WebSphere® Partner Agreement Manager

Business Process Integration Adapter Guide

Version 2.1

WebSphere® Partner Agreement Manager

Business Process Integration Adapter Guide

Version 2.1

Note
Before using this information and the products it supports, read the information in "Notices" on page 49

First Edition (June 2001)

This edition applies to Version 2.1 of the IBM® WebSphere® Partner Agreement Manager (program number 5724-A85) and to all subsequent release and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this book

This book provides a guide to the configuration and use of the Business Process Integration (BPI) Adapter. The BPI Adapter is an MQSeries Adapter Offering (MQAO) adapter that runs at the Partner Agreement Manager. It provides a means for other MQAO adapters to business applications to interact with public processes running on Partner Agreement Manager.

Who should read this book

The audience for this book includes anyone who needs to use the BPI Adapter to allow public processes to interact with backend business processes and applications. Users of the BPI Adapter should use this book to install, configure, and implement the BPI Adapter in a variety of integration scenarios.

What you need to know

You should be familiar with the basic concepts of Partner Agreement Manager, and with the environment in which the BPI Adapter is being used, for example, IBM WebSphere Business Integrator, or WebSphere Commerce Suite.

You need to understand the Partner Agreement Manager terms: *extension actions, events*, and *business objects*, which are essential to understanding the adapters. Reading the books: *Partner Agreement Manager User's Guide, Partner Agreement Manager Administrator's Guide,* and the *Partner Agreement Manager Installation Guide* will provide you with the necessary background.

An understanding of MQAO is also desirable, particularly the concepts of *source adapter* and *target adapter*. See "Bibliography" on page 53 for a list of MQAO documentation.

Conventions and terminology used in this book

The Partner Agreement Manager publications contain a glossary that you should refer to for definitions of relevant terminology.

Other useful glossaries

The Web Site at http://www.ibm.com/ibm/terminology consolidates several of the main glossaries created for IBM products in one convenient location, including:

- Glossary of Computing Terms

- DB2® Glossary
- Tivoli® Glossary

How to send your comments

IBM welcomes your comments. You can send your comments by any one of the following methods:

1. Electronically to this address:

   `idrcf@hursley.ibm.com`

   Be sure to include your network address if you want a reply.
2. By FAX, to the following numbers:

   `UK: 01962-842327`
   `Other countries: +44-1962-842327`
3. By mail to the following address:

   User Technologies
   Mail Point 095
   IBM United Kingdom Laboratories
   Hursley Park
   Winchester
   Hampshire
   SO21 2JN
   United Kingdom

Chapter 1. Overview

This chapter introduces the Business Process Integration (BPI) Adapter by describing:

- What the BPI Adapter is and in which environments you can use it
- How the BPI Adapter allows other MQAO adapters to interact with Partner Agreement Manager public processes
- How the BPI Adapter can be invoked as a private process
- The subcomponents that comprise the BPI Adapter

The BPI Adapter is an MQSeries® Adapter Offering (MQAO) adapter that runs at the Partner Agreement Manager. It allows public processes running on Partner Agreement Manager to interact, via other MQAO adapters, with back-end business processes and applications, which can be:

- The Business Flow Manager within a WebSphere Business Integrator environment
- Business applications using MQAO adapters
- Other products such as WebSphere Commerce Suite that need Partner Agreement Manager functionality.

Partner Agreement Manager is a business-to-business product that allows trading partner companies to develop and execute joint business processes over the Internet. These processes operate on two levels:

1. The *public process*, which determines the flow of messages, or information, between the partner companies.
2. The *private processes*, which for each company, are the implementation of its steps in the shared process. These are internal to the company concerned.

Figure 1 illustrates a typical public process. The process has three process steps and involves two partners; one partner taking the role of a Buyer, the other a Seller. Between each step in the public process a *business object (BO)* both transfers control of the process and defines the information that is transmitted between the partners. The process above is initiated by the buying company, perhaps as the result of an inventory check. The Buyer builds a purchase order containing the item(s) that make up the order, Partner Agreement Manager transmits this to the Seller. The Seller is now in control of the process and responds to the request with an order acceptance business object that is returned by Partner Agreement Manager to the Buyer.

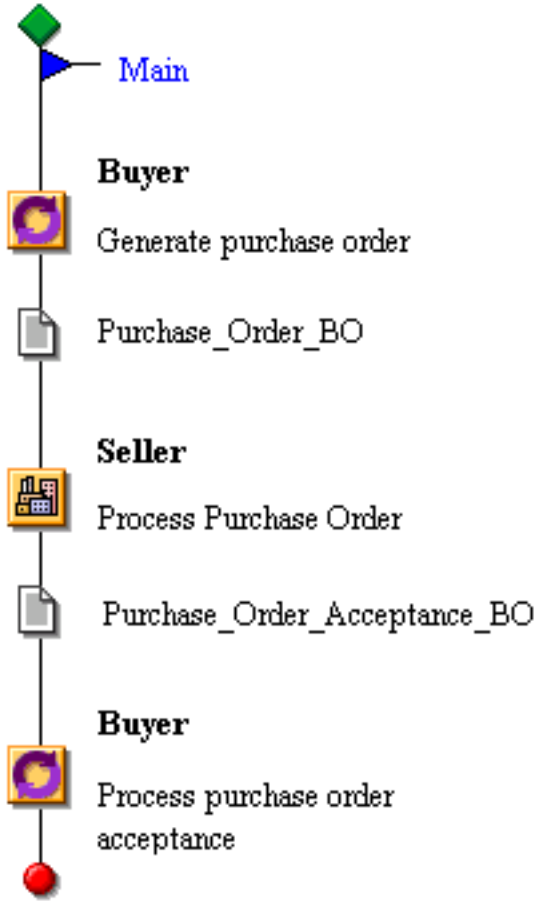

Figure 1. A typical public process

The public process defines the external behavior of the process. Each partner typically needs to invoke its own business systems to actually implement its steps in the process. This is done within private processes, which if back-end integration is required, need to invoke the services of an adapter to interact with the business systems.

The BPI Adapter is one such adapter. It provides connectivity between other MQAO adapters and the public processes being executed by the local partner. Typically these adapters are associated with business applications that need to be invoked to process the request originating from the partner site. Business objects received from partner sites are passed to the BPI Adapter by Partner Agreement Manager and then routed to the appropriate MQAO target adapter for processing. Similarly, business objects can be created by an MQAO source adapter and then sent to the BPI Adapter, which receives them and passes them onto Partner Agreement Manager for sending to a partner, see Figure 2.

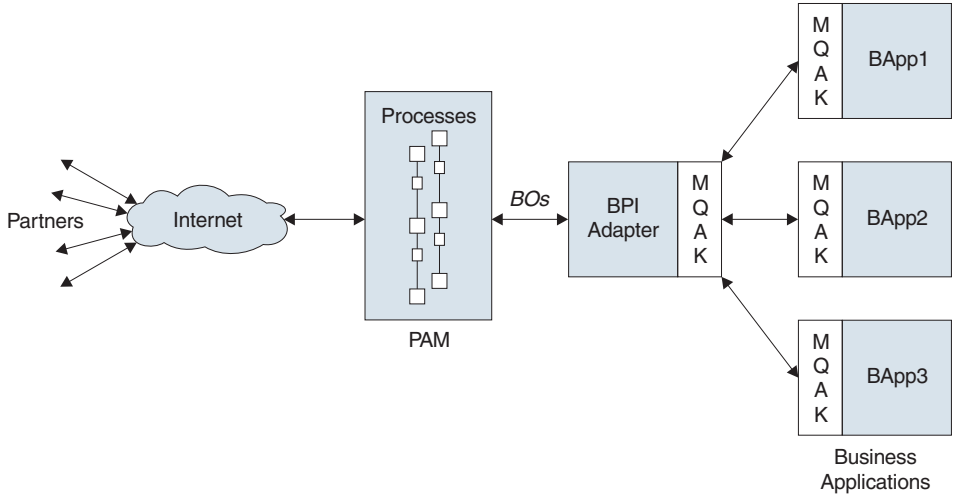

Figure 2. Partner Agreement Manager, BPI Adapter, and MQAO adapters

Figure 2 illustrates the role of MQSeries Adapter Kernel (MQAK), which enables the deployment and execution of adapters, and which provides related functions such as the routing of messages. For more information, refer to the *MQSeries Adapter Kernel for Multiplatforms: Quick Beginnings* book.

If we consider the purchase order process again, a source MQAO adapter could be developed for the inventory system of the Buyer. When stock needs to be reordered, the source adapter is invoked by the inventory system, which produces the purchase order BO and sends it to the BPI Adapter. The BPI Adapter initiates a new public process in Partner Agreement Manager with the Seller. The business object is sent over the Internet to the Seller, which responds some time later with its acceptance business object. Partner

Agreement Manager then calls the BPI Adapter again passing it the business object, which is routed to the appropriate business application for processing.

Note: The BPI Adapter facilitates both synchronous and asynchronous public process invocations. In this example, instead of the acceptance business object being routed to a separate adapter, it could be returned as a reply to the waiting source adapter that initiated the public process

.

Private processes

The BPI Adapter, like other Partner Agreement Manager Adapters, can be invoked in a private process step. Typically, private processes must be developed manually using the Partner Agreement Manager Process Editor, however, within a WebSphere Business Integrator environment, generation of private processes is performed automatically when the public process is deployed to Partner Agreement Manager. For more information, refer to *WebSphere Studio Business Integrator Extensions Developer's Guide*.

Private processes that use the BPI Adapter must send and/or receive business objects to and from other MQAO adapters.

