WebSphere® Partner Agreement View

User's Guide

Version 2 Release 2

GC34-5965-02

WebSphere® Partner Agreement View

User’s Guide

Version 2 Release 2

GC34-5965-02

Note: Before using this information and the product it supports, read the information in *Notices* on page 137.

Third Edition (July 2001)

This edition applies to version 2, release 2 of WebSphere Partner Agreement View (product number 5724-A86) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can make comments on this information via e-mail at idrcf@hursley.ibm.com.

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.

Table of Contents

Welcome to the Partner Agreement View User's Guide

This document describes Partner Agreement View 2.2 and explains how to install, configure and use it.

WebSphere Partner Agreement View provides a well-defined interface to WebSphere Partner Agreement Manager, allowing you to seamlessly integrate it with your Web applications. With Partner Agreement View, Partner Agreement Manager has visibility and control over partners connecting to your Web application in a consistent fashion with all other partners in your business-to-business environment.

All of the advantages of Partner Agreement Manager business-to-business relationship management are available to intercompany interactions that are based on a web browser.

To use Partner Agreement View, follow these general steps:

- To become familiar with the Partner Agreement View concepts, see *Introducing WebSphere Partner Agreement View 2.2* on page 1.
- To install and configure Partner Agreement View, see *Installing WebSphere Partner Agreement View* on page 9.

- To design processes that will be used with Partner Agreement View, see *Designing processes using Partner Agreement View* on page 29.
- To manage Partner Agreement View using the Channel Manager, see *Managing WebSphere Partner Agreement View* on page 39.
- To use conversation threading with Partner Agreement View, see *Creating conversations* on page 47.
- For a description of the key concepts of the Channel API, see *Using the Channel API* on page 59.
- To see a set of example processes, see *Sample public processes* on page 81.
- To install and configure Tomcat and the Apache Web server, see *Configuring Tomcat and Apache on UNIX* on page 119.
- To install and configure the IBM HTTP Server, see *Installing the IBM HTTP Server and configuring SSL* on page 125.
- To configure the Channel Interface and Channel API queues, see *Configuring per-partner or shared queues* on page 129.
- To change how Partner Agreement View participates in conversations, see *Setting the conversation aware mode* on page 133.

Who should use this information

Read this guide if you are:

- an administrator who will install and configure Partner Agreement View.
- a developer who will write custom web applications using Partner Agreement View.
- with business partner(s) using a web application.

Related information

For additional information see the following:

- The StartHere.htm file. This file contains links to the Partner Agreement View Readme and Release Notes. Before installation, you can link to the StartHere.htm file (via a redirect) by opening the StartHere.htm file in the root directory of the product CD-ROM. After installation, you can find the StartHere.htm file in the Docs\Partner_Agreement_View_StartHere directory of the Partner Agreement Manager installation.
- The *Partner Agreement Manager Installation Guide*, form number GC34-5964-02, which describes how to install Partner Agreement Manager.
- The *Partner Agreement Manager Administrator's Guide*, form number BIAAAB02, which describes how to set up, configure, and administer Partner Agreement Manager after you install it.
- The *Partner Agreement Manager User's Guide*, form number BIAAAC02, which describes how to start a Partner Agreement Manager session, design public and private processes, define element definition sets, create business objects, and manage process distribution.
- The *Partner Agreement Manager Adapter Developer's Guide*, form number BIAAAD02, which describes how to develop and administer adapters using the Partner Agreement Manager Adapter Development Environment.
- The *Partner Agreement Manager Script Developer's Guide*, form number BIAAAE02, which describes how to write scripts used in Partner Agreement Manager private processes and elsewhere.
- The *Partner Agreement Manager External API Guide*, form number BIAAAF02, which describes principles behind the Partner Agreement Manager External API. See also the Javadoc for the External API, which is installed in the Partner Agreement Manager Docs folder.
- The *Partner Agreement Manager Adapters for MQSeries User's Guide*, form number BIAAAG02, which describes how to install, configure, and run the Partner Agreement Manager Adapters for MQSeries.

Summary of Changes

This edition includes these changes since the previous, first, edition:

- *Logout*. Partner Agreement View 2.2 allows the PAV application to log out of the channel interface, providing additional security and simplifying session management.
- *Outbound Proxy Support*. Partner Agreement View 2.2 can work with outbound proxies that use authentication. Outbound proxy authentication is used within internal networks to ensure that only people and applications that are authenticated may communicate with an external network. Authentication in the outbound proxy is done with a standard user name and password combination. You can turn on the outbound proxy feature in Partner Agreement Manager after installation. Thereafter, all outbound HTTP communication will use the same user name and password combination for the proxy.

CHAPTER

1

Introducing WebSphere Partner Agreement View 2.2

Topics in this chapter include:

- *What you need to know* on page 2.
- *About WebSphere Partner Agreement View* on page 2.
- *About your PAV application* on page 5.
- *What you must set up to use Partner Agreement View* on page 6.
- *Planning your Partner Agreement View system* on page 7.
- *Support for conversation threading* on page 8.
- *Where to go from here* on page 8.

What you need to know

This guide assumes you know how to use Partner Agreement Manager to construct and manipulate business objects and processes. For more information on using Partner Agreement Manager, see the *Partner Agreement Manager User's Guide*. This guide also assumes that you are familiar with your organization's business processes.

The administrator deploying WebSphere Partner Agreement View needs to understand your WebSphere Partner Agreement Manager installation and be familiar with your HTTP server's configuration and administration.

The developer writing Web applications using Partner Agreement View must know how to develop Web applications in the Java language.

The process designer must know how to use Partner Agreement Manager. In particular, the process designer needs to know how to perform the following tasks:

- define and use business object types
- create public and private processes, and distribute and install them
- associate events with public processes, so the public processes run when the events are received

It is also useful for a process designer to write and read scripts. See the *Partner Agreement Manager Script Developer's Guide* for more information.

About WebSphere Partner Agreement View

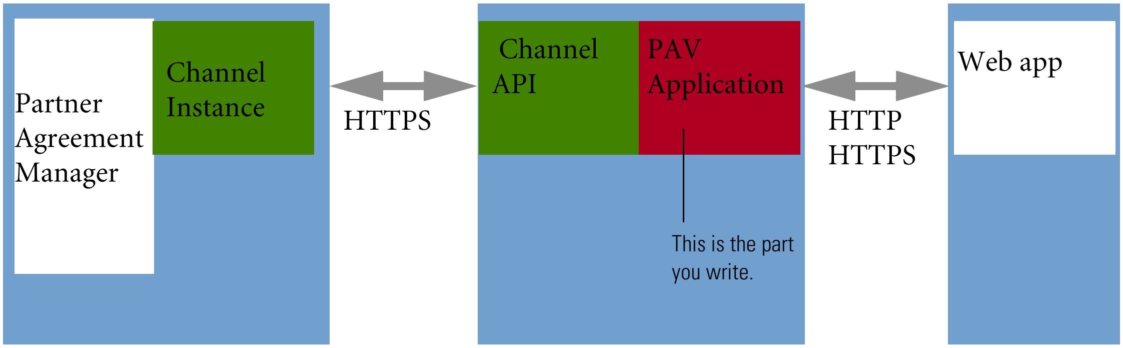
WebSphere Partner Agreement View (occasionally written PAV) consists of two distinct parts:

- One part of Partner Agreement View is installed on the computer running your Process Server. This part is called the **Channel Interface.**
- The second part provides the interfaces for your Web application. This part is called the **Channel API**.

The application you write to communicate between Partner Agreement Manager and your Web application is called your Partner Agreement View **application**.

Using the Channel API, your PAV application can communicate with Partner Agreement Manager, display business objects on a Web page, and generate business object forms. The Channel API includes libraries that allow you to do all these things quickly and easily with a minimum of custom coding.

These Partner Agreement View components enable Partner Agreement Manager processes to use your PAV application as an alternative channel for communicating with your business partners. The model looks like this:



Using Partner Agreement View in your processes

Partner Agreement View 2.2 supports Partner Agreement Manager public processes. You can create public processes that use Partner Agreement View to communicate with your partners. For complete information on public processes and how to design them, see the *Partner Agreement Manager User's Guide*.

Using your Partner Agreement View-enabled application, process designers can construct public processes that send and receive business objects in one-way and two-way communications.

You can start processes manually or schedule them. In addition, you can start processes from Partner Agreement View based on input from your partners.

About the Channel API

The Channel API provides classes and interfaces that allow your PAV application to:

- participate in public processes via a web application
- create and populate new business objects
- send business objects to Partner Agreement Manager
- receive business objects from Partner Agreement Manager via a queue or an event listener interface
- display business objects on your Web pages
- generate Web forms from business objects
- start Partner Agreement Manager public processes (initiated by a Partner Agreement View partner).

The Channel API can be used in your PAV applications to perform these functions, minimizing custom coding on the part of the developer.

The Channel API consists of a set of servlets and some supporting classes. It must run in a servlet engine. Any servlet engine that supports Java Servlet 2.2 can be used to host the Channel API. For your convenience, the Tomcat 3.2.1 servlet engine is included with the Partner Agreement View distribution. For more information on the Tomcat servlet engine, see http://jakarta.apache.org.

Tomcat 3.2.1 supports both HTTP and HTTPS. Although it is possible to send requests directly to Tomcat using HTTPS, you may prefer to use a different SSL-enabled web server and then redirect to Tomcat. Tomcat redirectors are available for many web servers, including Apache, IPlanet and the IBM HTTP Server. The Partner Agreement View distribution includes a redirector for the most popular web server on each platform. For more information on configuring the redirector for common configurations, see *Configuring Tomcat and Apache on UNIX* on page 119.

About your PAV application

Your PAV application can be any application that participates in a Partner Agreement Manager business process by exchanging Partner Agreement Manager business objects. A typical application might be a web user interface that exposes your business objects and processes to your partners.

To integrate with Partner Agreement Manager, your PAV application uses the Channel API.

Sending information to a partner

Sending information to a partner via PAV application follows this scenario:

Step 1 A Partner Agreement Manager process reaches a public step that includes this Partner Agreement View partner.

Step 2 Partner Agreement Manager sends the business object data in XML format to the PAV application.

Step 3 Using the Channel API, the PAV application gets the business object.

Step 4 The PAV application calls the Channel API to interpret the XML business object data and do whatever is appropriate, based on the nature of the application.

Receiving information from a partner

Receiving information from a partner via a PAV application follows this scenario:

Step 1 The PAV application receives data from the partner, converts it to one or more Partner Agreement Manager business object instances using the Channel API, and sends those business objects, now in XML format, to the Process Server.

Step 2 Partner Agreement Manager gets the business objects and the Partner Agreement Manager process proceeds.

