the buyer process, initiates the PIPs and another process, the seller process, receives the PIPs.

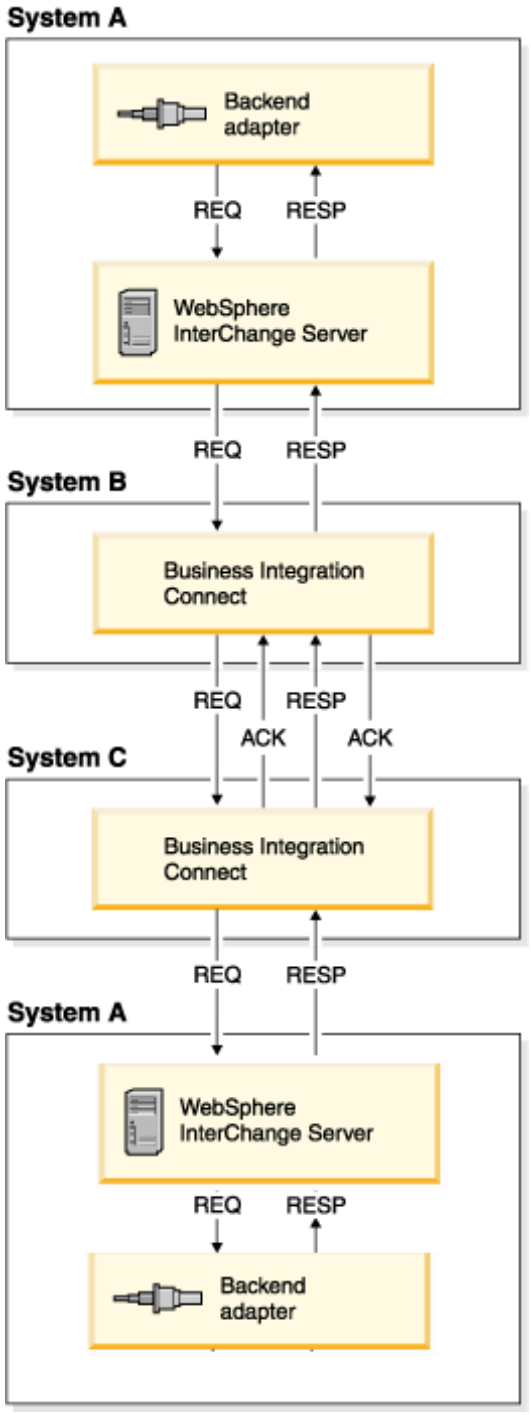
System B has Business Integration Connect Enterprise Edition performing the role of PIP requester. This system receives the PIP content from the buyer process in System A and sends the PIP message to System C. System C has Business Integration Connect Enterprise Edition performing the role of PIP responder.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

The following diagram shows the topology:



Scenario 1: Processing a two action PIP

Scenario 1 demonstrates how Business Integration Connect processes a two action PIP as a sender and receiver. The following diagram shows the flow of PIP or PIP content messages between the systems in the scenario.



The scenario starts with the buyer process in WebSphere Interchange Server receiving a 3A4 request business object from the port connected to the backend adapter. 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 these IDs along with the status of the request message.

The buyer process sends the request on JMS to the Business Integration Connect instance configured as the buyer's gateway, which is the instance on System B. This instance generates a 3A4 PIP for RNIF 2.0 and sends it to the seller's gateway. This gateway is the Business Integration Connect instance on System C. The seller's Business Integration Connect receives the RNIF request message, validates it and sends an acknowledgment signal to the buyer's gateway. The buyer's Business Integration Connect sends an EventNotification message with a statusCode of 100 to the buyer process on the WebSphere Interchange Server. The buyer process updates the status of the PIP transaction.