Consider the role of the Seller in the example public process, see Figure 1 on page 2. The purchase order business object must be sent by the BPI Adapter to another MQAO adapter for processing. After processing of the order is complete, the same, or a different, MQAO adapter must then return the acceptance business object to the BPI Adapter. The private process that the Seller must implement calls the BPI Adapter to send the business object and then calls it again to receive the reply business object. Partner Agreement Manager Adapters are invoked within private processes as *Extension actions* and the private process would consist of seven separate steps and have the structure shown in Figure 3 on page 5.

Figure 3. Private process for Seller

Notes on private process:

1. **Script to set BO details.**

   This is a script that is used to access the business object to be sent and context information such as the identity of the partner that sent it.

2. **Invoke BPI Adapter to send BO.**

   This extension action invokes the BPI Adapter to send the business object to the MQAO adapter. The context information determined in step 1 is passed as inputs to the operation.

3. **Invoke BPI Adapter to complete send of BO.**

   For integrity, both the sending and receiving of business objects need to be performed by two separate invocations of the BPI Adapter. This completes the send performed in step 2.

4. **Invoke BPI Adapter to receive BO.**

   This step calls the BPI Adapter to receive the business object that is to be output by the local partner in the public process step. Typically, Partner Agreement Manager must retry this operation a number of times before the business object is available.

5. **Invoke BPI Adapter to complete receive of BO.**

   After the business object has been received, this operation is called to complete its receive.

6. **Script to convert string to output BO.**

   The business object is returned to the private process as a string. This script uses the string to create the business object to be returned to the public process.

7. **Output received BO**

   This final step returns the received business object to the public process. After the private process has completed, Partner Agreement Manager sends the business object to the remote partner.

The BPI Adapter instance that is called during execution of the above private process resides in a separate component of Partner Agreement Manager called the Adapter Server. This is responsible for controlling private process steps that interact with adapters.

BPI Adapter architecture

The BPI Adapter consists of three separate sub-components (see also Figure 4).

1. An **Interaction Service** that provides the adapter operations to send and receive business objects.
2. An **Initiation Service** that is concerned only with starting new public processes in Partner Agreement Manager, and
3. A **Process Completion Service** that provides an optional service to the back-end to inform them when a public process completes normally or abnormally.

Both the Initiation Service and the Process Completion Service run as separate threads within the BPI Adapter. They communicate directly with the Partner Agreement Manager Process Server using the Partner Agreement Manager external API.

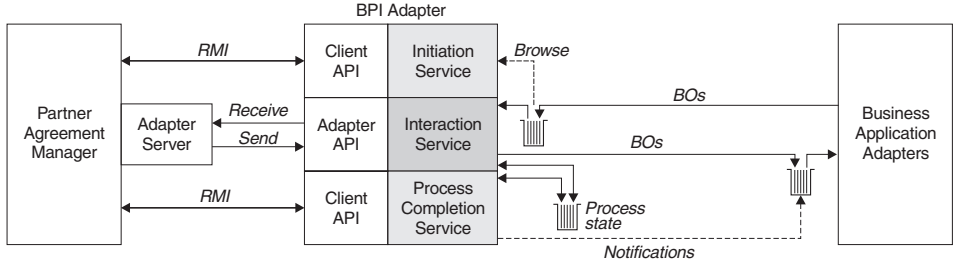

Figure 4. Partner Agreement Manager and the BPI Adapter

Messages containing business objects that must initiate a new public process within Partner Agreement Manager carry a special architected TransportCorrelationId of "GWNONE" that allows them to be targeted by the Initiation Service from the same receiver queue being used by the BPI Adapter. Such messages are browsed (that is, not removed) by the Initiation Service, which uses fields set within the message to determine the public process that is to be started, and if the process has been defined with a partner group, with which partner. The Initiation Service starts the processes and sets an input variant in the public process to contain the message identifier of the initiating message. The private process step for the first partner action passes this to the BPI Adapter, which uses it to retrieve the message from the queue. The BPI Adapter uses an internal queue to store persistent state information for all active public processes with which it is involved.

The association of the Partner Agreement Manager public process instance identifier with that of the corresponding back-end business process is logged by the BPI Adapter. In a WebSphere Business Integrator environment, this allows the Partner Agreement Manager audit log information for the public

process to be federated with that of the rest of the solution. To support its own operation the BPI Adapter also stores this association persistently for the duration of each public process.

The BPI Adapter manages the necessary correlation information to bind public processes in the gateway with business processes in a business application such as the Business Flow Manager of Business Integrator.

Chapter 2. Installation

This chapter lists software prerequisites and describes how to install the BPI Adapter.

Prerequisite software

The following software must be installed on the machine that is hosting the Adapter Server. Note that this may be a different machine than that hosting the main Partner Agreement Manager Process Server.

- MQSeries Version 5.2
- MQSeries Support for Java and Java Message Service, SupportPak MA88.
- MQSeries Adapter Kernel Version 1.2

See "Chapter 3. Configuration" on page 11 for information about configuration of this software.

Installation

Within the Business Integrator environment, the BPI Adapter is installed automatically as part of the wrapper installation.

Refer to the *WebSphere Business Integrator Installation Guide* for more information.

Chapter 3. Configuration

This chapter describes the configuration that you must do after you have installed the software prerequisites and the BPI Adapter. You must perform configuration in the following order:

1. MQSeries configuration
2. MQAK configuration
3. BPI Adapter configuration

Note

In the WebSphere Business Integrator environment, the configuration described in the following sections is done as part of the wrapper installation, and by running the impgwadp.bat file. Refer to the *WebSphere Business Integrator Installation Guide* for more information.

MQSeries configuration

The BPI Adapter requires a local queue manager to be created on the machine hosting the Partner Agreement Manager Adapter Server. The name of this queue manager is also defined as a property of the BPI Adapter, see "Adapter properties" on page 21.

When the BPI Adapter is installed within a WebSphere Business Integrator environment, this queue manager and its corresponding adapter property are both automatically configured. When you are installing the BPI Adapter in other environments, create a queue manager with the following default name:

```
PAM.QUEUE.MANAGER
```

Actually a queue manager with a different name can be created as long as the corresponding BPI Adapter property (**InternalQueueManager**) is set to the same name.

The BPI Adapter uses two local queues on this queue manager, (see Figure 4 on page 7). One is its public queue where it receives messages from other MQAO adapters. The other is a private queue that is used by the BPI Adapter to store state information about the public processes it is servicing. When the BPI Adapter is being installed manually, you should define these queues with the following default names:

```
SYSTEM.PAM.RECEIVER.QUEUE
SYSTEM.PAM.INTERNAL.STATE.QUEUE
```

You can choose a different queue name for the internal state queue as long as the corresponding BPI Adapter property (**InternalQueue**) is set to the same name.

The BPI Adapter accesses both of these queues using two logical MQAK application identifiers that must be set up as adapter properties (**MQAOApplicationID, MQAOInternalApplicationID**) and also defined to the MQAK. The internal state queue is accessed directly using the MQSeries Java interface, which is why both the queue manager and internal queue names must be defined as BPI Adapter properties.

This section only outlines the minimum configuration needed to make the BPI Adapter startable by the Partner Agreement Manager Adapter Server. In a real system, MQSeries connectivity to other queue managers would be required so that MQAO adapters running on other machines could send and receive messages to and from the BPI Adapter.

MQAK configuration

The MQSeries Adapter Kernel provides a logical messaging infrastructure to MQAO adapters. Adapters are assigned logical application identifiers and messages are statically routed by the MQAK to other adapters based upon the type of message that is being sent. The configuration required to achieve this is defined separately, an MQAO adapter need only concern itself with producing and/or consuming messages.

The configuration of MQAK is based on the Lightweight Directory Access Protocol (LDAP), therefore, configuration information can be defined in a LDAP directory structure, or in an XML file with a structure that mirrors LDAP.

In a WebSphere Business Integrator environment, all configuration information is defined within a LDAP directory structure, but for simplicity, the examples in this book assume that an XML configuration file is being used. The top-level XML element, <Epic>, represents the top level of the LDAP directory, and subordinate LDAP objects are represented by XML elements nested within the top-level element, see Figure 5 on page 13.

Each application and adapter has its own logical identifier and entry in the configuration data. The BPI Adapter receives messages from its receiver queue and sends or receives messages to and from its internal queue. It therefore requires two application identifiers, though one is internal to it, no other MQAO adapter apart from the BPI Adapter itself sends messages there. The default names for the two application identifiers are:

```
PAM
PAMInternal
```

If different application identifiers are used, the corresponding BPI Adapter properties (**MQAOApplicationID, MQAOInternalApplicationID**) need to be changed from these default values, see "Adapter properties" on page 21.

The following example shows entries within the MQAK configuration file for these two applications. This is the minimum configuration required to start the BPI Adapter, further entries are required for the MQAO adapters that need to communicate with Partner Agreement Manager. See "Chapter 5. Integration scenarios" on page 27 for a fuller example of MQAK configuration required to implement the anticipated integration patterns.

```
<Epic o="ePIC">
  <ePICApplications o="ePICApplications">
    <ePICApplication epicappid="PAM">
      <AdapterRouting cn="epicadapterrouting">
        <epicmqppqueuemgr>PAM.QUEUE.MANAGER</epicmqppqueuemgr>
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <epicreceivemqppqueue>SYSTEM.PAM.RECEIVER.QUEUE</epicreceivemqppqueue>
            <epicreceivemode>MQPP</epicreceivemode>
          </ePICBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
    <ePICApplication epicappid="PAMInternal">
      <AdapterRouting cn="epicadapterrouting">
        <epicmqppqueuemgr>PAM.QUEUE.MANAGER</epicmqppqueuemgr>
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <epicreceivemqppqueue>SYSTEM.PAM.INTERNAL.STATE.QUEUE</epicreceivemqppqueue>
            <epicreceivemode>MQPP</epicreceivemode>
          </ePICBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
  </ePICApplications>
</Epic>
```

Figure 5. Example of MQAK configuration

When an XML file is being used for configuration, the entries for the PAM and PAMInternal applications would be added to a file called aqmconfig.xml. When an LDAP directory structure is being used, the LDAP editor could be used to add equivalent objects/contexts into the o=ePIC schema. This minimum configuration defines the two application identifiers and the means for the BPI Adapter to send or receive messages to and from these queues.