What you must set up to use Partner Agreement View

To use Partner Agreement View, you must install and configure the following:

- WebSphere Partner Agreement Manager — You must have Partner Agreement Manager installed and configured correctly before starting the Partner Agreement View installation.
- HTTP server — This software must be installed and set up on the computer that will host your Partner Agreement View application to allow the Channel API to communicate with the Channel Interface. The HTTP server used must support servlets; if secure communication is required, it must support SSL. For security reasons, the HTTP server computer is not usually the same computer that is running WebSphere Partner Agreement Manager.

  Any computer can be turned into a HTTP server by installing Web server software and connecting the computer to your intranet or the Internet. There are many Web server software packages, including public domain software from NCSA and Apache, as well as commercial packages from IBM, Microsoft, Netscape, Sun, and others. We include the Tomcat server for use as a servlet engine in the Channel API distribution.

  In addition, you may wish to configure your HTTP server to use a proxy, particularly if you will be using it to host a Web application. You can use a proxy that is already installed at your location, or you can purchase a third-party proxy.

  IBM provides a web proxy implementation (on the Partner Agreement View installation CD) if you don't have your own third party proxy software or if your security policy does not allow you to open a port from your DMZ into your enterprise through the firewall.
- Partner Agreement View application — This is an application that you must write and install. Your Partner Agreement View application uses the Channel API to connect your Web application to Partner Agreement Manager. Generally speaking, writing this application involves custom Java coding. This document, the Partner Agreement View tools, samples, and the Partner Agreement View Javadocs are available to help application developers integrate their applications with Partner Agreement Manager.
- Partner Agreement View — An administrator must install and configure Partner Agreement View as described in this guide.

Planning your Partner Agreement View system

The following members should be on the planning team:

- A systems administrator, who will install and configure the Partner Agreement View software. This person will also install and configure the HTTP server.
- A process designer, who will create processes that use Partner Agreement View and define the Partner Agreement Manager business object types that are sent to and received from partners via the PAV application.
- An application developer, who will write the PAV application.

Setting up your Partner Agreement View system has the following steps:

1. After the team has looked through this guide and is familiar with Partner Agreement View, have a planning meeting to decide:
   - What partners will you communicate with using Partner Agreement View-enabled PAV applications and what are their Partner IDs?
   - What business object types you will send and receive?
   - Will you receive business objects by using processes that you run manually or processes that are scheduled? Or you will start processes when you receive input from your partners?
   - Will you require conversation threading? If so, what business object information will be used to define a conversation. For more information on conversation threading, see *Creating conversations* on page 47.
   - Will you be using a shared queue for all partners or a separate queue for each partner? For more information, see *Per-partner vs. shared queues* on page 64.
2. The process designer defines the business object types and the application developer writes the PAV application. If necessary, the administrator installs and configures the HTTP server on a test system. The administrator then installs Partner Agreement View and the PAV application on the test system.
3. The process designer designs the processes that use the PAV application.
4. After the PAV application is set up, team members test the Partner Agreement View system and modify it as needed.
5. Team members deploy the Partner Agreement View system.

Support for conversation threading

Partner Agreement View shares some common code with other channels supported by Partner Agreement Manager. Because messages sent to the channel applications from Partner Agreement Manager are sent asynchronously, it is important for Partner Agreement Manager to be able to distinguish the context of an incoming message.

The concept of **conversation threading** enables Partner Agreement Manager to associate an incoming response message from a channel partner with a previous request message sent to that partner.

Note: Because Partner Agreement View maintains its own association between outbound and inbound business objects (that is, it is process-aware), it does not need to use conversation threading. However, this feature is documented in this manual for completeness.

You define Conversations in the Conversation Editor. Conversations are described in detail in *Creating conversations* on page 47.

Where to go from here

Depending on your role, you will be interested in different parts of this document.

Administrators will want to go to *Installing WebSphere Partner Agreement View* on page 9.

Process designers will want to read *Designing processes using Partner Agreement View* on page 29 and *Sample public processes* on page 81.

Developers will need *Using the Channel API* on page 59 and the Javadocs installed with the Channel API. If upgrading applications from Partner Agreement View 1.1 to 2.2, see also *Upgrading Partner Agreement View applications from 1.1 to 2.2* on page 105.

CHAPTER

2

Installing WebSphere Partner Agreement View

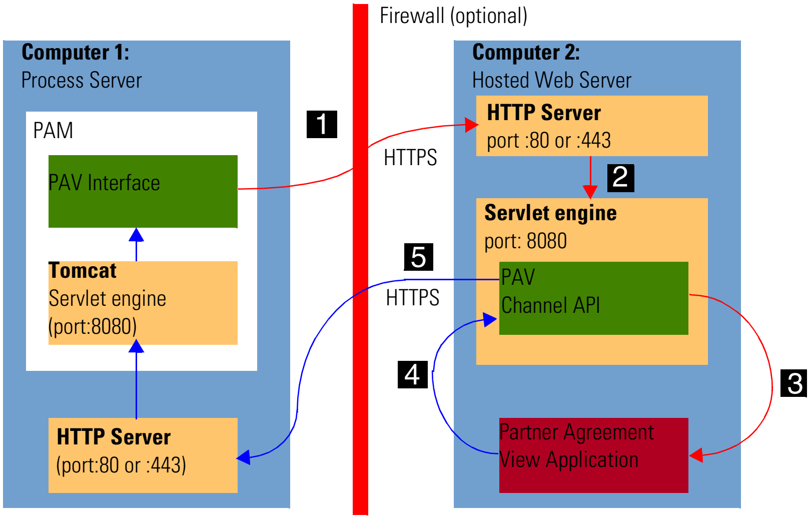
Read this chapter for information about installing Partner Agreement View. For a quick start on installing Partner Agreement View on Windows NT, see *Windows NT Quick Install Instructions* on page 95.

Sections in this chapter include:

- *Installing Partner Agreement View on Windows NT* on page 11.
- *Installing Partner Agreement View on UNIX* on page 18.

For information on upgrading from Partner Agreement View 1.1 to 2.2, see *Upgrading Partner Agreement View* on page 101.

Before beginning the installation, it is a good idea to understand how the components used by Partner Agreement View relate to one another.



In the figure, the red arrows (steps 1, 2, and 3) indicate communication from Partner Agreement Manager to the Partner Agreement View application; the blue arrows (steps 4 and 5) indicate communication from the Partner Agreement View application to Partner Agreement Manager.

The enumerated steps are:

1. The Partner Agreement View Channel Interface sends Active State to Computer 2 via HTTP or HTTPS on the Outbound_URL.
2. The HTTP server on Computer 2 redirects the request to the servlet engine.
3. The Partner Agreement View application requests the relevant Active State from the queue or receives it via a callback.
4. The Partner Agreement View application uses the Channel API to deliver a response business object.
5. The Channel API sends the response to Computer 1 via HTTP or HTTPS on the host address specified in AppChannel.properties.

Installing Partner Agreement View on Windows NT

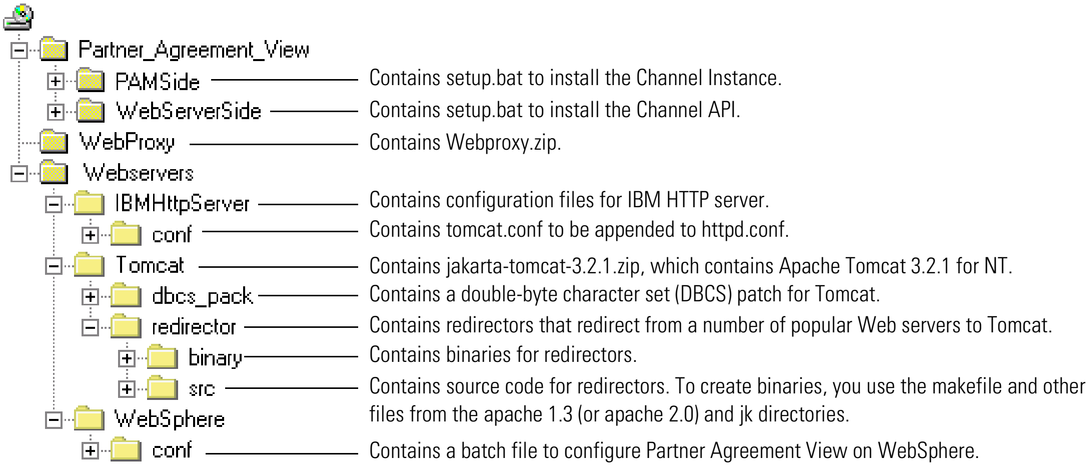
Typically, on Windows NT, you install Partner Agreement View in two parts, each on a separate computer.

- The first part, called the Channel Interface, is installed on the same computer that runs the Process Server. This Channel Interface is, as the name suggests, an interface between Partner Agreement View and Partner Agreement Manager.
- The second part, the Channel API, is installed on the computer where your Partner Agreement View application will reside. This Channel API exposes Partner Agreement Manager functionality such as business objects as well as communication functions.

There are other possible arrangements of the two parts of Partner Agreement View, but for clarity these directions only address this one.

Windows NT distribution image directory hierarchy

This figure shows the hierarchy of the Windows NT distribution image for Partner Agreement View.



Contains setup.bat to install the Channel Instance.

Contains setup.bat to install the Channel API.

Contains Webproxy.zip.

Contains configuration files for IBM HTTP server.

Contains tomcat.conf to be appended to httpd.conf.

Contains jakarta-tomcat-3.2.1.zip, which contains Apache Tomcat 3.2.1 for NT.

Contains a double-byte character set (DBCS) patch for Tomcat.

Contains redirectors that redirect from a number of popular Web servers to Tomcat.

Contains binaries for redirectors.

Contains source code for redirectors. To create binaries, you use the makefile and other files from the apache 1.3 (or apache 2.0) and jk directories.

Contains a batch file to configure Partner Agreement View on WebSphere.

Before you begin

This section lists the Partner Agreement View system requirements for a Windows NT installation. These requirements include what software you need to have installed before installing Partner Agreement View and what information you need to gather.

System requirements

Following are the minimum requirements for the Channel Interface portion of Partner Agreement View:

- IBM Partner Agreement Manager 2.2. For details on the requirements of Partner Agreement Manager, see chapter 1, "Installing the Process Server" in the *Partner Agreement Manager Installation Guide*.
- JDBC driver. The Partner Agreement Manager installation includes a JDBC driver.
- 20 MB disk space for the database, depending on the size of your business object instances and how frequently you intend to archive.

Following are the minimum requirements for the Channel API portion of Partner Agreement View:

- Tomcat version 3.2.1 HTTP server software (or any other Web server that supports servlets), along with the necessary data communications software and hardware. In the remainder of this manual, this is just called the "Tomcat servlet engine".
- If you intend to use SSL, an HTTP server that supports SSL (such as IBM HttpServer)
- JDK 1.2.2.
- JSDK 2.1 (or more recent)
- The Java integrated development environment (IDE) of your choice.
- 20 MB disk space for the Channel API.