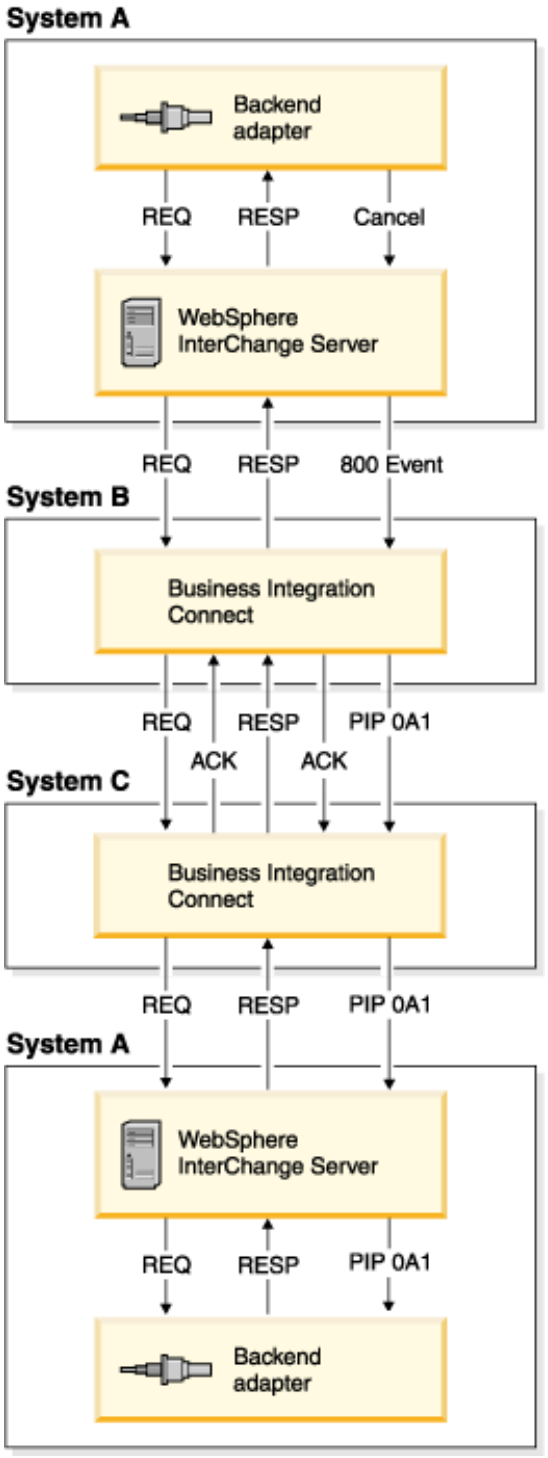
The seller's Business Integration Connect packages the 3A4 content in Backend Integration packaging and sends this message to the seller process running in WebSphere Interchange Server. The seller process saves the IDs contained in the Backend Integration packaging and sends the request to the port connected to the backend adapter.

The backend adapter asynchronously sends the PIP 3A4 response message to the seller process. The seller process retrieves the unique IDs (x_aux_process_instance_id and x_aux_system_msg_id) from the database and populates the response message with them. The seller sends the response message to the seller's Business Integration Connect, which packages the response in RNIF format and sends it to the buyer's gateway.

The buyer's Business Integration Connect validates the response and sends an acknowledgment back to the seller's gateway, which in turn sends an Event Notification message with a statusCode of 100 to the seller process in the WebSphere Interchange Server. The buyer's Business Integration Connect asynchronously sends the PIP 3A4 response message to the buyer process in WebSphere Interchange Server. The buyer process updates the status of the PIP transaction and sends the response message to the port that connects to the backend adapter.

Scenario 2: Processing a 0A1PIP

Scenario 2 is a continuation of Scenario 1. The following diagram shows the messaging of the first scenario and the messaging used to cancel the PIP, which is Scenario 2.



After the buyer process sends the response to the port that connects to the backend adapter, it receives a cancellation event from the backend of the buyer process. The buyer process updates the status of the PIP transaction to 800. The buyer process then populates an Event Notification message with the following information:

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 in the Documentid column in the database table used to store message metadata
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 process then sends the event notification message to its gateway. The buyer's Business Integration Connect instance receives the event notification message and generates an 0A1 PIP based on the message. The instance sends the 0A1 PIP to the seller's gateway. The seller's Business Integration Connect instance receives the PIP 0A1 message and sends it to the seller to process.

Scenario 3: Processing a PIP with attachments

Scenario 3 demonstrates how Business Integration Connect processes a PIP that has attachments. When a request message arrives from the backend adapter of the buyer process, it may contain attachments. Each attachment can be a file that WebSphere Interchange Server passes through or an XML representation of a WebSphere Interchange Server business object. This scenario provides an example of each type of attachment. For the file attachment, WebSphere Interchange Server reads the file and base64 encodes it. WebSphere Interchange Server then stores the encoded attachment content as the defaultAttachment as shown in line 5.1.1 in the following figure:

General | Attributes

	Pos	Name	Type	Key	Foreign	Requi	Card	Maximu	Default
1	1	XMLDeclaration	String	☐	☐	☐		255	xml
2	2	DocType	String	☑	☐	☐		255	DOCTYPE
3	3	⊞ ROOT	BCG_ROOT_Pip3A4PurchaseOrderRequest	☐	☐	☐	1		
4	4	⊞ contentTypeEncoding	BCG_ContentTypeEncoding	☐	☐	☐	1		
5	5	⊟ attachments	BCG_AttachmentContainer	☐	☐	☐	1		
5.1	5.1	⊟ defaultAttachment	BCG_Default_Attachment	☑	☐	☐	N		
5.1.	5.1.1	attachment	String	☑	☐	☐		255	
5.1.	5.1.2	⊞ contentTypeEncoding	BCG_ContentTypeEncoding	☐	☐	☐	1		
5.1.	5.1.3	ObjectEventId	String						
5.2	5.2	⊟ attachmentOne	BCG_Person	☐	☐	☐	N		
5.2.	5.2.1	FirstName	String	☑	☐	☑		255	
5.2.	5.2.2	LastName	String	☐	☐	☑		255	
5.2.	5.2.3	AddressOne	String	☐	☐	☑		255	
5.2.	5.2.4	AddressTwo	String	☐	☐	☑		255	
5.2.	5.2.5	City	String	☐	☐	☑		255	
5.2.	5.2.6	State	String	☐	☐	☑		255	
5.2.	5.2.7	Country	String	☐	☐	☑		255	
5.2.	5.2.8	ZipCode	String	☐	☐	☑		255	
5.2.	5.2.9	ObjectEventId	String						
5.3	5.3	ObjectEventId	String						

For the business object attachment, WebSphere Interchange Server stores it unchanged as attachmentOne in the above figure.

The buyer process sends the request along with the attachments to the buyer's gateway. The buyer's Business Integration Connect instance generates the PIP (including attachments) and sends it to the seller's gateway. The seller's Business Integration Connect instance sends the request and its attachments to the seller's WebSphere Interchange Server. On the seller's WebSphere Interchange Server, the attachment data handler converts the BCG_Persons attachment to a business object.

The seller process retrieves the defaultAttachment, decodes the attachment file, and saves it in the attachment directory. While constructing the message to the backend adapter, the seller process replaces the attachment content with the path to the file in the attachment directory. The seller process does not make any changes to the BCG_Persons business object. The seller then sends a request message to the port connecting to the backend adapter. The backend application uses the file path to retrieve the attachment file when needed.

WebSphere Interchange Server artifacts

The PIP sample uses the business objects, collaboration templates, and the persistence schema listed in this section to support the scenarios.

Business objects

The PIP sample uses the following business objects:

- **BCG_Pip3A4PurchaseOrderRequest** - The 3A4 PIP request message for Scenario 1 and 2. For the sample, it is modified using the BO designer to include JMSDynMO. The payload part was generated using XML ODA and the DTD for the PIP.
- **BCG_Pip3A4PurchaseOrderResponse** - The 3A4 PIP response message for Scenario 1 and 2. For the sample, it is modified using the BO designer to include JMSDynMO. The payload part was generated using XML ODA and the DTD for the PIP.
- **BCG_Pip3C3InvoiceNotification** - The 3C3 PIP request message for Scenario 3. For the sample, it is modified using the BO designer to include JMSDynMO. The payload part was generated using XML ODA and the DTD for the PIP.
- **BCG_Pip0A1FailureNotification** - The 0A1 PIP Notification message for Scenario 2. For the sample, it is modified using the BO designer to include JMSDynMO. The payload part was generated using XML ODA and the DTD for the PIP.
- **BCG_EventNotification** - The event notification messages sent between WebSphere Interchange Server and Business Integration Connect. It includes JMSDynMO but does not contain any attachments.
- **BCG_AttachmentContainer** - The top-level business object for attachments. It can contain child business objects that WebSphere Interchange Server data handler can convert into attachments.
- **JMS_DynMO** - The dynamic meta-object for JMS. See the Integration Overview for information on this business object.

Collaboration templates

The PIP sample uses the following collaboration templates:

- **BCG_PIP3A4_Request** - Represents the buyer process. One end connects to the buyer's backend system and the other end to Business Integration Connect through WebSphere BI adapters.

The collaboration template has the following ports:

BCG_PIP3A4_Request ports

PortName	BusinessObject	Description
RequestFromBackend	BCG_Pip3A4PurchaseOrderRequest	Receives the request from the backend
RequestToWBIC	BCG_Pip3A4PurchaseOrderRequest	Sends the request to Business Integration Connect
EventFromBackend	BCG_EventNotification	Receives the PIP cancellation request from the backend
EventToWBIC	BCG_EventNotification	Sends the PIP cancellation event to Business Integration Connect
ResponseFromWBIC	BCG_Pip3A4PurchaseOrderConfirmation	Receives the response from Business Integration Connect
ResponseToBackend	BCG_Pip3A4PurchaseOrderConfirmation	Sends the response to the backend

Configuration Properties:

1. DB_CONN_POOL_NAME
   Default Value = CWLDPool

2. ATTACHMENT_FILE_DIR
   Default Value = C:\temp

- **BCG_PIP3A4_Response** - Represents the seller process. One end connects to the seller's backend system and the other end to Business Integration Connect through WebSphere BI adapters.