You should note the following important points:

epicappid="PAM", epicappid="PAMInternal"
These names must match the corresponding BPI Adapter properties (**MQAOApplicationID, MQAOInternalApplicationID**).

epicBodyCategory="DEFAULT", epicBodyType="DEFAULT"
These default entries must be defined for the receiving and internal application identifiers. These entries are used by the BPI Adapter to receive all of its messages so must always be present.

epicreceivemode
This determines the communications mode that other MQAO adapters use to communicate with Partner Agreement Manager. In the example above a communications mode of MQPP has been defined. MQAK supports a number of other MQSeries communications modes, for example, MQRFH2 could be used if messages to/from the broker need to pass through an MQSI broker. The communications mode defined for Partner Agreement Manager must be consistent with that defined for PAMInternal. Either both must be MQ based, for example, MQRFH2 and MQPP are consistent, or if JMS messaging is required, it must be specified for both PAM and PAMInternal.

Actually if JMS messaging is being used, all adapters communicating with the BPI Adapter must all be defined with a communications mode of JMS. If a mixture of JMS and MQ based adapters need to communicate with Partner Agreement Manager, another instance of the BPI Adapter can be configured using different application identifiers and different queues. MQAO adapters would then target the BPI Adapter that supported their communications mode.

For more information about MQAK configuration and communications modes, see the *MQSeries Adapter Kernel for Multiplatforms: Quick Beginnings* book.

BPI Adapter configuration

This section outlines the steps that need to be followed to import the BPI Adapter into Partner Agreement Manager. Upon successful completion of these, you should be able to start the default instance of the BPI Adapter. In the WebSphere Business Integrator environment this configuration is done by running the impgwadp.bat file.

To use the BPI Adapter, it is recommended that you first consult the following two chapters:

1. "Chapter 4. Operation" on page 17, which describes how you can change the behavior of the BPI Adapter to suit particular integration requirements.

2. "Chapter 5. Integration scenarios" on page 27, which describes how other MQAO adapters can be connected to the BPI Adapter to implement public processes within Partner Agreement Manager.

Manual importing of the BPI Adapter involves three separate import steps. The files that are imported should be found in the <Install_dir>\config directory after the Adapter Installation Program has been executed. These are:

BPI_Adapter_Type_1.xml
This defines the adapter type to Partner Agreement Manager, the operations it supports, and the properties that can be configured.

BPI_Adapter_Type_1_Java_Imp.xml
This defines a Java implementation of the adapter type. It names the class that Partner Agreement Manager invokes when an adapter operation is invoked within a private process.

BPI_Adapter_Type_1_Default_Instance.xml
This defines a default instance of the adapter implementation. All properties are set to default values. It is possible to run multiple instances of the BPI Adapter, perhaps with differing operational properties. This is discussed in "Adapter properties" on page 21.

To import the BPI Adapter, start the Partner Agreement Manager Adapter Server and perform the following tasks:

1. From the Adapter Server, launch the Adapter Designer from the **Tools** pull-down.
2. From the Adapter Designer **File** pull-down click **Import** and then click **OK** to select **Adapter Type**.
3. In the resulting dialog, import the adapter type by clicking file **BPI_Adapter_Type_1.xml**, which can be found in <Install_dir>\config.
4. Repeat steps 2 to 4 but this time to import file **BPI_Adapter_Type_1_Java_Imp.xml** as an Adapter Implementation.
5. Exit from the Adapter Designer and launch the **Adapter Manager** from the **Tools** pull-down.
6. From the **Adapter Manager**, **Server** pull-down, click **Import Adapter Instance** and select file **BPI_Adapter_Type_1_Default_Instance.xml**, which can again be found in <Install_dir>\config.

Following successful completion of the above steps, an adapter instance called **BPI Adapter Type 1 Default Instance** should be visible within the Adapter Manager. It will have a red light icon associated with it meaning that it is currently stopped. Before starting the instance ensure that:

- The queue manager PAM.QUEUE.MANAGER has been started.

If the queue manager is not started, Partner Agreement Manager can be configured to recover from this by retrying startup of the BPI Adapter instance a set number of times. The status traffic light shows a flashing shade of amber.

- The updates to the MQAK configuration that defined the PAM and PAMInternal names is visible to the Partner Agreement Manager Adapter Server. You can do this by setting a system environment variable called AQMSETUPFILE to point to the MQAK setup file, which in turn defines the location of the configuration information.
- The system classpath has been updated to include the location of the BPI Adapter class files. This update is made by the BPI Adapter installation program, but you may need to restart the machine for it to take effect.

You can now start the BPI Adapter by clicking the green start button in the toolbar in the Adapter Manager window.

Chapter 4. Operation

This chapter describes:

- The operations provided by BPI Adapter that can be used within private processes to both send and receive business objects.
- The properties that you can set to modify the behavior of the BPI Adapter, and tune it for use in different environments.

Adapter operations

The BPI Adapter provides operations to both send and receive business objects to and from other MQAO adapters. Both sending and receiving business objects must be performed as two separate adapter operations. This ensures the integrity of the request, ensuring that on a send no more than one message containing that business object is sent, and on a receive that the message is not lost before the business object has been returned to the private process.

Two kinds of receive operations are performed—a specialized form (**ReceiveInitiatingBO**) is required to receive a message that has initiated the public process.

InquireProcess and **SetProcess** operations are also provided, though these are not used within the private processes automatically generated within a WebSphere Business Integrator environment.

Note: You may find that each operation has an optional OperationStatus business object defined. As the optionality of this is not implementable within the Partner Agreement Manager Adapter API framework, these outputs should never be set.

SendBO

This operation sends the specified business object to a business application for processing. MQAO is responsible for sending the business object to the target adapter associated with that business application. The destination, or application identity, of the target adapter is inferred from the source of the last business object received (see "ReceiveBO" on page 18) for the current public process instance. In the case of public processes that originate at remote partners, then for the first **SendBO** operation, no destination will have yet been associated and the business object is routed to an appropriate destination by MQAO based upon its BodyCategory (the partner that defined it) and

BodyType (root element of the business object). An optional **DestinationID** parameter can be used to route to a specific destination.

INPUTS

BO_String (mandatory)
The business object to be sent as an XML string. To convert a business object to suitable string form, a script using the toString() procedure can be used.

BO_Category (optional)
The high-level qualifier for the business object to be sent. The recommended setting is the name of the partner who owns the definition of the business object. If not supplied, the MQAO message is sent with a default BodyCategory of `PAM`.

SendingPartner (optional)
The identity of the trading partner that originated the business object to be sent. This is typically a Data Universal Numbering System (DUNS) number. If not supplied, the MQAO message is sent without the ExternalID field set.

DestinationID (optional)
An override for the MQAO destination of the business object. Overriding with a destination of `NONE` results in the business object being routed by its BodyType (root element of business object) and BodyCategory (owning partner, that is, BO_Category above).

OUTPUTS
None.

CompleteSend

This operation completes a previous **SendBO** operation. This operation must be called as the next step in the private process after each **SendBO** operation.

INPUTS
None.

OUTPUTS
None.

ReceiveBO

This operation receives the next business object associated with the current public process instance. The business object is returned as an XML string and a script action in the private process should convert it into a business object of the correct type using the getBusinessObject() and fromXMLString() script procedures. For more information about script procdures, refer to the *Partner Agreement Manager Script Developer's Guide.*

For long-running processes, two adapter properties (**MinRetryInterval, MaxRetryInterval**) control the minimum and maximum retry intervals for the operation. Note that this operation must ALWAYS be followed in the private process by an additional extension action that calls the **CompleteReceive** operation.

INPUTS

None.

OUTPUTS

BO_String (mandatory)

The business object that was received from the business application as an XML string.

BO_Name (optional)

The root element of the received business object.

CompleteReceive

This operation completes a previous **ReceiveBO** operation in the private process. This operation must be called as the next step in the private process after each **ReceiveBO** operation.

INPUTS

None.

OUTPUTS

None.

ReceiveInitiatingBO

This operation works as for the **ReceiveBO** operation, but is used to receive the first business object in a public process instance that has been started by the BPI Adapter. The BPI Adapter browses for messages that are sent to it with a special TransportCorrelationID value of `GWNONE`. This value denotes that a new public process instance should be started and the business object associated with that message is received using this operation in the first step of the public process. The BPI Adapter stores the identifier of the initiating message in the BPI_Initial_MessageID variant of the process. This must then be specified on this call to receive the business object that started the process. Like the **ReceiveBO** operation, this operation must ALWAYS be followed by a further extension action, this time calling the **CompleteReceiveInitiating** operation.

INPUTS

Message_ID (mandatory)

The identifier of the message that started the public process instance. The BPI Adapter stores this as a string in the BPI_Initial_MessageID public process input variant.

OUTPUTS

BO_String (mandatory)
The business object that was received from the business application as an XML string.

BO_Name (optional)
The root element of the received business object.