Other necessary information

In addition to the hardware and software requirements, you must also know a few pieces of information. You must know:

- Your Partner Agreement Manager Partner ID. This is usually your company's D-U-N-S number. For more information on Partner Agreement Manager Partner IDs, see the *Partner Agreement Manager Installation Guide*.
- The host name and port for the host running the Partner Agreement Manager. This will be the host running your Partner Agreement View-interface applications. You set this up as part of the installation.

Installing the Channel Interface

As mentioned earlier, there are two parts to installing Partner Agreement View. This section describes how to install the Channel Interface. For information on installing the Channel API, see *Installing the Channel API* on page 14.

If you use just the Partner Agreement Manager client (the Process Server is installed elsewhere at your site) and you wish to develop Partner Agreement View-enabled applications, you must also install the Channel Interface on your local computer. The Channel Interface must be installed on all computers that need to use Partner Agreement View.

Note: If you are installing over an existing installation (upgrading), some of the dialogs in the Channel Interface installation might not appear.

To install the Channel Interface:

1 Shut down the Partner Agreement Manager Adapter Server and the Process Server.

2 Insert the installation CD into your CD-ROM drive.

A Click Start and choose Run.

B Type the letter of your CD-ROM drive followed by a colon and `Partner_Agreement_View\PAMSide\setup.bat` in the Open field and click OK. The Welcome screen appears. Click Next to continue.

3 The license screen appears.

Check the box to accept the terms of the license and click Next to continue.

4 The Partner Agreement Manager directory screen appears.

The default directory is the directory into which you installed Partner Agreement Manager. The screen also displays the Partner Agreement Manager Partner ID. Accept the defaults for both these fields, and click Next.

5 The confirmation screen appears.

The screen lists the name of the channel you are installing, such as Partner Agreement View. Ensure this is correct, then click Next to accept the Partner Agreement Manager installation information.

6 The Channel Instance/ID screen appears. Enter your Channel Instance name and Channel ID. The default name is `Partner Agreement View 1001`; the default ID is `1001`.

Click Next.

7 The final screen displays the destination folder, click Next to begin installing Channel Interface to the designated directory

8 Click Finish to complete the Channel Interface installation.

NOTE: Once you have installed the Channel Interface, you will need to set the Outbound_URL property to point to the Channel API computer's HTTP server. For more information, see *Setting Channel Profiles* on page 40. Note that the Outbound URL is case-sensitive.

INSTALLING THE CHANNEL API

The second part of the Partner Agreement View installation takes place on the computer that will host your Partner Agreement View application.

INSTALLING THE TOMCAT SERVLET ENGINE

Partner Agreement View for NT requires a servlet engine, such as Tomcat, or you can use any other web server that supports servlets. For your convenience, Tomcat 3.2.1 is included in the Partner Agreement View distribution.

NOTE: These instructions presume that you are installing on your computer's D: drive.

To install Tomcat:

1 Insert the installation CD into your CD-ROM drive.

2 Navigate to the `Webservers\Tomcat` directory on the CD.

3. Extract jakarta-tomcat-3.2.1.zip to D: to install Tomcat.
4. Rename the newly created jakarta-tomcat-3.2.1 directory to tomcat. You can get more information about the Tomcat from www.apache.org.
5. Add these Java packages to your Web server's CLASSPATH (that is, D:\Tomcat\lib). You can download them from the JavaSoft web site (www.javasoft.com/products):
   - Java Beans Activation Framework version 1.0.1—activation.jar.
   - Java Mail version 1.2—mail.jar.
6. In tomcat.bat (located in D:\tomcat\bin), add the following line after set _CLASSPATH:

   ```
   set JAVA_HOME=<path to your jdk>.
   ```

   Of course, you will need to make sure that you are pointing to the correct directories on your own computer.
7. You can test the Tomcat installation by starting Tomcat:
   A. Open up an MS-DOS window.
   B. Change directory to the Tomcat root directory:

      ```
      cd d:\tomcat
      ```
   C. Start Tomcat by entering the command:

      ```
      .\bin\tomcat start
      ```
   D. Open a browser and view one of the Tomcat sample pages such as: http://<*tc-computer*>:<*port*>/examples/jsp/index.html where tc-computer is the computer running Tomcat and port is the port on which Tomcat listens.
8. To shut down Tomcat, you use an MS-DOS window and:
   A. Change directory to the Tomcat root directory:

      ```
      cd d:\tomcat
      ```
   B. Stop Tomcat by entering the command:

      ```
      .\bin\tomcat stop
      ```

   Once you have Tomcat installed, you can configure it using the tomcat.bat file that is located in the \tomcat\bin directory. If you are using another HTTP server to handle SSL, you need to redirect your HTTP server to communicate with Tomcat.

NOTE: To enable Tomcat to work with DBCS data, you will need to copy the classes directory from `<channel>\Webservers\Tomcat\dbcs_pack\classes` on your distribution medium to your tomcat home directory

INSTALLING THE CHANNEL API

Now that you have the Tomcat servlet engine installed, you are ready to install the Channel API.

NOTE: If you are installing over an existing installation (upgrading), some of the dialogs in the Channel API installation might not appear.

To install the Channel API:

1 Start the Channel API installer.

  A Put the installation CD into your CD-ROM drive.

  B Click Start and choose Run.

  C Type the letter of your CD-ROM drive followed by a colon and `Partner_Agreement_View\WebServerSide\setup.bat` in the Open field and click OK.

  The Welcome screen appears. Click Next to continue.

2 The license screen appears. Read the license agreement. If you agree with the terms, check the Accept box and click Next to continue.

3 The conditions screen appears. Ensure that your Web server is not running and check the box. Click Next to continue.

4 The Web server type screen appears. Choose the Web server type (Tomcat or Other) and click Next to continue.

5 The servlet engine directory screen appears. Enter the path to your Web servlet engine directory.

  If you are using Tomcat, enter your Tomcat directory (`D:\tomcat`).

  Click Next to continue.

6 The Virtual Root (Relative to Web Server Root) screen appears.

  Take the default value.

IMPORTANT: The virtual root that you specify must be used for all other Partner Agreement View instances used on your Web server.

Click Next to continue.

7 The Channel ID and Channel Instance Name screen appears. The Channel ID is used by Partner Agreement Manager to distinguish between the channels you have installed, for example, RosettaNet Channel and Partner Agreement View. Each channel must have a Channel ID that is unique within your Partner Agreement Manager installation.

Unless you have another Partner Agreement View installed, the default Channel ID will be correct. If you need to change this default so it doesn't conflict with another Partner Agreement View installation, enter a number greater than 1000 that is not used by any other channel you have installed.

IMPORTANT: This number must be the same as the Channel ID you specified during the installation of the Channel Interface.

Note this number so you don't use it when you install any other channels. You can always check which channel ID values are in use by opening the Channel Manager from the Process Manager window and viewing the Channel profiles.

Unless you have another instance of your channel installed, accept the default for the channel instance name.

IMPORTANT: The Channel Instance name cannot start with a number. The installer does not check this.

Click Next to continue.

8 The Partner ID screen appears. Enter your Partner Agreement Manager Partner ID.

This must be the same Partner Agreement Manager Partner ID you used when you installed Partner Agreement Manager. For more information on Partner Agreement Manager Partner IDs, see the *Partner Agreement Manager Installation Guide*.

Click Next to continue.

9 The Web server information screen appears. Enter the host name and domain name of the Partner Agreement Manager Web server.

In Partner Agreement Manager WebServer Port, type in the value 80 for HTTP connections or 443 for HTTPS connections. Contact your Web server administrator to verify these values.

Important: The Process Server requires a Web server to properly handle approvals. However, for security reasons, this is not the Web server you will be using to serve your Partner Agreement View-enabled Web applications.

Click Next to continue.

10 The Extracting files screen appears. You will see a progress bar as Partner Agreement View installs. When the installer is finished extracting files, this screen will disappear and the completion screen appears.

11 Click Finish to complete the installation.

Opening the Partner Agreement View home page

When you are finished with the installation, start Tomcat, then use a Web browser to view the Partner Agreement View home page, which contains links to the documentation, samples, and much more information. The URL for the home page is:

http://<*machine.domain.com*>:<*port*>/WebSphere/
Partner_Agreement_View/jsp/home.jsp

Where *machine.domain.com* and *port* is the Web server and port where you installed the Process Server. Note that you must start Tomcat before you can view this page.

For example:

```
http://foo.bar.com:8080/WebSphere/Partner_Agreement_View/jsp/
home.jsp
```

Installing Partner Agreement View on UNIX

Currently supported UNIX platforms include AIX 4.3.3 and Solaris 2.7.

On UNIX, you typically install Partner Agreement View in three parts. Generally each of these parts is on a separate computer.

- The first part, called the Channel Interface, is installed on the same computer that runs the Process Server. This Channel Interface is, as the name suggests, an interface between Partner Agreement View and Partner Agreement Manager.

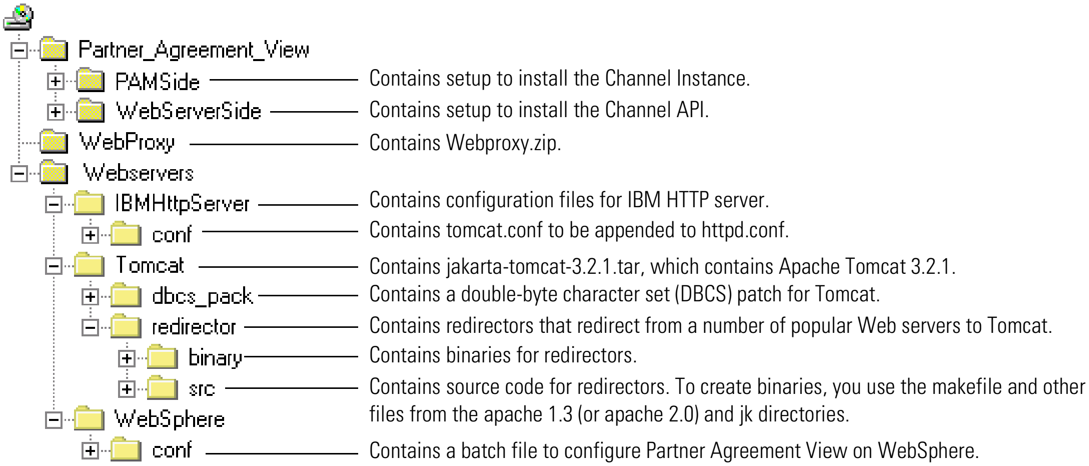
- The second part, the Channel API, is installed on the computer where your Partner Agreement View application will reside. This Channel API exposes Partner Agreement Manager functionality such as business objects as well as communication functions.
- The UNIX version of Partner Agreement View also includes a Channel addition to the Partner Agreement Manager client. Note that this is not needed for Partner Agreement Manager running on Windows NT, because on that platform, the server and client installations are integrated.