The collaboration template has the following ports:

BCG_PIP3A4_Response ports

PortName	BusinessObject	Description
RequestFromWBIC	BCG_Pip3A4PurchaseOrderRequest	Receives the request from Business Integration Connect
RequestToBackend	BCG_Pip3A4PurchaseOrderRequest	Sends the request to the backend
EventFromBackend	BCG_EventNotification	Receives the PIP cancellation event from the backend
EventFromWBIC	BCG_EventNotification	Receives the PIP acknowledgment event from Business Integration Connect
ResponseFromBackend	BCG_Pip3A4PurchaseOrderConfirmation	Receives the response from the backend
ResponseToWBIC	BCG_Pip3A4PurchaseOrderConfirmation	Sends the response to Business Integration Connect

Configuration Properties

1. DB_CONN_POOL_NAME
   Default Value = CWLDPool1

2. ATTACHMENT_FILE_DIR
   Default Value = C:\temp

- **BCG_PIP3C3_Notifier** - Represents the notifier process. One end connects to the notifier's backend system and the other end to Business Integration Connect through WebSphere BI adapters.

The collaboration template has the following ports:

BCG_PIP3C3_Notifier ports

PortName	BusinessObject	Description
RequestFromBackend	BCG_Pip3C3InvoiceNotification	Receives the notification from the backend
RequestToWBIC	BCG_Pip3C3InvoiceNotification	Sends the notification to Business Integration Connect
EventToWBIC	BCG_EventNotification	Receives the PIP acknowledgment event from Business Integration Connect
EventFromBackEnd	BCG_EventNotification	Receives the PIP cancellation event from the backend

Configuration Properties

1. DB_CONN_POOL_NAME
   Default Value = CWLDPool
2. ATTACHMENT_FILE_DIR
   Default Value = C:\temp

- **BCG_PIP3C3_Receiver** - Represents the receiver process. One end connects to the receiver's backend system and the other end to Business Integration Connect through WebSphere BI adapters.

The collaboration template has the following ports:

BCG_PIP3C3_Receiver ports

PortName	BusinessObject	Description
RequestFromWBIC	BCG_Pip3C3InvoiceNotification	Receives the request from Business Integration Connect
RequestToBackend	BCG_Pip3C3InvoiceNotification	Sends the request to the backend
EventFromBackEnd	BCG_EventNotification	Receives the PIP cancellation event from the backend
EventFromWBIC	BCG_EventNotification	Receives the PIP acknowledgement event from Business Integration Connect

Configuration Properties

1. DB_CONN_POOL_NAME
   Default Value = CWLDPool1
2. ATTACHMENT_FILE_DIR
   Default Value = C:\temp

- **BCG_0A1FailureNotification** - Represents the FailureNotification process. One end connects to the receiver's backend system and the other end to Business Integration Connect through WebSphere BI adapters.

The collaboration template has the following ports:

BCG_0A1FailureNotification

PortName	BusinessObject	Description
NOFFromWBIC	BCG_Pip0A1FailureNotification	Receives the PIP 0A1 from Business Integration Connect
NOFToBackend	BCG_Pip0A1FailureNotification	Sends the PIP 0A1 to the backend
EventFromWBIC	BCG_EventNotification	Receives the PIP acknowledgement event from Business Integration Connect
EventToBackEnd	BCG_EventNotification	Sends the Acknowledgment event to the backend

Configuration Properties

DB_CONN_POOL_NAME
Default Value = CWLDPool

PIP persistence schema

The PIP sample uses a database table to persist the IDs and status of the PIP messages. The following table describes the schema of the database table:

PIP message persistence schema

Column	Corresponding JMSProperties attribute	Description
Senderid	x-aux-sender-id	The ID of the initiator of the PIP
Receiverid	x-aux-receiver-id	The ID of the destination of the PIP
Documentid	x-aux-msg-id	The ID of the message assigned by the process initiating the PIP
PIPInstanceid	x-aux-process-instance-id	The ID of the PIP to which the message belongs
Msgid	x-aux-system-msg-id	The ID of the message assigned by the system sending the PIP message
Status	x_aux_event_status_code	The state of the PIP: 0 – Initiated 100 – Ack received 800 – PIP terminated by backend 900 – Exception received

PIP message persistence schema

DocType	None	The type of message: REQ – for request message RESP – for response message NOTI – for notification message 0A1 – for Failure notification message

Setting up the sample

Setting up the sample involves setting up Business Integration Connect, WebSphere MQ, and WebSphere Interchange Server. The following sections describe how to do this.

Setting up Business Integration Connect

The following procedure describes how to set up Business Integration Connect so that it has the settings and resources it needs to run all the scenarios of the PIP sample. The setup for System B and C is the same except where noted.