CompleteReceiveInitiating

This operation completes a previous **ReceiveInitiatingBO** operation in the private process. This operation must be called as the next step in the private process after each **ReceiveInitiatingBO** operation.

INPUTS

Message_ID (mandatory)
The identifier of the message that started the public process instance as passed to the previous **ReceiveInitiatingBO** operation.

OUTPUTS
None.

InquireProcess

This operation returns information from the BPI Adapter for the current or specified public process instance.

INPUTS

PublicProcessID (optional)
The public process instance to inquire about. If omitted, defaults to the current public process instance.

OUTPUTS

CorrelationID (optional)
Correlation information relating to the business application that is communicating with the public process instance, for example, a purchase order number.

DestinationID (optional)
The logical identity of the MQAO adapter that is currently associated with the public process instance.

LocalPartner (optional)
Identity of the local partner associated with the public process instance.

PublicProcessType (optional)
The name of the public process type being executed by the public process instance.

SetProcess

This operation sets BPI Adapter properties for the current process instance.

INPUTS

LocalPartner (optional)
Identity of the local partner associated with the public process instance.

PublicProcessType (optional)
The name of the public process type being executed by the public process instance.

OUTPUTS
None.

Adapter properties

The BPI Adapter has a number of properties that provide configuration information, change its behavior, and tune the BPI Adapter to the environment it is executing in. It is possible to create multiple instances of the BPI Adapter and deploy a public process against the instance of the BPI Adapter that provides the required behavior. For example:

- Two instances could be created, one tuned for long-running processes, the other for short-running processes.
- Two instances could be created one that serviced MQAO adapters using MQSeries messaging, the other that communicated using JMS messaging.
- Two instances could be created, one that generated process completion notifications, the other that did not.

New BPI Adapter instances and adapter properties are created and modified using the Partner Agreement Manager Adapter Manager. The rest of this section discusses the properties of the BPI Adapter, which are categorized as follows:

configuration properties
These are mandatory and define how messaging is accessible to the BPI Adapter. Defaults are provided but they must be specified.

behavioral properties
These are optional and define how the BPI Adapter should operate with respect to the public process instances it becomes involved in. Default behavior is provided but they should be reviewed.

tuning properties
These provide tuning information that can be used to improve the

performance of the BPI Adapter. Default values are provided but they should be reviewed if the BPI Adapter is being used in a large-scale engagement.

debugging properties
These are used to make the BPI Adapter produce diagnostic information. By default, no debugging information is produced.

When running with multiple BPI Adapter instances, be very careful when modifying adapter properties. Property changes, most notably to the tuning properties, may affect other instances that are currently started. Where this is the case, it is noted in the following sections.

Configuration properties

Once defined for a BPI Adapter instance, the following properties should not be changed while active processes are using the adapter instance.

MQAOApplicationID
DEFAULT: PAM

The application identifier for the BPI Adapter as configured within MQAO. This is a logical identifier for the queue that the BPI Adapter uses to receive messages. If multiple BPI Adapter instances are required, you only need to configure a different application identifier, if the instances are to receive messages from a different queue.

MQAOInternalApplicationID
DEFAULT: PAMInternal

The internal application identifier for the BPI Adapter as configured within MQAO. This is a logical identifier for a queue that the BPI Adapter uses internally. Normally, the same internal queue is shared between all BPI Adapter instances.

InternalQueueManager
DEFAULT: PAM.QUEUE.MANAGER

The name of the queue manager that the BPI Adapter connects to for access to its internal queue. Normally this is the same queue and queue manager as configured within MQAO for the **MQAOInternalApplicationID** property.

InternalQueue
DEFAULT: SYSTEM.PAM.INTERNAL.STATE.QUEUE

The name of the queue that the BPI Adapter uses internally. Normally this is the same queue and queue manager as configured within MQAO for the **MQAOInternalApplicationID** property.

Behavioral properties

NotifyOnProcessCompletion
DEFAULT: true

Whether the BPI Adapter notifies business applications communicating with the adapter instance of normal and abnormal process completion. If the implementation of some public process types require process completion notifications, but others do not, two separate instances of the BPI Adapter can be created.

InitiationMessageTimeout
DEFAULT: 0 (no time-out)

Messages that initiate new public processes are removed from the queue by the **ReceiveInitiatingBO** adapter operation. If the public process fails before this operation is called, the message remains on the queue indefinitely and could cause further public processes to be started on restart of the BPI Adapter. The BPI Adapter monitors the length of time initiating messages remain on its input queue and can be configured to automatically remove failing messages. This is enabled by setting a non-zero value for this property that specifies the time in seconds after which messages are removed. Be careful not to set too low a value as this could cause the message to be removed too quickly and initiated public processes to fail. When different values are set in multiple adapter instances, an average is used.

Be careful when enabling this behavior if multiple BPI Adapter instances are being used. If a process is started that is using an instance that is currently stopped, the initiating message is still subject to time-out and could be removed before the adapter instance that is to receive it has been started again.

Tuning properties

DisableProcessInitiation
DEFAULT: false

Disables the adapter service that starts public processes. The service could be disabled to improve performance, if the BPI Adapter is only being used in processes that are initiated at partner sites. Processes are initiated from specific messages on the MQAO queue, as identified by the **MQAOApplicationID** property of the BPI Adapter. When multiple adapter instances are started for the same MQAOApplicationID, the initiation service is only started if it is enabled in all adapter instances.

MaxConnections
DEFAULT: 3

The maximum number of MQSeries connections being used by the BPI Adapter. In busy operation, the number of connections in use within the BPI Adapter equals at maximum the number of threads within the Adapter Server. This property can be used to restrict the number to less than this amount. Note that this property is cumulative across multiple instances of the BPI Adapter.

MaxRetryInterval
DEFAULT: 180000

The maximum retry interval in milliseconds used by the BPI Adapter during receive operations when a message is not immediately available. If the BPI Adapter is being used to service both rapid and slowly executing public processes, multiple adapter instances can be created with appropriate retry characteristics.

MinRetryInterval
DEFAULT: 1000

The minimum retry interval in milliseconds used by the BPI Adapter during receive operations when a message is not immediately available. For long-running processes, the retry interval increases gradually to that specified by the **MaxRetryInterval** property. If the BPI Adapter is being used to service both rapid and slowly executing public processes, multiple adapter instances can be created with appropriate retry characteristics.

ProcessCompletionFrequency
DEFAULT 15:

Governs the rate at which the BPI Adapter checks and therefore generates process completion events. It is only applicable to adapter instances that have the **NotifyOnProcessCompletion** property enabled. Consider changing this setting only if you are experiencing performance problems. The frequency should match the rate (per minute) at which processes using the BPI Adapter are expected to complete. For example, if one process completes every second, then set the frequency to sixty (per minute). Lowering the frequency could improve performance at the expense of delaying the generation of notifications. Raising the limit could improve the rate at which notifications are generated at the expense of performance. When different values are set in multiple adapter instances, an average is used.

ProcessInitiationFrequency
DEFAULT: 15

Governs the rate at which the BPI Adapter checks its receiver queue for messages that are to initiate new public processes. It is only applicable to adapter instances that have the **DisableProcessInitiation**

property set to false. Consider changing this setting only if you are experiencing performance problems or delays in starting new processes. The frequency should match the rate (per minute) at which new processes are expected to be started by the BPI Adapter. For example, if one process is expected to be started every second, then set the frequency to sixty (per minute). Lowering the frequency could improve performance at the expense of delaying the initiation of new processes. Raising the limit could improve the rate at which processes are initiated at the expense of performance. When different values are set in multiple adapter instances, an average is used.

Debugging properties

Debug

DEFAULT: false

Causes the BPI Adapter to write debugging information to the console of the Adapter Server. When enabled within a single adapter instance, this property causes all adapter instances to produce debugging information.

Chapter 5. Integration scenarios

This chapter describes possible integration scenarios for the BPI Adapter, which is designed to support a wide variety of integration scenarios.

Public processes can be executed synchronously or asynchronously by one or more MQAO adapters.

Typically an MQAO adapter is developed and associated with a single business object. However, if a business object can be processed simply, and the process is expected to execute in real time with the partner, a *synchronous* MQAO adapter can be developed that both produces and consumes business objects. Such an adapter could either:

- send a business object to the Partner Agreement Manager and wait synchronously for a reply business object, or,
- receive a business object from Partner Agreement Manager and then send back a reply business object.

Typically though, public processes are long lived, the internal processing to be performed involves human intervention, and it is more usual for business objects to be produced by some *asynchronous* trigger within the business application. However, the synchronous case is the easiest integration scenario to implement and this is the first integration scenario discussed. Later in the chapter, integration scenarios requiring multiple MQAO adapters to be developed are illustrated, see "Asynchronous adapters" on page 37.

First, let us summarize the responsibilities of an MQAO adapter that wants to achieve connectivity with the BPI Adapter, and consequently public processes executing on the Partner Agreement Manager Server.

Interacting with the BPI Adapter

MQAO adapters either send messages to the BPI Adapter or receive messages from it. A single adapter can even fulfill both roles if the public process is being executed in real time (that is, synchronously).

This section discusses considerations for developing target adapters to receive messages from the BPI Adapter and source adapters to send messages to it. Source adapters must adhere to a simple convention to ensure that the messages they produce are associated with the correct public process instance executing on Partner Agreement Manager. For more information, refer to the MQAO documentation.

Messages sent to Partner Agreement Manager must contain the TransportCorrelationID field set to the value contained within the last message received from Partner Agreement Manager for that public process instance. The exception to this is messages that are to initiate a new instance of public process, which must set the TransportCorrelationID field to the value GWNONE.