There are other possible arrangements of the Channel Interface and Channel API parts of Partner Agreement View, but for clarity these directions only address this one.

IMPORTANT: For specifics on installing Partner Agreement View on your particular variant of UNIX, see the StartHere.htm in the root of your installation CD.

UNIX install image directory hierarchy

This figure shows the hierarchy of the UNIX install image for Partner Agreement View.



```
Partner_Agreement_View
    PAMSide ———————— Contains setup to install the Channel Instance.
    WebServerSide —— Contains setup to install the Channel API.
WebProxy ——————————— Contains Webproxy.zip.
Webservers
    IBMHttpServer ——— Contains configuration files for IBM HTTP server.
        conf ———————— Contains tomcat.conf to be appended to httpd.conf.
    Tomcat —————————— Contains jakarta-tomcat-3.2.1.tar, which contains Apache Tomcat 3.2.1.
        dbcs_pack ——— Contains a double-byte character set (DBCS) patch for Tomcat.
        redirector ——— Contains redirectors that redirect from a number of popular Web servers to Tomcat.
            binary——— Contains binaries for redirectors.
            src ————— Contains source code for redirectors. To create binaries, you use the makefile and other
                      files from the apache 1.3 (or apache 2.0) and jk directories.
    WebSphere
        conf ———————— Contains a batch file to configure Partner Agreement View on WebSphere.
```

Before you begin

This section lists the Partner Agreement View system requirements. These requirements include what software you need to have installed before installing Partner Agreement View and what information you need to gather.

System requirements

The Channel Interface portion of Partner Agreement View must be installed on the computer running the Process Server. These are the minimum requirements for the Channel Interface portion of Partner Agreement View on UNIX:

- 20 MB hard disk space for Partner Agreement View Channel Interface.
- 150MB free disk space, after the installation is completed, to run Partner Agreement View.
- One of the supported UNIX platforms:
    - AIX 4.3.3, patch level 4330-05_AIX_ML. You can download the recommended patches from IBM at: http://www-1.ibm.com/servers/aix/download/index.html

      You can determine the current AIX patch level with this command:

      ```
      instfix -i | grep AIX_ML
      ```
    - Solaris 2.7, fully patched. You can download the recommended patches from Sun at: ftp://sunsolve.sun.com
- IBM Partner Agreement Manager 2.2 for one of the supported UNIX platforms (currently AIX and Solaris). For details on the requirements of Partner Agreement Manager, see chapter 1, “Installing the Process Server on UNIX” in the *Partner Agreement Manager Installation Guide*.
- A Web server that is accessible from the Partner Agreement Manager host computer (note that this is built into Partner Agreement Manager).

For security reasons, the Channel API is typically installed on a computer other than the one running the Process Server. Following are the minimum requirements for the Channel API portion of Partner Agreement View:

- Apache Tomcat version 3.2.1 HTTP server software (called “Tomcat servlet engine” in the rest of this manual) or any other Web server that supports servlets, along with the necessary data communications software and hardware. For information on configuring Tomcat with the Apache Web server, see *Partner Agreement Manager Installation Guide*.
- JDK 1.2.2.

- JSDK 2.1 (or more recent)
- The Java integrated development environment (IDE) of your choice.
- 20 MB disk space for the Channel API.

Other necessary information

In addition to the hardware and software requirements, you must also know a few pieces of information. You must know:

- Your Partner Agreement Manager Partner ID. This is usually your company's D-U-N-S number. For more information on Partner Agreement Manager Partner IDs, see the *Partner Agreement Manager Administrator's Guide*.
- The host name and port for the host running Partner Agreement Manager. This will be the host running your Partner Agreement View-interface applications. You set this up as part of the installation.

Important: Partner Agreement Manager requires a Web server to properly handle approvals. However, this is not the Web server you will be using to serve your Partner Agreement View-enabled applications.

Installing the Channel Interface

As mentioned earlier, there are three parts to installing Partner Agreement View. This section describes how to install the Channel Interface. For information on installing the Channel API, see *Installing the Tomcat servlet engine* on page 23 and *Installing the Channel API* on page 24. For information on installing the Channel addition to the Partner Agreement Manager client, see *Running the Channel Installer* on page 26.

Important: You must run the installer using an X11 graphical environment, such as XWindows or XServer. Before running the installer, be sure that the DISPLAY environment variable is properly configured for the computer running the X11 graphical environment.

Important: Before installing the Channel Interface, back up your Partner Agreement Manager and Partner directories.

The Channel Interface, must be installed on the same computer as the Process Server. To install the Channel Interface:

1. Log on as root.
2. Shut down the Partner Agreement Manager Adapter Server and the Process Server.
3. Mount the installation CD-ROM.
4. Navigate to the `Partner_Agreement_View/PAMSide` directory on the installation CD.
5. Execute the script named `setup` from a shell:

   ```
   sh setup
   ```

6. The Welcome screen appears. Click Next to continue.
7. The license screen appears.

   Check the box to accept the terms of the license and click Next to continue.
8. The Partner Agreement Manager directory screen appears.

   Enter the full path to the root directory of your Partner Agreement Manager installation and your Partner Agreement Manager Partner ID, then click Next.
9. The confirmation screen appears.

   The screen lists the name of the channel you are installing, such as Partner Agreement View. Ensure this is correct, then click Next to accept the Partner Agreement Manager installation information.
10. The Channel Instance/ID screen appears. Enter your Channel Instance name and Channel ID. The default name is `Partner Agreement View 1001`; the default ID is `1001`.

    Click Next.
11. The final screen displays the destination folder, click Next to begin installing Channel Interface to the designated directory
12. Click Finish to complete the Channel Interface installation.

NOTE: Once you have installed the Channel Interface, you will need to set the Outbound_URL property to point to the Channel API computer's HTTP server. For more information, see *Setting Channel Profiles* on page 40. Note that the Outbound URL is case-sensitive.

Installing the Tomcat servlet engine

The second part of the Partner Agreement View installation takes place on the computer that will host your Partner Agreement View application. Before you install the Channel API, you need to install Tomcat.

Important: Tomcat and the Channel API must be installed on the same computer where you are running the HTTP server you will use to communicate with your Partner Agreement Manager partners.

Partner Agreement View for both AIX and Solaris requires the Tomcat servlet engine, version 3.2.1 or any other HTTP server that supports servlets. Tomcat 3.2.1 is included in the Partner Agreement View distribution. For information on setting up your Web server with Tomcat, see *Configuring Tomcat and Apache on UNIX* on page 119.

To install the Tomcat servlet engine:

1. Navigate to the /WebServer/Tomcat directory on the installation CD-ROM. Tomcat for UNIX is distributed as a tar file: jakarta-tomcat-3.2.1.tar.
2. Make a directory on your Web server computer for Tomcat and untar the jakarta-tomcat-3.2.1.tar file into that directory (in the rest of this section, this directory is identified as <Tomcat>.

   You can get more information about Tomcat from www.apache.org.
3. Test the Tomcat installation by starting Tomcat:

   A. Set the JAVA_HOME environment variable to the full path to the root directory of your JDK. Be sure to export the environment variable.

   B. Set the TOMCAT_HOME environment variable to the full path to the directory where you installed Tomcat (usually <*Tomcat*>/jakarta-tomcat-3.2.1).

   C. Change directory to <*Tomcat*>/jakarta-tomcat-3.2.1. This is necessary for the startup script to find the files it needs.

   D. Start Tomcat by entering the command

   ```
   ./bin/startup.sh
   ```

   E. Open a browser and view one of the Tomcat sample pages such as: http://<*tc-computer*>:<*port*>/examples/jsp/index.html where tc-computer is the computer running Tomcat and port is the port on which Tomcat listens.

4 Shut down Tomcat by entering the command:

```
./bin/shutdown.sh
```

You must shut down Tomcat before you install the Channel API.

Once you have Tomcat installed, you can configure it using the server.xml file that is located in the <*Tomcat*>/jakarta-tomcat-3.2.1/conf directory. For additional information on configuring Tomcat with your Web server, see *Configuring Tomcat and Apache on UNIX* on page 119.

Note: To enable Tomcat to work with DBCS data, you will need to copy the classes directory from WebServer\Tomcat\dbcs_pack\classes on your distribution medium to your tomcat home directory

Installing the Channel API

Now that you have Tomcat installed, you are ready to install the Channel API.

To install the Channel API, you install Partner Agreement View on the Web server computer:

1 Log on as root.

2 Navigate to the Partner_Agreement_View/WebServerSide directory on the installation CD-ROM.

3 Execute the script named **setup**:

```
sh setup
```

4 A Welcome screen appears. Click Next to continue.

5 The license screen appears. Read the license agreement. If you agree with the terms, check the Accept box and click Next to continue.

6 The Web server type screen appears. Choose the Web server type (Tomcat or Other) and click Next to continue.

7 The servlet engine directory screen appears. Enter the full path to your Web servlet engine root directory.

If you are using Tomcat, enter your Tomcat directory.

Click Next to continue.

8 The Virtual Root (Relative to Web Server Root) screen appears.

Take the default value.

IMPORTANT: The virtual root that you specify must be used for all other Partner Agreement View instances used on your Web server.

Click Next to continue.

9 The Channel ID and Channel Instance Name screen appears. The Channel ID is used by Partner Agreement Manager to distinguish between the channels you have installed, for example, RosettaNet Channel and Partner Agreement View. Each channel must have a Channel ID that is unique within your Partner Agreement Manager installation.

Unless you have another Partner Agreement View installed, the default Channel ID will be correct. If you need to change this default so it doesn't conflict with another Partner Agreement View installation, enter a number greater than 1000 that is not used by any other channel you have installed.

IMPORTANT: This number must be the same as the Channel ID you specified during the installation of the Channel Interface.

Note this number so you don't use it when you install any other channels. You can always check which channel ID values are in use by opening the Channel Manager from the Process Manager window and viewing the Channel profiles.

Unless you have another instance of your channel installed, accept the default for the channel instance name.

IMPORTANT: The Channel Instance name cannot start with a number. The installer does not check this.

Click Next to continue.

10 The Partner ID screen appears. Enter your Partner Agreement Manager Partner ID.

This must be the same Partner Agreement Manager Partner ID you used when you installed Partner Agreement Manager. For more information on Partner Agreement Manager Partner IDs, see the *Partner Agreement Manager Installation Guide*.

Click Next to continue.

11 The Web server information screen appears. Enter the host name and domain name of the Partner Agreement Manager Web server.

In Partner Agreement Manager WebServer Port, type in the value 80 for HTTP connections or 443 for HTTPS connections. Contact your Web server administrator to verify these values.