1. Start Business Integration Connect and log in to the Community Console as Hub Admin.
2. Create a Community Manager profile to represent Business Integration Connect and a Community Participant profile for the other system. For example, on System B, you create a Community Participant profile for System C. For information on creating profiles, see the Administrator's Guide.
3. Create the gateways for the profiles:

   See the Administrator's 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. In the Gateway List section, click **Create**.

   e. In the Gateway Detail section, type or select the following values:

      Use the default values for all other parameters.

Community Manager gateway values

Parameter	Value to type or select
Gateway name	Type any name for the gateway
Transport	JMS
Target URI	Type the context URL such as file:///export/jndi/myctx
JMS Factory Name	Type the name of the JMS factory Example: `myqcf`

Community Manager gateway values

JMS Queue Name	Type the name of the JMS queue Example: `SENDERQ`
JMS JNDI Factory Name	Type the name of the JNDI factory for JMS Example: `com.sun.jndi.fscontext.RefFSContextFactory`
JMS Message Class	Type the JMS Message Class Example: TextMessage

NOTE: Refer to the Integration Overview guide for more information on backend configuration using JMS with WebSphere MQ 5.3.

f. Click **Save**.

g. Create the gateway for the Community Participant in the same way but use the following values for the Gateway Detail section:

Community Participant gateway values

Parameter	**Value to type or select**
Gateway name	Type any name for the gateway
Transport	HTTP/1.1
Target URI	Type the URL for the other Business Integration Connect system. That is, if you are creating System B, type the URL for System C. Example: http://<IPAddress of System C:57080>/bcgreceiver/submit/test

For the other parameters, use the default values.

h. 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
   - Package_RNSC_1.0_RNIF_V02.00.zip
   - BCG_Package_RNIFV02.00_3A4V02.02.zip
   - BCG_Package_RNIFV02.00_3C3V01.01.zip
   - BCG_Package_RNIFV02.00_0A1V02.00.zip

- BCG_Package_RNSC1.0_RNIFV02.00_3A4V02.02.zip
- BCG_Package_RNSC1.0_RNIFV02.00_3C3V01.01.zip
- BCG_Package_RNSC1.0_RNIFV02.00_0A1V02.00.zip

See "Uploading packages" in the Administrator's 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: 3C3 (V01.01)
- 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 screen, 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.
        - 3C3 Invoice Notification 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 screen, 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 **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 screen, click **Manage Interactions.**
    c. In the Valid Document Flow 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 **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. In the Target Name field, type a name.

k. In the Transport field, select JMS.

l. In the Target Configuration section, type the appropriate values for the following fields:

- JMS Provider URL
  Example: file:///export/jndi/myctx
- JMS Queue Name
  Example: RECEIVERQ
- JMS Factory Name
  Example: myqcf
- JNDI Factory Name
  Example: com.sun.jndi.fscontext.RefFSContextFactory

NOTE: Refer to the Integration Overview guide for more information on backend configuration using JMS with WebSphere MQ 5.3.

m. Select the appropriate Gateway Type. (Example: Production).

n. Click **Save.**

10. Refer to the Administrator's 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), active 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: 3C3 (V01.01) 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: 3C3 (V01.01) 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 **Activate** for the following interaction:

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. Enable all other interactions 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 **Activate** for the following interaction:

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. Enable all other interactions on the screen.

Setting up WebSphere MQ

To set WebSphere MQ to support the sample, create the following queues in the Queue Manager:

- CWLD_Unsubscribed
- CWLD_InProgress
- CWLD_Result
- CWLD_Error
- The input queue that the JMS connector polls for incoming messages.
- The output queue that the JMS connector uses for outbound messages.

Consult the WebSphere MQ documentation for information on how to create the queues.

Setting up the WebSphere Interchange Server

The following procedure describes how to set up WebSphere Interchange Server so that it has the settings and resources it needs to run all the scenarios of the PIP sample. For more information on any of the steps in the procedure, see the WebSphere Interchange Server documentation.

1. Once the WebSphere Interchange Server is running, start the system manager and create an integration component library (ICL).
2. Import the contents of the ICS Repository into the newly created ICL.
3. Create the Database Connection Pool at Initiator side:
    a. Create a database for Requestor and create the RNState table using the BCG_pip_sample_table_creation.sql script.
    b. In the ICL, right click the Database Connection Pool folder and select **Create new Database Connection Pool**.
    c. Specify the database, database driver (DB2), DBConnection name, login, password, and maximum number of connections.
    d. In the new connection pool section, right click and select **New Connection Pool**.
    e. Specify the name of the pool as CWLDPool and set the minimum number of connections to 1.
    f. Click **OK** and then click **Finish** to create the Database Connection Pool.
4. Create the Database Connection Pool at Responder side:
    a. Create second database for Responder and create the RNState table using the BCG_pip_sample_table_creation.sql script.
    b. In the ICL, right click the Database Connection Pool folder and select **Create new Database Connection Pool**.
    c. Specify the database, database driver (DB2), DBConnection name, login, password, and maximum number of connections.