Basically this means that a target adapter upon receiving a message from Partner Agreement Manager must remember the TransportCorrelationID field from the received message and ensure that it is available to the source adapter that is to produce the next business object for that instance of the public process. In the synchronous case, this is relatively simple since the message containing the correlation information is still available. When the reply is to be generated asynchronously, some means of persisting the transport correlation identifier is required.

Just as the BPI Adapter uses the TransportCorrelationID to correlate which messages are associated with which public process instances, the MQAO adapter(s) that are interacting with the business system performing the backend processing usually requires a means of correlating incoming messages. It is suggested that the CorrelationID field be used for this purpose, for example, it could be used to store a business process identifier generated by the business system. This field is among a number of context fields that the BPI Adapter can associate with Partner Agreement Manager public process instances. When set by a source adapter, the BPI Adapter returns them in the next message sent to a target adapter.

Target adapter considerations

Before a target adapter can be developed to process a business object sent from a remote trading partner, it is important to understand the format of the message that the BPI Adapter sends to contain the business object. Table 1 on page 29 summarizes how MQAK header fields are set in the message. The business object itself is sent as body data, accessible as an XML string using the getBodyData() procedure. Some fields are always set by the BPI Adapter, other context fields are only propagated by the BPI Adapter and are only present if they were set in a message sent to the BPI Adapter in a previous message associated with the public process.

Table 1. Message field settings for target adapters

Field	Always set	Value
MessageType	Yes	Always set to `GatewayMessage` for messages containing business objects
SourceLogicalID	Yes	Identifies the application identifier of the BPI Adapter, typically `PAM`
RespondToLogicalID	Yes	Set as for the SourceLogicalID field, all business objects should be routed back to this application identifier
TransportCorrelationID	Yes	Contains correlation information that **must** be returned in the next message to be sent to the BPI Adapter for the public process instance
CorrelationID	No	Optional correlation identifier as set by an MQAO source adapter in a previous message (if any) sent to the BPI Adapter for the public process instance
TransactionID	Yes	The identifier that Partner Agreement Manager has associated with the public process instance
ProcessingCategory	Yes	The name by which Partner Agreement Manager identifies the type of public process being executed
SessionID	No	Optional session identifier as set by an MQAO source adapter in a previous message (if any) sent to the BPI Adapter for the public process instance
RelatedSubjectID	No	Optional related subject identifier as set by an MQAO source adapter in a previous message (if any) sent to the BPI Adapter for the public process instance
InternalID	Yes	The identity of the local trading partner, typically a DUNS number, associated with the Partner Agreement Manager Server that received the BO
ExternalID	Yes	The identity of the trading partner, typically a DUNS number, that originated the business object
BodyCategory	Yes	The name of the partner that defined the business object. This may not be an actual partner in the case of business objects associated with Partner Agreement Manager ProcessPaks, for example, contains ″AllianceInteract for RN 1_0″ for RosettaNet.
BodyType	Yes	The root element of the business object

Table 1. Message field settings for target adapters (continued)

Field	Always set	Value
BodyData	Yes	The business object as an XML string

Source adapter considerations

The most important consideration for a source adapter, or any adapter producing a message that is to be sent to the BPI Adapter, is the setting of the TransportCorrelationID field. If this field is not set to the correct value, the BPI Adapter cannot associate the business object with the correct public process instance. The business object itself, should be sent as an XML string, set using the setBodyData() procedure. The BPI Adapter does not validate the business object, if malformed or the wrong business object is sent, the private process that received it fails in the script that tries to create the business object of the expected type, thus causing the public process to fail. Last of all, the source adapter, as with any MQAO messaging, must ensure that the message is routed to the correct destination, that is, the BPI Adapter. This can be achieved by setting an explicit DestinationID (typically `PAM`) in the message, or relying upon MQAK routing configuration if a destination is not set.

Apart from messages that are to initiate a new public process, (these must name the type of public process to start, and also, if the process has been defined with a partner group, the partner with which to start it) the BPI Adapter does not perform stringent checking of any of the other fields set within the messages it receives. For example, it does not check that the BodyType field matches the root element of the business object that has been sent. For consistency within a WebSphere Business Integrator environment, all applications are recommended to propagate those fields that form the WebSphere Business Integrator context. Outside of this environment context propagation is not necessary, though it is recommended as a serviceability aid if nothing else. For example, setting the TransactionID field to contain the Partner Agreement Manager public process identifier at least makes it easier to identify which messages are for which public process instance both in a trace log and when viewing messages on a queue.

The following table summarizes how a source adapter should set the fields within the message that it sends to the BPI Adapter. Some of these are mandatory and must be set, recommendations are provided for those fields that are optional

Table 2. Message field settings for source adapters

Field	Optional	Suggested Value
MessageType	Yes	`GatewayMessage`
RespondToLogicalID	Yes	Nominates the MQAO adapter that is to receive the next message (a business object or a process completion notification) associated with this public process instance. If not set, MQAK defaults it to the identity of the sending adapter, that is, its SourceLogicalID. This field can be set to the value `NONE`, which causes the next message sent by the BPI Adapter to be routed by MQAK by its BodyCategory and BodyType.
TransportCorrelationID	No	If the message is to initiate a new public process, it must contain the value `GWNONE`. If the message is for an existing public process, it must contain the value of TransportCorrelationID from the last message sent by the BPI Adapter for the public process instance.
CorrelationID	Yes	Can be set to contain business system correlation information. This is returned by the BPI Adapter in the next message it produces for the public process instance.
TransactionID	Yes	The identifier that Partner Agreement Manager has associated with the public process instance
ProcessingCategory	Yes/No	The name by which Partner Agreement Manager identifies the type of public process being executed. Must be set for messages that are initiating new public processes.
SessionID	Yes	Can be set to contain session information, that could, for example, pertain to multiple related public process, This is returned by the BPI Adapter in the next message it produces for the public process instance.
RelatedSubjectID	Yes	Can be set to contain subject information, for example, something that identifies the process to the end user such as a PO number. This is returned by the BPI Adapter in the next message it produces for the public process instance.
InternalID	Yes	The identity of the local trading partner associated with the public process instance

Table 2. Message field settings for source adapters (continued)

Field	Optional	Suggested Value
ExternalID	Yes/No	The identity of the remote trading partner that is to receive the business object, typically their DUNS number. The actual trading partner that receives the business object is encoded within the Partner Agreement Manager public process definition. Must be specified in a message that is to initiate a public process that contains a partner group since it nominates the partner that the process is to be started with.
BodyCategory	Yes	Category for the business object. For consistency with messages produced by the BPI Adapter, could be set to the name of the partner that defined the business object. If not specified, it is set by the MQAK to `DEFAULT`.
BodyType	Yes	The root element of the business object. If not specified, it is set by MQAK to `DEFAULT`.
BodyData	No	The business object as an XML string.

Synchronous adapters

Let us return to the simple public process again, the most common type of public process (see Figure 1 on page 2). More complicated public processes are used later in this chapter to illustrate different integration scenarios. In this section we show how this public process could be implemented synchronously by both the Buyer and the Seller.

Since the process is being implemented in real time, it is possible for it to be implemented by both partners with a single MQAO adapter. If the processing that the Seller must perform is complicated, (perhaps human approval is needed before the order can be accepted), it would be more usual for the acceptance business object to be generated asynchronously by a second MQAO adapter. Similarly, it probably would not be practical for a single MQAO adapter to tie up resources waiting for this business object if it could take days to arrive. The next section ("Asynchronous adapters" on page 37) describes how separate MQAO adapters can be used by both Seller and Buyer.

Buyer implementation

The public process is initiated by the Buyer sending a message containing the purchase order business object to the BPI Adapter, see Figure 6 on page 33. Since the process is expected to execute synchronously (or at least within a short period of time) the source adapter can use the SendRequestResponse

function to both send the purchase order and receive the corresponding acceptance message back from the BPI Adapter.

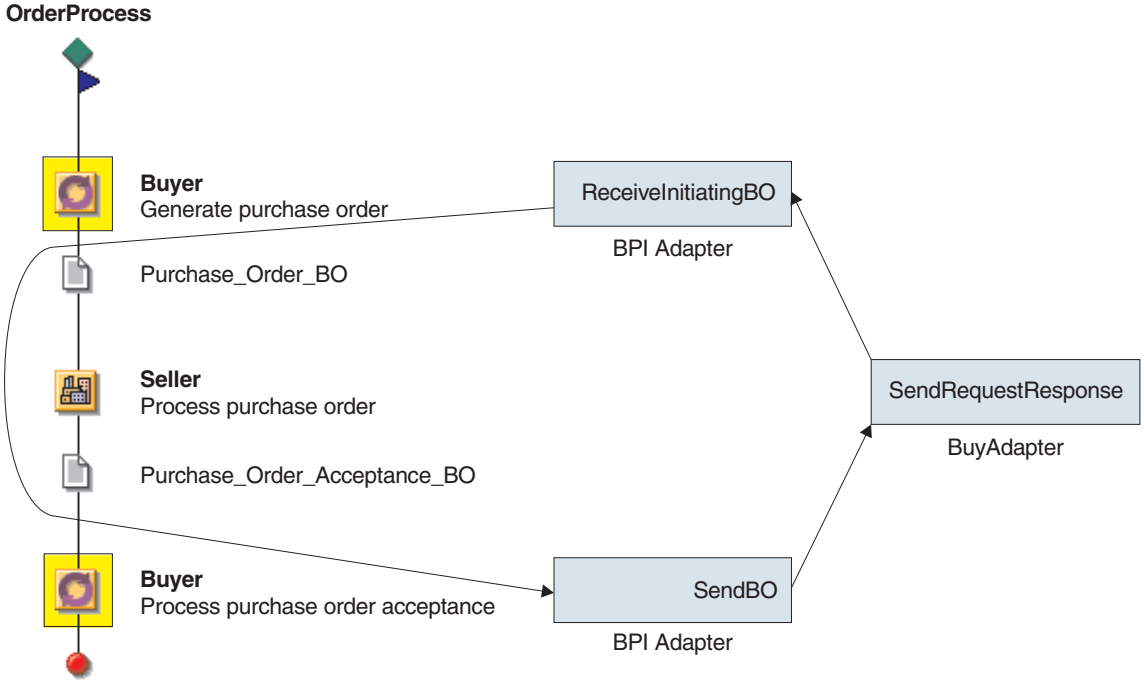

Figure 6. Integration scenario for synchronous adapters—Buyer implementation

The adapter in the Buyer, **Buy Adapter**, must be added to the MQAK configuration containing the entries for PAM and PAMInternal. The message that it produces to initiate the process must contain:

- TransportCorrelationID set to `GWNONE`
- ProcessingCategory set to `OrderProcess` (the name of the process to be run)
- BodyData set to contain the purchase order business object
- RespondToLogicalID defaults to the name of the source adapter, that is, `Buy Adapter`, in this case

The message must be routed to the BPI Adapter, which could be achieved by setting the DestinationID to `PAM`, though better dependence would be obtained by setting Partner Agreement Manager as a destination for the adapter's messages within the MQAK configuration as shown in Figure 7 on page 34. Note since this particular adapter only expects to send a single type of business object and receive a single type of business object, default adapter routing entries can be used, rather than entries that reflect the types of

business object being used. For simplicity in this example, it is assumed that the adapter is executing on the same queue manager as the BPI Adapter.

```
<ePICApplications o="ePICApplications">
  <ePICApplication epicappid="Buy Adapter">
    <AdapterRouting cn="epicadapterrouting">
      <epicmqppqueuemgr>PAM.QUEUE.MANAGER</epicmqppqueuemgr>
      <ePICBodyCategory epicbodycategory="DEFAULT">
        <ePICBodyType epicbodytype="DEFAULT">
          <epicdestids>PAM</epicdestids>
          <epicreceivetimeout>180000</epicreceivetimeout>
          <epicreceivemqppqueue>BUYER.QUEUE</epicreceivemqppqueue>
          <epicreplymqppqueue>BUYER.QUEUE<epicreceivemqppqueue></epicreceivemqppqueue>
          <epicreceivemode>MQPP</epicreceivemode>
        </ePICBodyType>
      </ePICBodyCategory>
    </AdapterRouting>
  </ePICApplication>
```

Figure 7. MQAK configuration—example 1

The <epicreceivetimeout> element defines in milliseconds the amount of time the SendRequestResponse method waits for the reply to be sent to it, in this case three minutes. When the SendRequestResponse function is used, the source adapter tries to receive the reply on the queue defined by <epicreplymqppqueue> (or the corresponding queue appropriate to the communication mode being used, for example, <epicreplyjmsqueue> for JMS messaging). In this case, the queue that has been nominated is the same as the normal receive queue for this application identifier. This allows the same application identifier to be used as the RespondToLogicalID field in the initiating message. Commonly, this would not be possible if the queue was being shared by other adapters, with the MQAK adapter daemon running against it. In this case, a different reply queue would need to be nominated, (see Figure 8 on page 35) and an alias application identifier set in the RespondToLogicalID field prior to calling SendRequestResponse. In this case, an alias of `Buy Adapter Reply` has been defined for a separate reply queue and this would need to be nominated as the recipient of the reply, see Figure 8 on page 35.

```
<ePICApplication epicappid="Buy Adapter">
  <AdapterRouting cn="epicadapterrouting">
    <epicmqppqueuemgr>PAM.QUEUE.MANAGER</epicmqppqueuemgr>
    <ePICBodyCategory epicbodycategory="DEFAULT">
      <ePICBodyType epicbodytype="DEFAULT">
        <epicdestids>PAM</epicdestids>
        <epicreceivetimeout>180000</epicreceivetimeout>
        <epicreceivemqppqueue>BUYER.QUEUE</epicreceivemqppqueue>
        <epicreplymqppqueue>BUYER.REPLY.QUEUE<epicreceivemqppqueue></epicreceivemqppqueue>
        <epicreceivemode>MQPP</epicreceivemode>
      </ePICBodyType>
    </ePICBodyCategory>
  </AdapterRouting>
</ePICApplication>
<ePICApplication epicappid="Buy Adapter Reply">
  <AdapterRouting cn="epicadapterrouting">
    <epicmqppqueuemgr>PAM.QUEUE.MANAGER</epicmqppqueuemgr>
    <ePICBodyCategory epicbodycategory="DEFAULT">
      <ePICBodyType epicbodytype="DEFAULT">
        <epicreplymqppqueue>BUYER.REPLY.QUEUE<epicreceivemqppqueue></epicreceivemqppqueue>
        <epicreceivemode>MQPP</epicreceivemode>
      </ePICBodyType>
    </ePICBodyCategory>
  </AdapterRouting>
</ePICApplication>
```

Figure 8. MQAK configuration—example 2

Seller implementation

The public process starts in the Seller with the arrival of the purchase order business object from the Buyer. A routing entry for this business object must be added to the MQAK configuration for the BPI Adapter to ensure that it is delivered to the correct MQAO adapter for processing. We could rely upon a default routing entry if this was the only public process, and consequently, the only business object that needed routing. This is unlikely to be the case, so for the following example we choose to add a routing entry that causes this business object to be processed by an adapter with an application identifier of `Sell Adapter`. Suppose the purchase order was defined by a partner with the name Buyer, and the order business object has a root element set to <PurchaseOrder>. These are used in the BodyCategory and BodyType of the routing entry added for the Partner Agreement Manager application identifier, see Figure 9 on page 36.

```
<ePICApplication epicappid="PAM">
  <AdapterRouting cn="epicadapterrouting">
    <epicmqpppqueuemgr>PAM.QUEUE.MANAGER</epicmqpppqueuemgr>
    <ePICBodyCategory epicbodycategory="Buyer">
      <ePICBodyType epicbodytype="PurchaseOrder">
        <epicdestids>Sell Adapter</epicdestids>
    <ePICBodyCategory epicbodycategory="DEFAULT">
      <ePICBodyType epicbodytype="DEFAULT">
        <epicreceivemqpppqueue>SYSTEM.PAM.RECEIVER.QUEUE</epicreceivemqpppqueue>
        <epicreceivemode>MQPP</epicreceivemode>
      </ePICBodyType>
    </ePICBodyCategory>
  </AdapterRouting>
</ePICApplication>
```

Figure 9. MQAK configuration for BPI Adapter

The Sell Adapter receives the purchase order, and assuming it can invoke the corresponding business application synchronously for processing, sends back acceptance business object as its reply, see Figure 10.

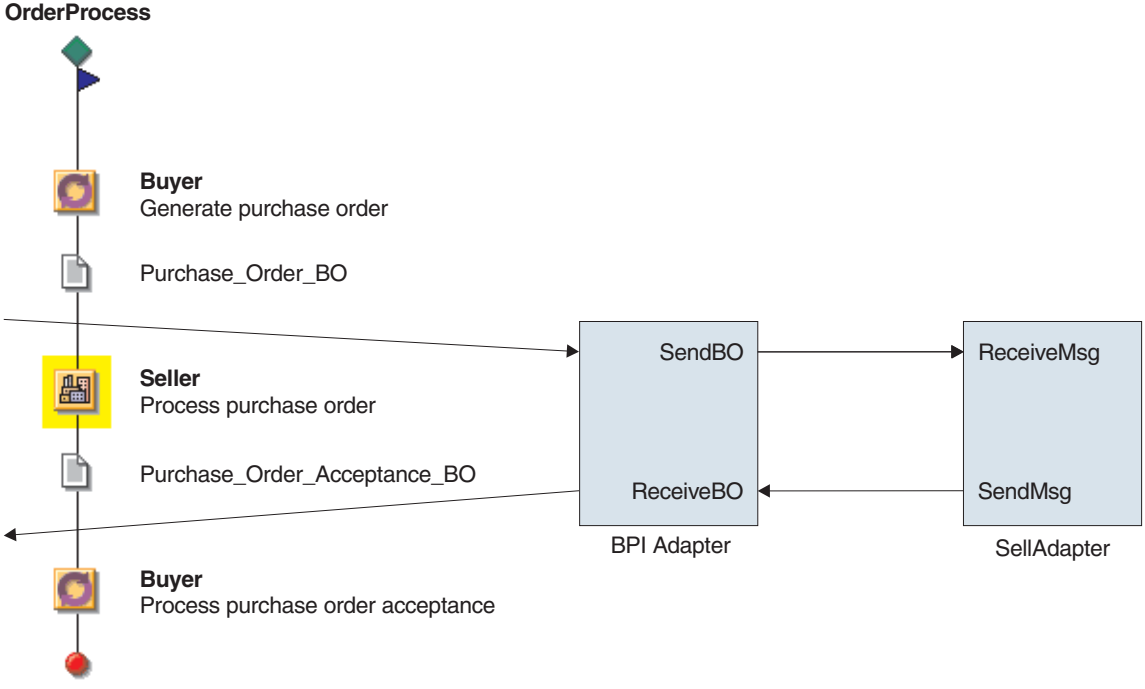

Figure 10. Integration scenario for synchronous adapters—Seller implementation

The reply message that the adapter produces must contain:

- TransportCorrelationID copied from the received message
- DestinationID copied from the RespondToLogicalID from the received message
- BodyData set to contain the purchase order acceptance business object

The MQAK configuration containing the entries for PAM and PAMInternal also must be modified to contain an entry for **Sell Adapter**. This is a standard entry so is not shown.