IMPORTANT: The Process Server requires a Web server to properly handle approvals. However, for security reasons, this is not the Web server you will be using to serve your Partner Agreement View-enabled Web applications.

Click Next to continue.

12 The Extracting files screen appears. You will see a progress bar as Partner Agreement View installs. When the installer is finished extracting files, this screen will disappear and the completion screen appears.

13 Click Finish to complete the installation.

14 Navigate to the <*tomcat root*>/bin folder and change the permissions of startup.sh and tomcat.sh to be executable.

Running the Channel Installer

You must run the Channel Installer on the same computer as the AIX- or Solaris-compatible Partner Agreement Manager client in order to interact with the Partner Agreement View channel from the Partner Agreement Manager client.

To run the Channel Installer:

1 Click Start and choose Run.

2 Navigate to Partner_Agreement_View\PAMSide\setup.bat in your installation CD-ROM and click OK.

The Welcome panel appears for the Partner Agreement View installation. The Partner Agreement View installer guides you through the installation, one panel at a time. Click Next to continue.

3 The Partner Agreement View license agreement panel appears. You must agree to all the terms of the license to proceed with the installation.

Click Accept all terms of the license, then click Next to proceed.

4 The next panel shows the details of your Partner Agreement Manager client installation on this computer. Confirm that the default root directory and Partner ID are correct for your Partner Agreement Manager installation.

Click Next to continue.

5 The confirmation screen appears. Confirm that the data is correct and click Next.

6 The Channel Instance panel appears.

This table describes the information you must provide in each field:

In this field	Provide this information
Channel Instance	The name of the Channel Instance. This name cannot start with a number. This Channel Instance must be the same Channel Instance you specified for the Channel Interface and Channel API portions of the Partner Agreement View installation. Accept the default unless you have multiple Partner Agreement View installations.
Channel Type	The type of channel you are installing. This defaults to and is not editable.
Channel ID	A number that uniquely identifies this channel within your Partner Agreement Manager installation. This Channel ID must be the same Channel ID you specified for the Channel Interface and Channel API portions of the Partner Agreement View installation. Accept the default unless you have multiple Partner Agreement View installations.

When you have entered the information, click Next.

7 The destination directory screen appears.

The destination directory is your Partner Agreement Manager client directory and this is not editable. Click Next.

8 The Channel Installer starts copying files.

You might be prompted to overwrite some files. These files are updates to your Partner Agreement Manager client. Click Yes to All.

9 The final screen appears.

Click Finish to complete the installation.

Opening the Partner Agreement View home page

When you are finished with the installation, start Tomcat, then use a Web browser to view the Partner Agreement View home page, which contains links to the documentation, samples, and much more information. The URL for the home page is:

```
http://<machine.domain.com>:<port>/WebSphere/
Partner_Agreement_View/jsp/home.jsp
```

Where machine.domain.com and port is the Web server and port where you installed the Process Server.

For example:

```
http://foo.bar.com:8096/WebSphere/Partner_Agreement_View/jsp/home.jsp
```

CHAPTER

3

Designing processes using Partner Agreement View

Read this chapter for specific information on how to design processes that will be used with Partner Agreement View. Sections in this chapter include:

- *About public and private processes* on page 30.
- *About Partner Agreement View and public processes* on page 30.

About public and private processes

Partner Agreement Manager's two-level process model is designed to automate transactions and the exchange of information between partners in an extended enterprise. This model promotes cooperation between partners, yet maintains total security and flexibility for each partner's internal activities.

The first level is the **public process**, which defines the step-by-step flow of information and actions between two or more partners in an extended enterprise. One partner develops the public process, and all partners involved review and accept the process before it is implemented. The partner who designs a process is its owner.

The second level is the **private process**, which defines each partner's internal sequence of actions for its steps in the public process. Although all partners in a public process see and agree to its flow, each partner's private process is just that—private. The partner who develops a private process is the only one who can ever see it.

See the *Partner Agreement Manager User's Guide* for design guidelines and a complete explanation of how to design and develop processes.

About Partner Agreement View and public processes

Partner Agreement View 2.2 supports Partner Agreement View partners in public processes. A Partner Agreement View partner is a partner who uses Partner Agreement View in a public process to exchange data. The Partner Agreement View partner uses the Partner Agreement View-enabled application to interact with the host partner via a Web application. The host partner has Partner Agreement Manager and is the owner of the Partner Agreement View-enabled application. As a host partner, you can construct a variety of processes that use Partner Agreement View.

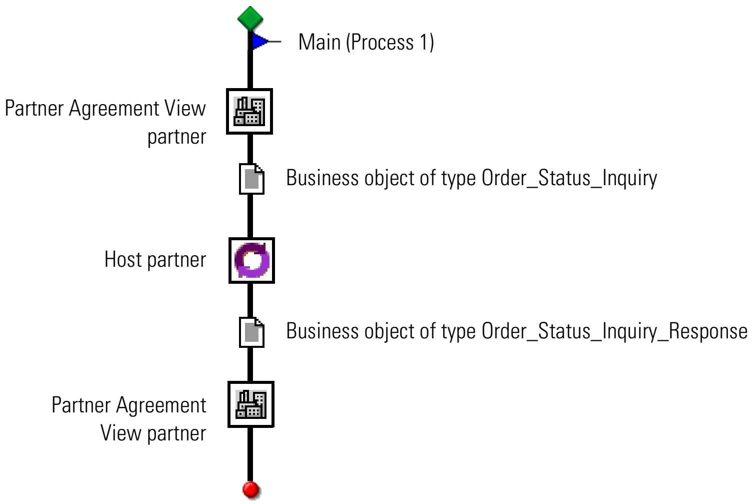
Processes you can create using Partner Agreement View

You can create a large variety of processes using Partner Agreement View, utilizing all the richness of the Partner Agreement Manager process development tools. These processes fall into three very general types:

- the Partner Agreement View partner initiates the process
- the host partner initiates the process
- the Partner Agreement View partner completes the process

The Partner Agreement View partner initiates the process

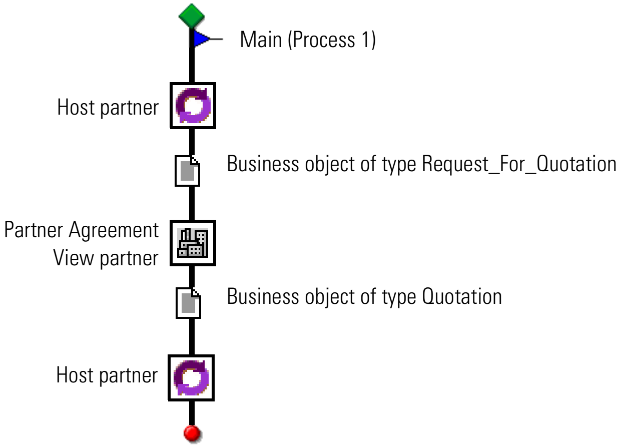
The Partner Agreement View partner can initiate a process. For example, if the Partner Agreement View partner is buying something from the host partner, the Partner Agreement View partner might issue an order status inquiry:


Main (Process 1)
Partner Agreement View partner
Business object of type Order_Status_Inquiry
Host partner
Business object of type Order_Status_Inquiry_Response
Partner Agreement View partner

The example SAMPLE_AppChannel_WAW_OSI_OSIR shows this process in more detail. See *OSI_OSIR* on page 88 for more details on this sample process.

The host partner initiates the process

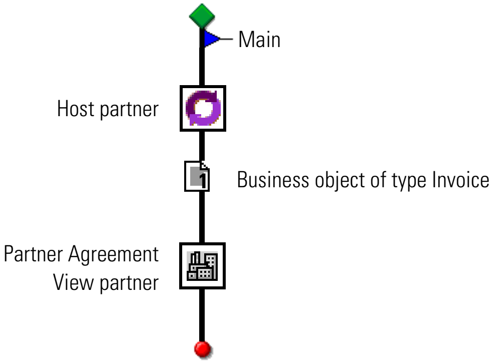
The host partner can initiate a process that includes a Partner Agreement View partner. For example, if the host partner is buying something from the Partner Agreement View partner, the host partner might start by asking for a quotation:


Main (Process 1)
Host partner
Business object of type Request_For_Quotation
Partner Agreement View partner
Business object of type Quotation
Host partner

The example SAMPLE_AppChannel_AWA_RFQ_Quotation shows this process in more detail. See *RFQ_Quotation* on page 92 for more details on this sample process.

The Partner Agreement View partner completes the process

The Partner Agreement View partner can be the terminal step of a process. This is shown in the first example *The Partner Agreement View partner initiates the process* on page 31. In that example, not only did the Partner Agreement View partner initiate the process, they also completed it. Another example would be if the Partner Agreement View partner were the buyer and the host partner sent an invoice:


Main
Host partner
Business object of type Invoice
Partner Agreement View partner

The example SAMPLE_AppChannel_AW_Invoice shows this process in more detail. See *Invoice* on page 91 for more details on this sample process.

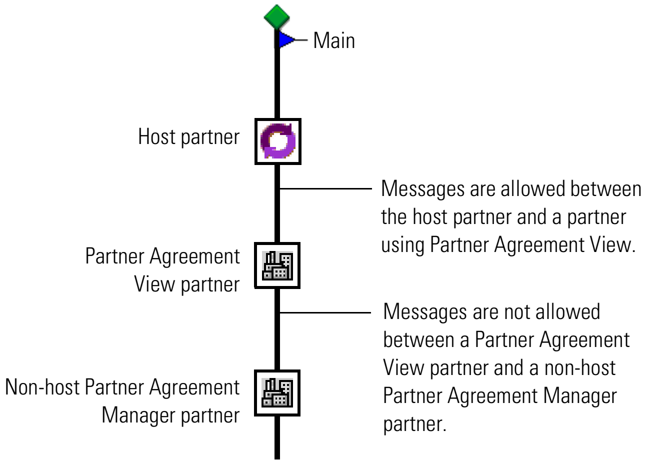
About the design considerations

When designing processes that include a Partner Agreement View partner, there are a few design considerations you must take into account. These considerations reflect a few corner cases and are unlikely to be present in your process design. The process verifier will reject any process that does not conform to these design considerations.

A Partner Agreement View partner can communicate only with the process owner

The process owner cannot author a process in which a non-host Partner Agreement Manager partner communicates directly with a partner using Partner Agreement View.

For example, this process flow is not valid:


Main
Host partner
Partner Agreement View partner
Non-host Partner Agreement Manager partner

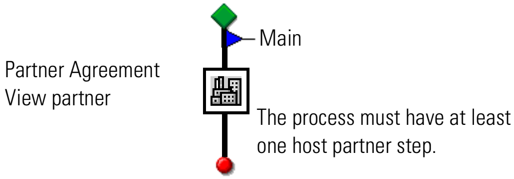
Messages are allowed between the host partner and a partner using Partner Agreement View.