d. In the new connection pool section, right click and select **New Connection Pool**.

e. Specify the name of the pool as CWLDPool1 and set the minimum number of connections to 1.

f. Click **OK** and then click **Finish** to create the Database Connection Pool

5. Create the Connectors:

a. In the ICL, right click the Connectors folder and select **Create new connector**.

b. The Connector Configurator window appears. In the New Connector panel, select **Cancel**.

c. Select **File > Open > From File**.

d. In the File Open dialog, select the connector configuration file for the JMS Connector and click **Open**.

e. Repeat a-d to create the Port Connector.

f. Open the JMS Connector and select **File > Save As > To Project**. Save a copy of the JMS Connector using the name JMSConnector1.

6. Configuring the JMS Connectors

a. In the ICL, open the Connectors folder and double-click the JMS Connector.

b. In the Connector Configurator window, select the **Connector-Specific Properties** tab.

c. Set values for the following attributes:

Attribute	**Value**
CTX_InitialContextFactory	com.sun.jndi.fscontext.RefFSContextFactory
ReplyToQueue	The name of the queue in which the JMS Connector puts the messages
UnsubscribedQueue	CWLD_Unsubscribed
CTX_ProviderURL	The URL of the JMS context provider
InProgressQueue	CWLD_InProgress
DataHandlerConfigMO	MO_DataHandler_Default
MessageResponseResultProperty	CWLD_Result
DataHandlerMimeType	Attachment
QueueConnectionFactoryName	The queue connection factory created in WebSphere MQ
ErrorQueue	CWLD_Error
InputQueue	The name of the queue that the JMS Connector polls for incoming messages

d. Select the **Supported Business Objects** tab.

e. Select the following Business Objects from the list and enable Agent Support for each one:

- BCG_Pip3A4PurchaseOrderRequest
- BCG_Pip3A4PurchaseOrderConfirmation
- BCG_Pip3C3InvoiceNotification
- BCG_Pip0A1FailureNotification
- BCG_EventNotification
- MO_DataHandler_Default

f. Select the **Trace/Log Files** tab and configure the log and trace files.

g. Repeat a-f for JMSConnector1

7. In the MO_DataHandler_Default business object, add the Attachment attribute and set its BO Type as MO_DataHandler_DefaultAttachmentConfig.
8. Configure the Port Connector:
   a. In the ICL, open the Connectors folder and double-click the Port Connector.
   b. In the Connector Configurator window, select the **Supported Business Objects** tab.
   c. Select the following Business Objects from the list and enable Agent Support for each one:
      - BCG_Pip3A4PurchaseOrderRequest
      - BCG_Pip3A4PurchaseOrderConfirmation
      - BCG_Pip0A1FailureNotification
      - BCG_Pip3C3InvoiceNotification
      - BCG_EventNotification
   d. Select the **Trace/Log Files** tab and configure the log and trace files.
9. Compile the collaboration templates by right clicking the Collaboration Templates folder and selecting **Compile All**.
10. Create the collaboration objects
    a. Right-click the Collaboration Objects folder and select **Create new collaboration object**.
    b. Select the template and provide a name for the collaboration object name. Click **Next**.
    c. Bind the ports with the appropriate connectors given in the following table:

Port-connector bindings for BCG_Pip3A4_Request collaborations

Port	Connector
RequestFromBackend	PortConnector

Port-connector bindings for BCG_Pip3A4_Request collaborations

RequestToWBIC	JMSConnector
EventToWBIC	JMSConnector
EventFromBackend	PortConnector
ResponseFromWBIC	JMSConnector
ResponseToBackend	PortConnector

d. Click **Next**.

e. Specify the e-mail notification address, set the system trace level to 2, and set the collaboration trace level to 5. Click **Next**.

f. Specify values for the following collaboration properties, using the default values where available:

- DB_CONN_POOL_NAME
- ATTACHMENT_FILE_DIR

g. Click 'Finish' to complete creation of collaboration object.

h. Repeat a-g for the following collaborations

Port-connector bindings for BCG_Pip3A4_Response collaborations

Port	**Connector**
RequestFromWBIC	JMSConnector1
RequestToBackend	PortConnector
EventFromBackend	PortConnector
EventFromWBIC	JMSConnector1
ResponseFromBackend	PortConnector
ResponseToWBIC	JMSConnector1

Port-connector bindings for BCG_Pip3C3_Notifier collaborations

Port	**Connector**
RequestFromBackend	PortConnector
RequestToWBIC	JMSConnector
EventToWBIC	JMSConnector
EventFromBackend	PortConnector