Asynchronous adapters

Most public processes are of a similar structure to the one used to illustrate the previous section about synchronous adapters. However, a more typical implementation of this process would involve the use of asynchronous adapters. In this scenario, both Buyer and Seller would require both a source and a target MQAO adapter to be developed. The process order may take days to process so it would be unrealistic for the Buyer to wait synchronously for the reply. Similarly, the business process that the Seller implements cannot be invoked synchronously, especially if human intervention is required to process the order.

Buyer implementation

As before, the public process is initiated by the Buyer sending a message containing the purchase order business object to the BPI Adapter, see Figure 11 on page 38. In this case, the source adapter would simply use the SendMsg function to send the purchase order. The purchase order acceptance when it arrived would be processed by a separate adapter.

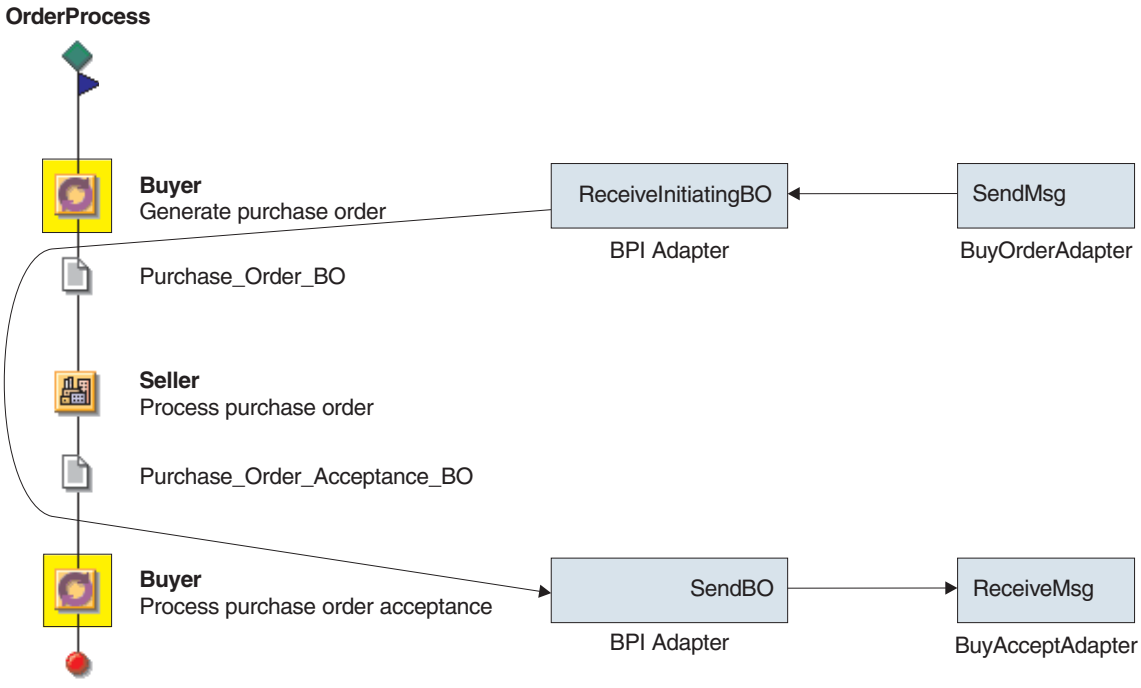

Figure 11. Integration scenario for asynchronous adapters—Buyer implementation

The **BuyOrderAdapter** would produce a similar message as before:

- TransportCorrelationID set to `GWNONE`
- ProcessingCategory set to `OrderProcess` (the name of the process to be run)
- BodyData set to contain the purchase order business object

This time however, you must set the RespondToLogicalID field to ensure that the acceptance business object is handled by the correct target adapter. There are two choices:

1. Explicitly name the target adapter by (in this case) setting RespondToLogicalID to `BuyAcceptAdapter`
2. Let the business object be routed to the appropriate adapter by setting RespondToLogicalID to `NONE`. In this case a similar routing entry would need to be added to the Partner Agreement Manager application identifier as shown in "Seller implementation" on page 35.

Seller implementation

The Seller would also require two separate adapters (see Figure 12 on page 39).

1. A target adapter called **SellOrderAdapter** to process the order.

2. When processing of the order was complete the business system would generate an event to trigger a source adapter, **SellAcceptAdapter** to send the acceptance business object back to the BPI Adapter.

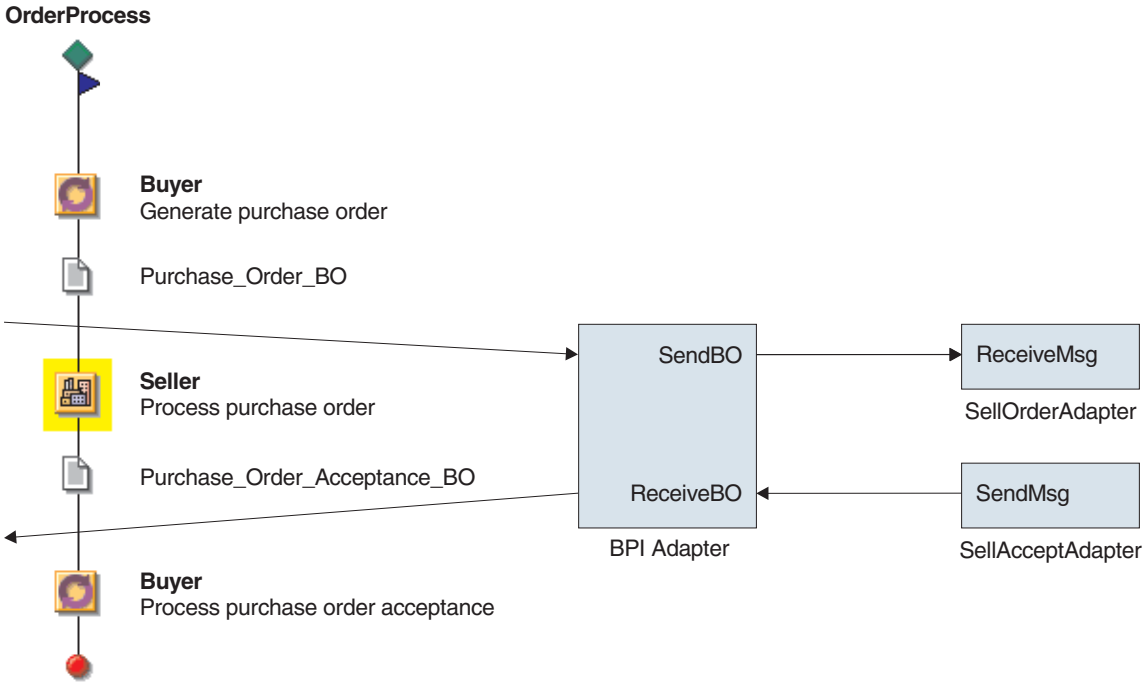

Figure 12. Integration scenario for asynchronous adapters—Seller implementation

The message that the adapter sent would be the same as for the synchronous adapter case. This time however, the TransportCorrelationID would need to be persisted so that it was available to the **SellAcceptAdapter** as it produced the message.

There actually is a third choice that involves setting an explicit DestinationID in the private process that sends the acceptance business object. This would involve editing the private process after it had been deployed to the Partner Agreement Manager Server.

Dealing with process completion notifications

The examples shown previously have assumed that the processing in the business system was complete when the last business object was sent back to the BPI Adapter. Typically however, the backend processing usually requires additional confirmation that this business object has been received by the

other partner. The BPI Adapter provides a means of informing the backend of the success or failure of the public process with the generation of process completion notifications. This feature must be enabled by turning on the **NotifyOnProcessCompletion** property of the adapter. Alternatively, you could create another adapter instance with the property turned on, if this feature is not required for the implementation of all public processes.

Process completion notification messages are similar to the messages that the BPI Adapter produces to contain business objects, as described in "Target adapter considerations" on page 28. The only differences are:

- The message contains no body data
- The message has a BodyCategory of `PAM`
- The message has a BodyType of either `Completed` if the process completed successfully, or `Cancel` if the process failed or was cancelled
- The message has a MessageType of `GatewayCommand`