Messages are not allowed between a Partner Agreement View partner and a non-host Partner Agreement Manager partner.

You cannot author processes that contain only a Partner Agreement View partner

A process that contains a Partner Agreement View partner must also contain at least one host partner step. The single step of a single step process cannot contain only a Partner Agreement View partner, either as a static partner or a group member.

For example, the following configuration is not valid:



The process must have at least one host partner step.

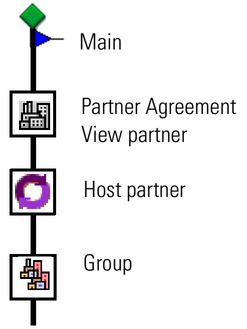
A Partner Agreement View partner step cannot be:

- immediately preceded or followed by a step assigned to anyone except the process owner or
- the single step of a single step process.

A Partner Agreement View partner cannot initiate a parametric process

A Partner Agreement View partner may not be the non-group initiator of a process which contains a group. In such a case, Partner Agreement Manager will not be able to determine which group member should be selected to participate in this process instance.

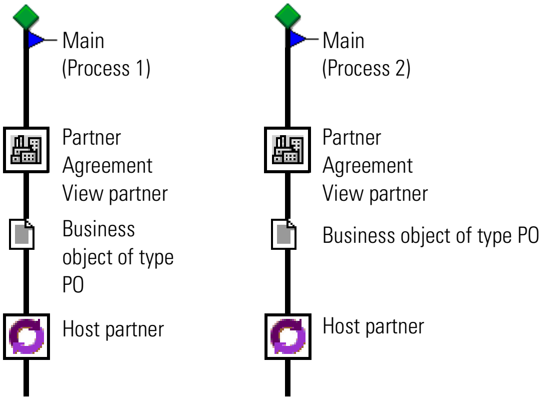
For example, this process is not valid:



A Partner Agreement View partner cannot initiate multiple processes with the same business object type

One Partner Agreement View partner cannot initiate multiple processes with the same business object type. In such a case, Partner Agreement Manager cannot determine which process to initiate.

For example, these two processes cannot co-exist:


Main (Process 1)
Partner Agreement View partner
Business object of type PO
Host partner
Main (Process 2)
Partner Agreement View partner
Business object of type PO
Host partner

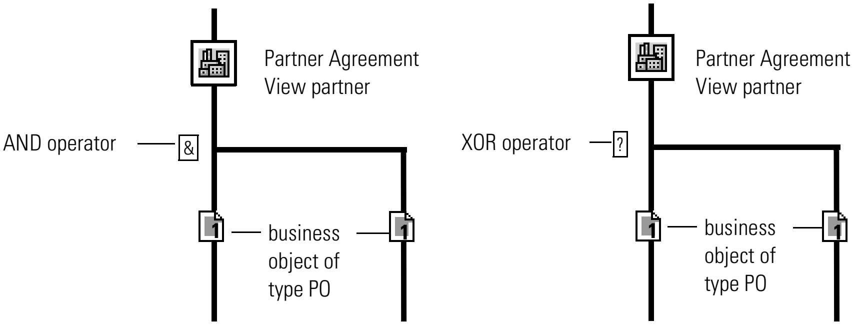
Note that two different Partner Agreement View partners could initiate different processes with the same business object type.

The default configuration includes this design consideration. If you need this functionality, contact IBM customer support about changing the configuration.

A Partner Agreement View partner cannot output two or more business objects of the same type

A Partner Agreement View partner cannot send two or more business objects of the same type from a single step. In such a case, the channel cannot determine the process path for each business object. This applies to both conjunctive and disjunctive out paths.

For example, neither of these processes is valid:

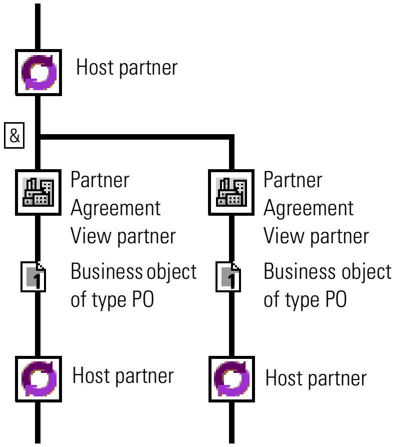

Partner Agreement View partner
AND operator
&
business object of type PO
Partner Agreement View partner
XOR operator
?
business object of type PO

The default configuration includes this design consideration. If you need this functionality, contact IBM customer support about changing the configuration.

A Partner Agreement View partner cannot output the same business object type on concurrent paths

A Partner Agreement View partner cannot output the same business object type on concurrently executing paths. In such a case, the channel cannot identify the path to which a newly arrived business object belongs. This applies only to conjunctive paths.

For example, this process is not valid:

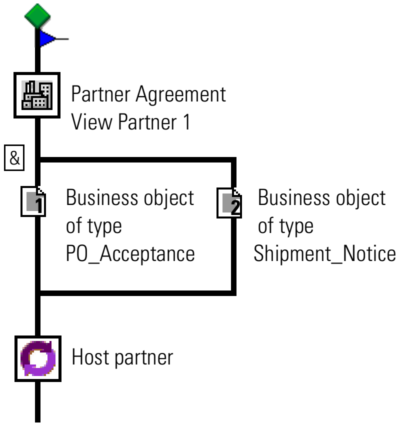


The default configuration includes this design consideration. If you need this functionality, contact IBM customer support about changing the configuration.

A Partner Agreement View partner cannot initiate a process with two or more business objects on conjunctive paths

If a Partner Agreement View partner is the initiating step, it cannot output two or more business objects on conjunctive paths.

For example, this process is not valid:



The default configuration includes this design consideration. If you need this functionality, contact IBM customer support about changing the configuration.

CHAPTER

4

Managing WebSphere Partner Agreement View

Read this chapter for information on how to manage Partner Agreement View partners and processes. Sections in this chapter include:

- *About the Channel Manager* on page 40.
- *Setting Channel Profiles* on page 40.
- *Adding Partner Agreement View partners* on page 43.

About the Channel Manager

Partner Agreement Manager 2.2 includes a tool, the Channel Manager, that you can use to manage Partner Agreement View processes and partners.

Important: The Channel Manager is part of the Partner Agreement Manager client. If your Process Server is running on UNIX, you must use the UNIX variant-compatible Partner Agreement Manager client.

Setting Channel Profiles

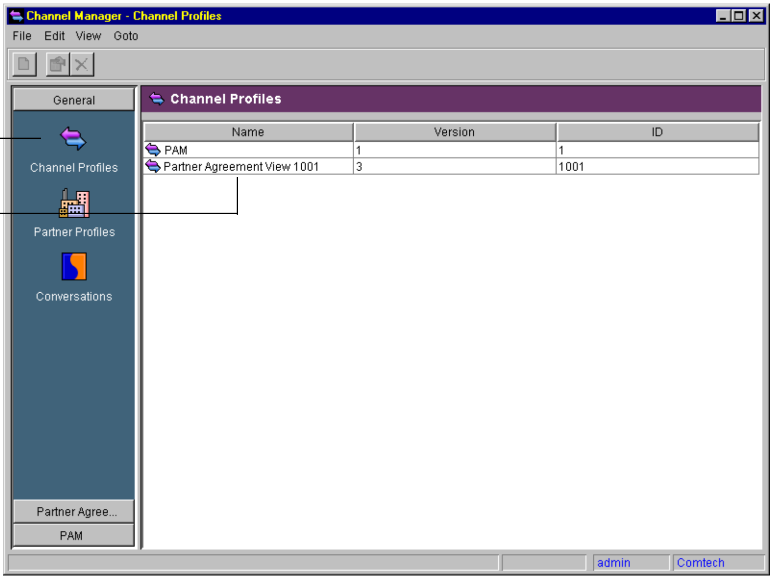
Before you can use a channel, you must edit the channel's profile. You edit the channel's profile using the Channel Manager. To launch the channel manager:

1. In the Partner Agreement Manager window, click Actions and select Channel Manager.
2. The channel manager window appears.

To edit a channel profile:

1. In the Channel Manager, click on the Channel Profiles icon.

The installed Channels appear in the right-hand pane.



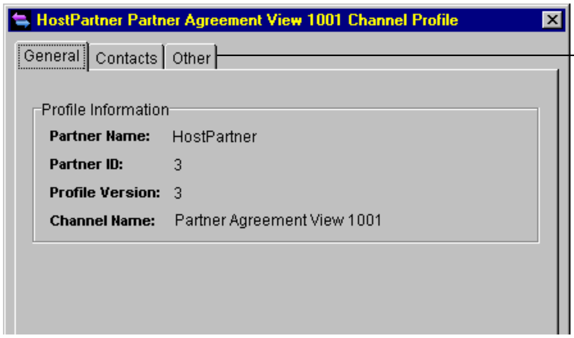
Name	Version	ID
PAM	1	1
Partner Agreement View 1001	3	1001

Click the Channel Profiles icon...

...to display the channels here.

2 Double-click on the name of the channel for which you want to edit the profile.

The Channel Profile screen appears.



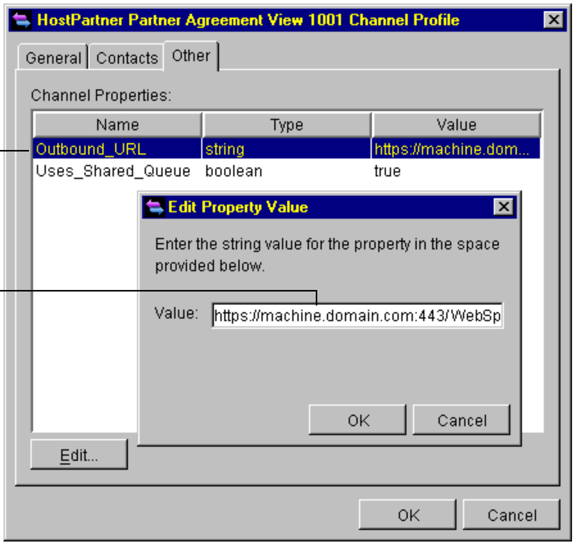
Partner Name: HostPartner
Partner ID: 3
Profile Version: 3
Channel Name: Partner Agreement View 1001

Click on Other tab.

3 Click on the Other tab, and then double-click on the Outbound_URL property.

The Edit Property Value screen appears, with this default value:

https://machine.domain.com:443/WebSphere/PAV/servlet/AppChannelPOBox



Channel Properties:

Name	Type	Value
Outbound_URL	string	https://machine.dom...
Uses_Shared_Queue	boolean	true

Enter the string value for the property in the space provided below.

Value: https://machine.domain.com:443/WebSp

Double-click on the property...