Port-connector bindings for BCG_Pip3C3_Receiver collaborations

Port	**Connector**

Port-connector bindings for BCG_Pip3C3_Receiver collaborations

RequestFromWBIC	JMSConnector1
RequestToBackend	PortConnector
EventFromBackend	PortConnector
EventFromWBIC	JMSConnector1

Port-connector bindings for BCG_0A1FailureNotification collaboration for Requestor

Port	**Connector**
NOFFromWBIC	JMSConnector
NOFToBackend	PortConnector
EventFromWBIC	JMSConnector
EventToBackend	PortConnector

Port-connector bindings for BCG_0A1FailureNotification collaboration for Responder

Port	**Connector**
NOFFromWBIC	JMSConnector1
NOFToBackend	PortConnector
EventFromWBIC	JMSConnector1
EventToBackend	PortConnector

NOTE: Two 0A1 Failure Notification collaboration objects are needed; one for the initiator side and another for the responder side.
Initiator side 0A1 collaboration: One end connects to the buyer's backend system and the other end connects to Business Integration Connect through WebSphere BI adapters, i.e. JMSConnector.
Receiver/Responder side 0A1 collaboration: One end connects to the receiver's backend system and the other end connects to Business Integration Connect through WebSphere BI adapters, i.e. JMSConnector1. Change the value of the property `DB_CONN_POOL_NAME` to `CWLDPool1`.

11. Deploy the ICL into the ICS Repository
    a. Open the User Projects folder.
    b. Right-click Interchange Server Projects and select **New User Project**.
    c. Type a name for the project.
    d. In the Available Integration Component Libraries drop down list, select the ICL you created.
    e. Click **Finish**. This associates the project with the ICL.

f. In InterChange Servers, connect to the ICS Server.

g. Right-click the server name and select **Add User Project**. Select the project you created.

h. Right-click the server name and select **Deploy Projects**.

i. Select the entire project and click **Next**.

j. Click **Next**.

k. Select the folder for the project and click **Finish**. This deploys the project on the server.

12. Restart the Interchange Server.
13. In a DB2 command window, run the DB creation scripts (db2RNtable_create.sql).
14. Verify that the scripts created the RNState table.
15. Start the JMS adapters. Ensure that the JMS queues are properly configured. Consult the adapter configuration guide for details.
16. Start Monitor.exe and verify that the collaborations and connectors are active.

Running Scenario 1

To run Scenario 1, do the following:

1. Start the VT connector and define a profile for the Port Connector. Select **File > Connect Agent** to begin simulating the agent.
2. Load the sample 3A4 request object (PIP3A4Request.bo) and update the following fields in the test BO:

   - x_aux_sender_id under JMSDynMO-> JMSProperties (This should be the Business ID of the sender. For example, Community Manager as configured in System B WBI-C Partner Profile.)
   - x_aux_receiver_id under JMSDynMO-> JMSProperties (This should be the Business ID of the receiver. For example, Community Participant as configured in System B WBI-C Partner Profile.)
   - thisDocumentIdentifier under ROOT (This should be a unique identifier)
   - OutputQueue under JMSDynMO (This should be the queue configured to send the messages to SystemB WBI-C. See **Hub Admin** > **Hub configuration** > **Targets**, **JMS Target** on System B WBI-C)

     Please refer to the Integration Overview guide for more information about the properties under JMSDynMO.

3. Send it in asynchronous mode.
4. Open the log viewer and load the Interchange Server trace file. Search for the following text:

   ```
   Collaboration Success: Collaboration Name {The collaboration
   name}, Scenario Name SendRequest, BLOCK Name SendBO.
   ```

   This indicates that the 3A4 request has been successfully posted to the JMS connector.

5. Verify that the Business Integration Connect on the buyer's side has received the 3A4 request and sent it to the Business Integration Connect instance configured as the seller's gateway.
6. The Port Connector receives the 3A4 request message. This is the 3A4 request at the seller's backend process. Select the request and click **Reply success**.
7. Load the 3A4 response object (PIP3A4Response.bo) and update the following fields in the test BO:
   - x_aux_sender_id under JMSDynMO-> JMSProperties (This should be the Business ID of the sender as is configured in System C WBI-C Partner Profile.)
   - x_aux_receiver_id under JMSDynMO-> JMSProperties (This should be the Business ID of the receiver, i.e, Community Participant as configured in System C WBI-C Partner Profile.)
   - requestingDocumentIdentifier under ROOT (This should be the same as 'thisDocumentIdentifier' for the Request received)
   - thisDocumentIdentifier under ROOT (This should be a unique identifier)
   - OutputQueue under JMSDynMO (This should be the queue configured to send the messages to SystemC WBI-C. See **Hub Admin** > **Hub configuration** > **Targets**, **JMS Target** on System C WBI-C)

     Please refer to the Integration Overview guide for more information about the properties under JMSDynMO.
8. Send it in asynchronous mode.
9. Open the log viewer and load the Interchange Server trace file. Search for the following text:

   ```
   Collaboration Success: Collaboration Name {The collaboration
   name}, Scenario Name SendResponse, BLOCK Name SendBO.
   ```

   The Port Connector receives the 3A4 response message is received in the Port Connector. This is the 3A4 response at the buyer's backend process. Select the response and click **Reply success**.