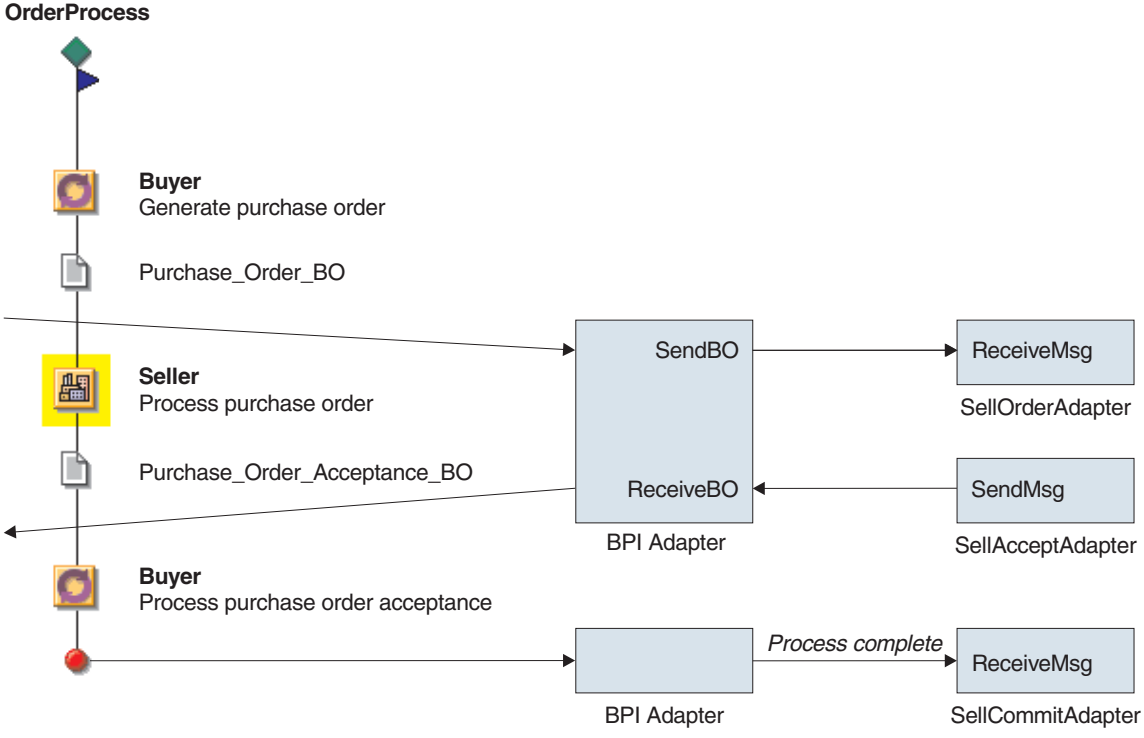

Figure 13. Process completion notification

Figure 13 shows how a process completion notification could be handled in the Seller after the process has been committed and acceptance business object has been successfully transmitted to the Buyer. In this case, a further adapter,

SellCommitAdapter is used to commit the order in the business system. Note that the implementation would also need to cope with process cancellation notifications should the public process fail or time-out before the acceptance business object was produced by **SellAcceptAdapter**. In this case, the BPI Adapter defers sending of the process cancellation notification until after it has received the outstanding business object.

Chapter 6. BPI Adapter verification

This chapter describes how to verify the installation and correct operation of the BPI Adapter.

The verification is done by the Adapter Installation Verification (AIV) program, which starts a Java process that sends a message to the BPI Adapter, which will initiate a Partner Agreement Manager public process, VerifyAdapter. VerifyAdapter comprises two private process steps. The first is ReceiveInitial which receives the message and creates a business object. This is passed to the second private process step, SendTerminal, which simply returns the message and terminates. A process completion message is then sent to the AIV program from the BPI Adapter.

The steps required to successfully run AIV are:

1. Install the BPI Adapter and AIV public process into Partner Agreement Manager.
2. Set up AIVTestQueue in MQSeries.
3. Enable the VerifyAdapter public process in Partner Agreement Manager via the Process Distribution Manager.
4. Grant privilege rights to PAMAdmin.
5. Run the AIV program to send and receive messages from Partner Agreement Manager and check the test and completion messages displayed.

Installing the BPI Adapter and AIV public process

Before you run the AIV program, you must have:

1. Run the Import Gateway Adapter Batch file (impgwadp.bat), if you are operating in a WebSphere Business Integrator environment, or
2. You would perform the configuration described in “Chapter 3. Configuration” on page 11 if you were operating in other environments.

You can confirm that installation was successful by viewing the list of available adapters in the Partner Agreement Manager Adapter Manager as follows:

1. Start the Adapter Manager.

2. Confirm the existence of the **BPI Adapter Type 1 Default Instance** in the list of Adapter instances. You can modify the properties for this adapter according to your requirements, but for this test you can use the default values.
3. Start the Adapter Designer application by selecting it from the **Tools** menu of the Adapter Manager application.
4. Confirm the existence of **BPI Adapter Type 1** and its Java implementation in the list of adapter types.
5. Start the BPI Adapter by clicking the green start button in the toolbar in the Adapter Manager window.

Setting up the AIVTestQueue in MQSeries

1. Start MQSeries and open the queue manager created during configuration.
2. Right-click on the **Queues** folder and click **new**, then **localQueue**.
3. Enter the queue name `AIVTestQueue` and accept all the default values.
4. Click **OK** to create the queue.

Enabling the VerifyAdapter process

The impgwadp.bat file imports a process called VerifyAdapter into Partner Agreement Manager. To enable this process to accept messages, perform the following steps:

1. Start the Partner Agreement Manager Process Manager.
2. Click **Processes** in the left pane of the Process Manager window. The name of your trading partner (TP) then appears as a sub-folder.
3. Expand your TP folder and select **VerifyAdapter** from the list of processes for the machine.
4. Click the sub-folder with the version number 1.0. Three process names should be listed: one public process and two private processes.
5. Right-click the public process, and select the **Process Distribution Manager** from the drop-down list. A separate Process Distribution Manager window should appear.
6. Activate each of the process states up to and including **Test Installation**. Confirm that a green flag appears on the 1.0 version folder of the VerifyAdapter process.

Granting privilege rights to PAMAdmin

PAMAdmin is set up as a user when you install Partner Agreement Manager. However, only minimal privilege rights are assigned as default and these are not sufficient to run AIV. To assign the required privilege rights:

1. Click **Administration** in the left pane of the Process Manager window.

2. Select the sub-folder **Users** and confirm the existence of two users **admin** and **pamadmin**.
3. Display the User Information panel by double-clicking the **pamadmin** entry.
4. Click on the **Access** tab and grant Edit privileges to **Business Objects, Processes** and **Auditor**.
5. Click **OK** to save these settings.

Running the AIV program

To run the AIV program:

1. At a command prompt change to the \config subdirectory where Partner Agreement Manager was installed.
2. Run the AIV.bat file.
3. Check that the test message is sent and received. All messages sent to and received by Partner Agreement Manager are sent to System.out by aiv.bat.
4. Check that the process completion message is received.

BPI Adapter verification

Chapter 7. Private process generation

This chapter describes the ways in which private process steps are generated for a public process.

Generating private processes at deployment time

In a WebSphere Business Integrator environment, public processes are created through Solution Studio and are then deployed to a run-time Partner Agreement Manager. You perform the deployment using the Solution Deployment Wizard in the Platform Console, which is part of the Solution Manager.

During deployment, the private process steps are generated for the public process steps owned by the local partner. This public process can then be distributed to the other partners involved (when using the PAM Channel) and finally put into production mode.

For more information about using the Solution Studio, refer to the *WebSphere Studio Business Integrator Extensions Developer's Guide*.

Generating private processes by importing a public process

There may be circumstances where you need to generate the private process steps without using the Solution Manager. An example is when you are the partner who has received the public process. In this case you use a tool to import the public process and generate the private process steps:

1. Using the Partner Agreement Manager Import/Export facility, select the public process for which you want to generate the private process steps.
2. Export the public process to a directory.
3. Invoke the tool for importing the process, bizProcessDeploy, which is in the *install directory*\config directory.

   ```
   bizProcessDeploy userid password filename
   ```

   where *userid* and *password* are those of a valid Partner Agreement Manager user, and *filename* is the location and name of the file that was exported in step 2.
4. You should now see a new version of the public process in the Process Manager along with the generated private process steps.

Limitations of using bizProcessDeploy

When importing the public process, a new version is always created. and the private process steps will have been generated for Version 2.0. This is not always desired because, if you have agreed on Version 1.0 of a public process with your trading partner, then this is the process that requires the private process steps.

To assign the private process steps to the correct version of the public process, you must perform some manual steps. This procedure assumes Version 1.0 is the original process and Version 2.0 is the process with the generated private process steps:

1. Open Version 2.0 of the public process. This will be used as reference.
2. In Version 1.0, select the step that should have a private process, and open up the private process using the menu. This private process will be blank.
3. From the toolbar select **Edit**, then **Insert Private Process**. A pop-up appears, which lists all the public process and their corresponding private processes.
4. You can now select the correct private process from Version 2.0. Use the opened Version 2.0 to make sure that the correct private process step has been inserted.
5. Save the private process step and when all steps have been performed save the public process. Version 2.0 will then match Version 1.0

You can now use Version 1.0 and delete Version 2.0 if required.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other 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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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 United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester SO21 2JN
United Kingdom

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 information 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 measures may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available system. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the application data of 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 claim related to non-IBM products. Questions on 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 include 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.

Trademarks

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

- DB2
- IBM
- MQSeries
- Tivoli
- WebSphere

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

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

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

Bibliography

This biblography lists publications in the Partner Agreement Manager and other libraries.

WebSphere Partner Agreement Manager library

- *Partner Agreement Manager Installation Guide*, GC34-5964
- *Partner Agreement Manager Administrator's Guide*
- *Partner Agreement Manager User's Guide*
- *Partner Agreement Manager Adapter Developer's Guide*
- *Partner Agreement Manager Script Developer's Guide*
- *Partner Agreement Manager API Guide*
- *Partner Agreement Manager Adapters for MQSeries User's Guide*
- *Partner Agreement View User's Guide*, GC34-5965

MQSeries publications

- *MQSeries for Windows NT Quick Beginnings*, GC34-5389

MQSeries Adapter Offering publications

- *MQSeries Adapter Kernel for Multiplatforms: Quick Beginnings*, GC34-5855
- *MQSeries Adapter Kernel for Multiplatforms: Problem Determination Guide*, GC34-5897
- *MQSeries Adapter Builder for Windows NT: Using the Control Center*, GC34-5882

Bibliography

Index

N

P

R

S

T

V

Part Number: BPIAAJ00
Program Number: 5724-A85

Printed in U.S.A.

(1P) P/N: BPIAAJ00