...and edit this value.

4. There are several changes you should make to the string value for the property (note that the Outbound URL is case-sensitive):
   - The protocol can be either HTTP or HTTPS. Chose the appropriate one.
   - You must replace machine.domain.com with the fully qualified dot path to the Web server that is running your Partner Agreement View-enabled application.
   - If your Web server is listening on a different port (usually 443 for HTTPS or 80 for HTTP), change the port value.
   - You can modify the virtual root of the servlet (WebSphere/PAV), however, if you do, it must be the same as the string that you specified in response to the Doc Base prompt when you installed the Channel API.
5. Click OK to set the new Outbound URL value.
6. Click OK to close the Channel Profile screen.
7. Close the Channel Manager window.

Adding Partner Agreement View partners

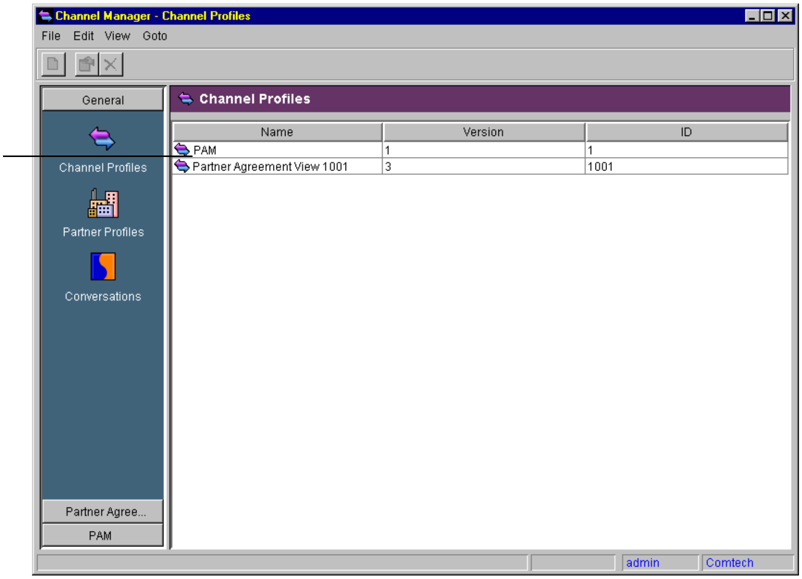
Before your partners can participate in processes using Partner Agreement View, you need to set that partner up in the Channel Manager as a Partner Agreement View user.

To add a Partner Agreement View partner:

1. In the IBM Partner Agreement Manager window, choose Channel Manager from the Actions menu.

The Channel Manager window appears.

These are the channel profiles currently set up.



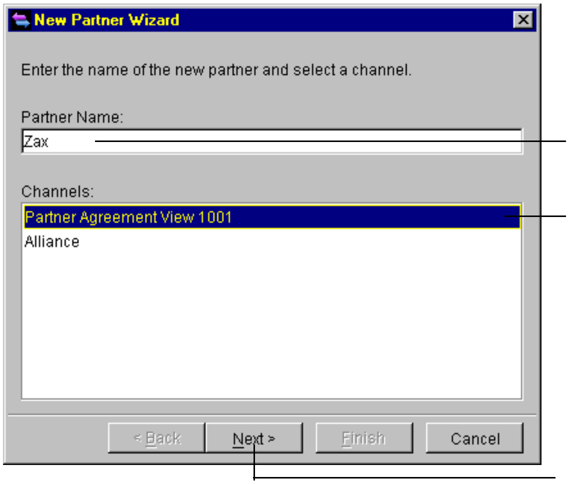
Channel Profiles

Name	Version	ID
PAM	1	1
Partner Agreement View 1001	3	1001

2 Click the Partner Profiles icon.

3 Click the New Partner button to start the New Partner Wizard.

The New Partner Wizard screen appears.



Enter the name of the new partner and select a channel.

Partner Name: Zax

Channels:
Partner Agreement View 1001
Alliance

New partner name

Channel for new partner to use

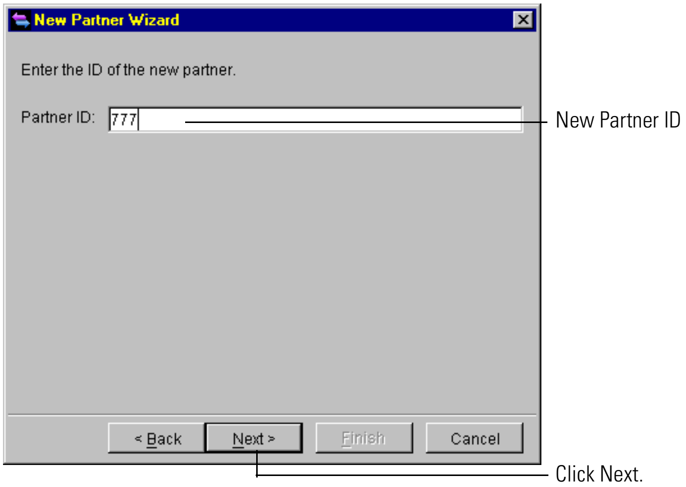
Click Next.

4 Enter a name for the new partner and choose the name of the channel used in installation as the channel you wish this partner to use.

If you used a channel ID different from the default, your Channel will reflect that.

5 Click Next to continue.

The Partner ID screen appears.



Enter the ID of the new partner.

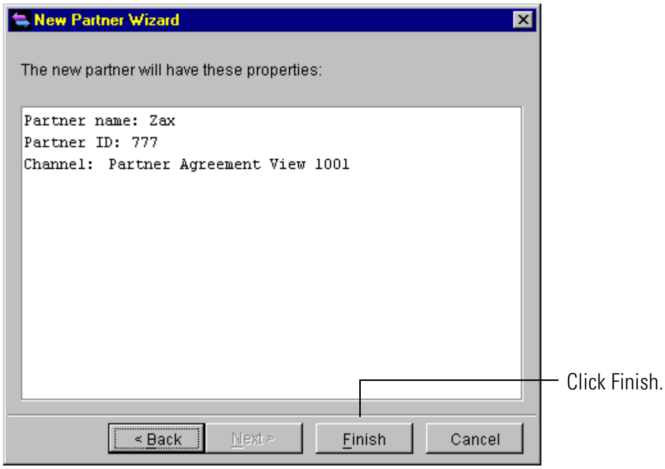
Partner ID: 777

New Partner ID

Click Next.

6 Enter a Partner ID for the new partner and click Next to continue.

The partner information confirmation screen appears.



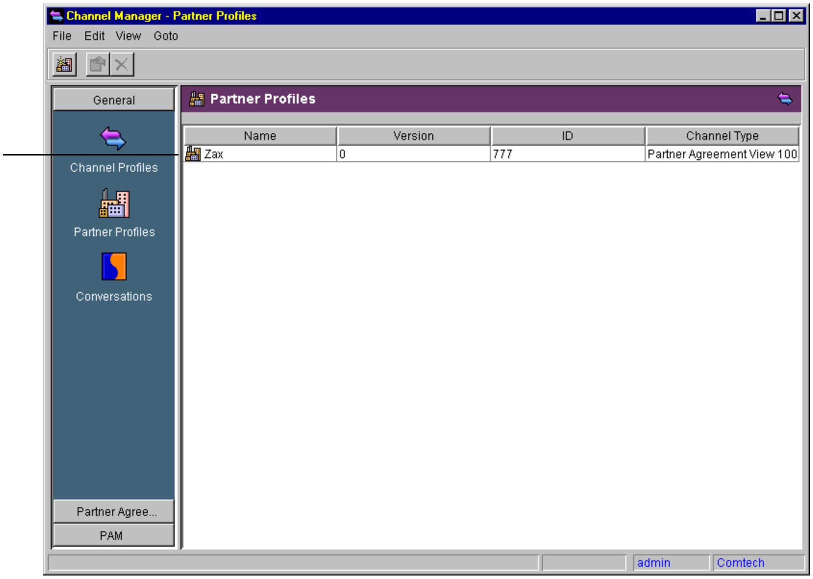
The new partner will have these properties:

```
Partner name: Zax
Partner ID: 777
Channel:  Partner Agreement View 1001
```

Click Finish.

7 Click Finish to complete.

The New Partner Wizard will close and you will see the new partner in the Partner Profiles window.



Name	Version	ID	Channel Type
Zax	0	777	Partner Agreement View 100

The partner you just added appears here.

CHAPTER

5

CREATING CONVERSATIONS

Partner Agreement Manager matches up responses to requests by associating business objects it receives with those that it has sent. Usually this information is maintained as process metadata between Partner Agreement Manager and its partners, but in some cases, the partner has no provision for maintaining this process metadata.

Read this chapter for information about creating conversations that allow Partner Agreement Manager to identify responses from a channel partner.

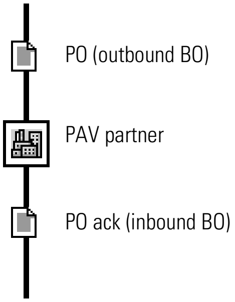
- *About conversations* on page 48.
- *About the conversation editor* on page 50.
- *Creating conversations* on page 53.

NOTE: Conversation threading is available in Partner Agreement View, but is not turned on by default. Generally, conversation threading is not necessary for Partner Agreement View because your Partner Agreement View application can maintain the association between outbound and inbound business objects. If you need to turn conversation threading on in Partner Agreement View, please see *Setting the conversation aware mode* on page 133.

About conversations

The *conversation threading* feature allows you to use unique information shared by outbound and inbound business objects (such as a PO number) to route inbound business objects to the appropriate Partner Agreement Manager process.

When a public process includes a channel partner, you usually pass a business object to the partner and, in many cases, the partner returns another business object.



The conversation associates fields in the outbound business object (in this case a purchase order) and uses the values in these fields to associate the inbound message (a purchase order acknowledgement) with the appropriate process.

Note: You cannot create two conversations that use the same inbound and outbound business objects, even if the fields used in conversations are different.

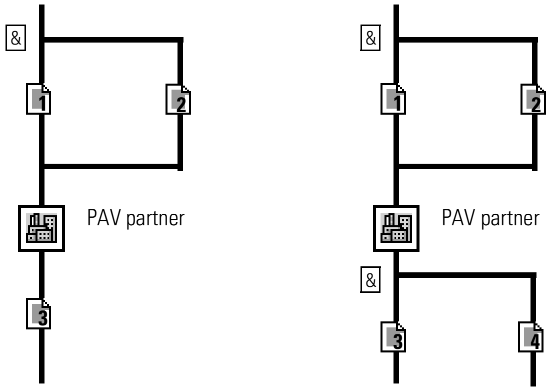
If conversation threading is turned on an you do not create a conversation for the outbound and inbound business objects used by a channel partner, the Partner Agreement Manager process verifier will issue an error, stating that the conversation is not defined.