Running Scenario 2

To run Scenario 2, run Scenario 1 and then do the following:

1. Load the PIP3A4Cancel.bo and update the following fields in the test BO:
   - x_aux_sender_id under JMSDynMO-> JMSProperties (This should be the Business ID of the sender , as is configured in SystemB WBI-C Partner Profile.)
   - x_aux_receiver_id under JMSDynMO-> JMSProperties (This should be the Business ID of the receiver, i.e, Community Participant as configured in SystemB WBI-C Partner Profile.)
   - EventMessageID under ROOT (This should be a unique identifier)
   - BusinessObjectID under ROOT (This should be the same as 'thisDocumentIdentifier' of the Request sent)

- GlobalMessageID under ROOT (This should be the same as ‘x_aux_msg_id’ of the Request sent)
- x_aux_process_instance_id under JMSDynMO-> JMSProperties (This should be the same as ‘x_aux_process_instance_id ‘of the Request under JMSDynMO-> JMSProperties sent)
- StatusCode under ROOT and x_aux_event_status_code under JMSDynMO-> JMSProperties e.g 800
- OutputQueue under JMSDynMO (This should be the queue configured to send the messages to SystemB WBI-C. See **Hub Admin** > **Hub configuration** > **Targets, JMS Target** on System B WBI-C)

  Please refer to the Integration Overview guide for more information about the properties under JMSDynMO.

2. Send it in asynchronous mode.
3. Open the log viewer and load the Interchange Server trace file. Search for the following text:

   ```
   Collaboration Success: Collaboration Name {The collaboration name}, Scenario Name SendEvent, BLOCK Name SendEvent.
   ```

   This indicates the event message has been successfully posted to the JMS connector.
4. The buyer's Business Integration Connect instance receives the event message. The instance sends a PIP 0A1 message to the seller’s gateway.
5. The Port Connector receives the PIPA1 message.

Running Scenario 3

To run Scenario 3, do the following:

1. Start the VT connector and define a profile for the Port Connector. Select **File > Connect Agent** to begin simulating the agent.

1. Load the sample 3C3 request object (PIP3C3Request.bo) with the URI of the attachment or with a default attachment BO or both. Update the following fields in the test BO:
   - x_aux_sender_id under JMSDynMO-> JMSProperties (This should be the Business ID of the sender as is configured in System B WBI-C Partner Profile.)
   - x_aux_receiver_id under JMSDynMO-> JMSProperties (This should be the Business ID of the receiver, i.e, Community Participant as configured in System B WBI-C Partner Profile.)
   - thisDocumentIdentifier under ROOT (This should be a unique identifier)
   - OutputQueue under JMSDynMO (This should be the queue configured to send the messages to SystemB WBI-C. See **Hub Admin** > **Hub configuration** > **Targets, JMS Target** on System B WBI-C)

     Please refer to the Integration Overview guide for more information about the properties under JMSDynMO.
2. Send it in asynchronous mode.

3. Open the log viewer and load the Interchange Server trace file. Search for the following text:

   ```
   Collaboration Success: Collaboration Name {The collaboration
   name}, Scenario Name SendRequest, BLOCK Name SendBO.
   ```

   This indicates the 3C3 request has been successfully posted to the JMS connector.

4. Verify that the Business Integration Connect instance on the buyer's side has received the 3C3 request and sent it to the trading partner (in this case the Business Integration Connect instance configured as the seller's gateway).

5. The 3C3 request message must reach the responder's gateway and the seller process. The collaboration at the seller side decodes the attachment and writes it to a file mentioned in the collaboration properties, and this URI is set in the business object and sent to back end.

6. The Port Connector receives the 3C3 request message. This is the 3C3 request at the seller’s backend process. Select the request and click **Reply success**.

Notices and Trademarks

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 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 contains 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 contains 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. 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 IBM's application programming interfaces.

Programming interface information

Programming interface information 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 of International Business Machines Corporation in the United States, other countries, or both:

IBM
the IBM logo
CrossWorlds
DB2
DB2 Universal Database
MQSeries
Tivoli
WebSphere

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

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

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

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