A conversation is global. Once you have created a conversation, it is used to evaluate all inbound messages routed through the channel.

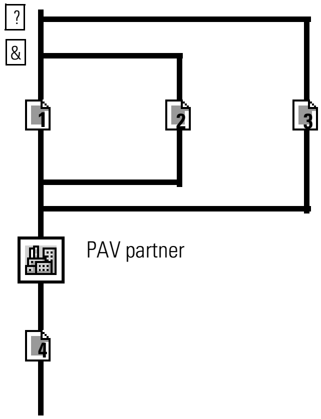
If Partner Agreement Manager receives an inbound business object from a channel partner, but cannot match the business object to a running process, Partner Agreement Manager looks for a process that can be started by the partner using that business object. If Partner Agreement Manager finds such a process, it starts it; if no process is found, Partner Agreement Manager reports an error initiating a process.

Requests with multiple business objects

Each conversation can have one or more outbound business objects, joined with an AND, but can only have one inbound business object. For example, the process flow on the left can be identified by a single conversation (object 1 and 2 outbound with object 3 inbound). Whereas the flow on the right must be identified by two separate conversations (object 1 and 2 outbound with object 3 inbound; object 1 and 2 outbound with object 4 inbound):



If the outbound business objects are joined by an OR, each outbound set of business objects requires a separate conversation. For example, this process flow requires two conversations (object 1 and 2 outbound with object 4 inbound; object 3 outbound with object 4 inbound).



About the conversation editor

This section describes some of the general functionality found in the Conversation editor. For information on how to start the Conversation editor and create conversations, see *Creating conversations* on page 53.

When creating a conversation, you start by selecting the inbound business object type (that is, the business object returned from the channel partner). Then, you select one or more outbound business object types (the business object passed to the channel partner).

The Conversation editor is divided into two panels: the outbound panel and inbound panel.

- The outbound panel (on the left) shows the business object type that is sent to the channel partner.
- The inbound panel (on the right) shows the business object types that can be returned from the channel partner.

NOTE: At times the terms “outbound” and “inbound” can be confusing. Just remember that the perspective of the Conversation editor (and this documentation) is from the Partner Agreement Manager perspective. Business objects are “outbound” when they are being sent from Partner Agreement Manager to the channel partner; business objects are “inbound” when they are sent from the channel partner to Partner Agreement Manager.

The lower portions of each panel display the description of an element selected in the corresponding upper portion.

You can resize the panels by dragging their borders.

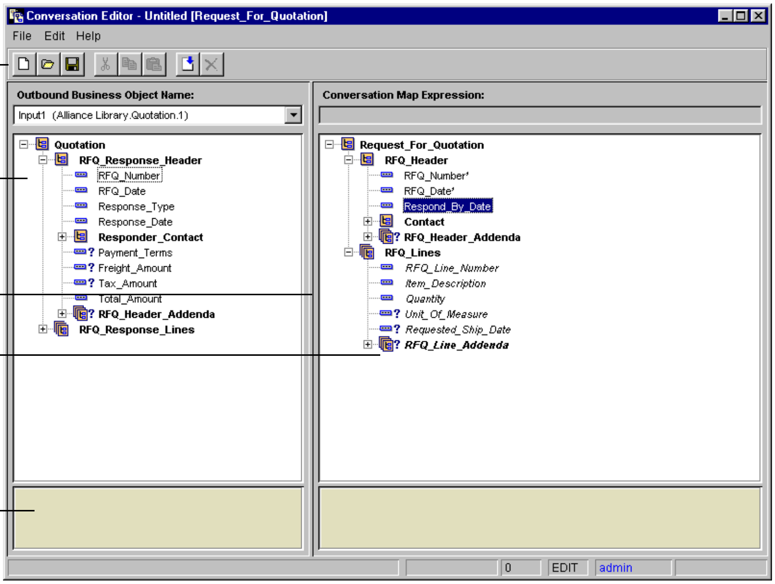
Toolbar buttons give you quick access to common commands.

This panel shows the business object type being sent to the partner.

Drag a panel’s border to resize the panel.

This panel shows the hierarchy of elements in business object type being returned from the partner.

This area shows a description of the selected element.



Conversation Editor - Untitled [Request_For_Quotation]

Outbound Business Object Name: Input1 (Alliance Library.Quotation.1)

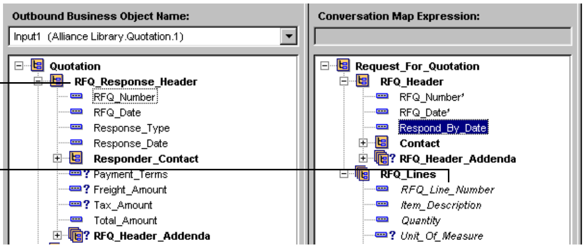
Conversation Map Expression:

Group icon

Field icon

The Conversation editor uses the same icons and graphical cues that appear in the Map editor.

These are the font cues in the outbound and inbound panels.



```
Outbound Business Object Name:
Input1  (Alliance Library.Quotation.1)

Quotation
  RFQ_Response_Header
    RFQ_Number
    RFQ_Date
    Response_Type
    Response_Date
    Responder_Contact
    ? Payment_Terms
    ? Freight_Amount
    ? Tax_Amount
    Total_Amount
    ? RFQ_Header_Addenda
```

```
Conversation Map Expression:

Request_For_Quotation
  RFQ_Header
    RFQ_Number'
    RFQ_Date'
    Respond_By_Date
    Contact
    ? RFQ_Header_Addenda
  RFQ_Lines
    RFQ_Line_Number
    Item_Description
    Quantity
    ? Unit_Of_Measure
```

Bold text indicates a group.

Italics indicates elements in a repeatable group.

This table summarizes the icons that can appear in the Conversation editor.

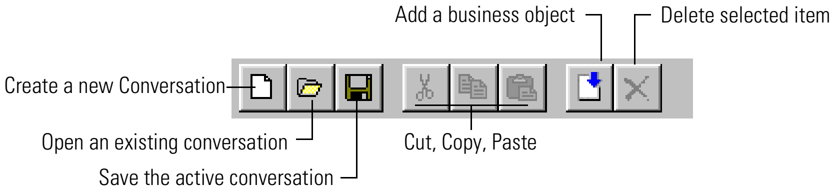
This icon	Represents	This icon	Represents
▭	A required field	▤	A required group
▭ ?	An optional field	▤ ?	An optional group
▭▭	A required field sequence*	▤▤	A required group sequence*
▭▭ ?	An optional field sequence*	▤▤ ?	An optional group sequence*

* This version of the Conversation editor does not allow you to use sequences to identify conversations.

Note: Unless they are specifically designated as optional (a question mark appears to the right of the icon), all elements are required. This means that any element without a question mark is required in its parent group. We recommend that you only use required fields to identify conversations.

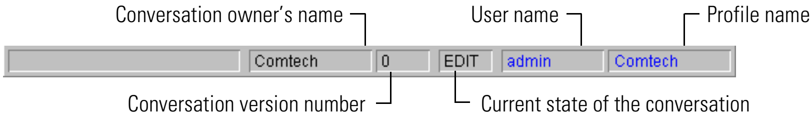
About the Command toolbar

The Command toolbar at the top of the Conversation editor gives you access to common File and Edit menu commands and lets you remove outbound business objects.


Add a business object
Delete selected item
Create a new Conversation
Open an existing conversation
Save the active conversation
Cut, Copy, Paste

About the status bar

The status bar at the bottom of the Conversation editor shows information about the current user and the state of the map.



Comtech | 0 | EDIT | admin | Comtech

Conversation owner's name
User name
Profile name
Conversation version number
Current state of the conversation

About the menu bar

The Conversation editor menu bar appears at the top of the window.

The File menu contains commands for creating, opening, saving, and closing conversations.

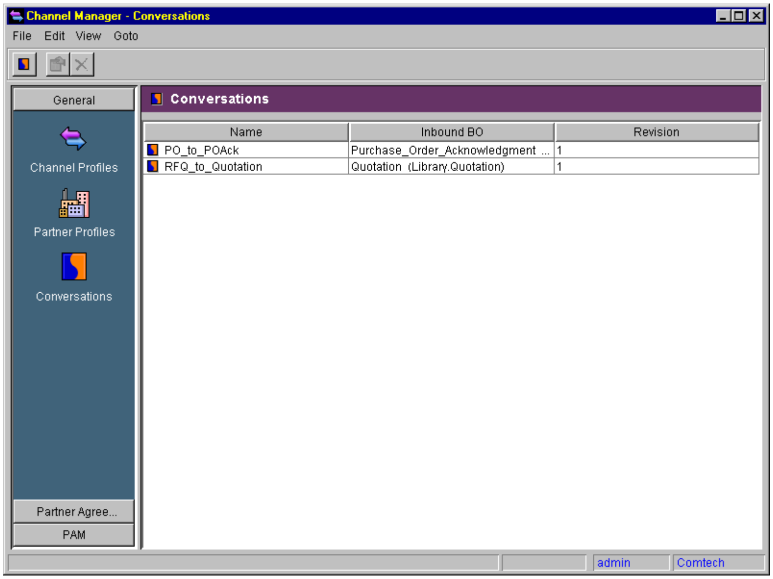
Creating conversations

The first step in creating a conversation is to select the inbound business object type. You then add outbound business object types as needed and establish associations between fields in the outbound business objects and the inbound business objects.

You create conversations in the Channel Manager. To launch the channel manager:

1. In the Process Manager window, click Actions and select Channel Manager from the Actions menu.

2 The Channel Manager window appears.



Name	Inbound BO	Revision
PO_to_POAck	Purchase_Order_Acknowledgment ...	1
RFQ_to_Quotation	Quotation (Library.Quotation)	1

To create a conversation:

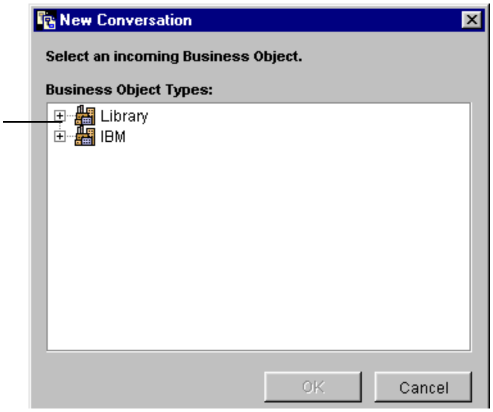
1 Click the New Conversation button in the Channel Manager's Command toolbar. This button is only enabled when you select Conversations in the panel on the left.

Or, you can click File on the Channel Manager menu and choose New>General>Conversation.

2 The Conversation Editor dialog appears. Click the Create a new conversation button from the command toolbar.

Or, you can click File on the Conversation Editor menu and choose New.

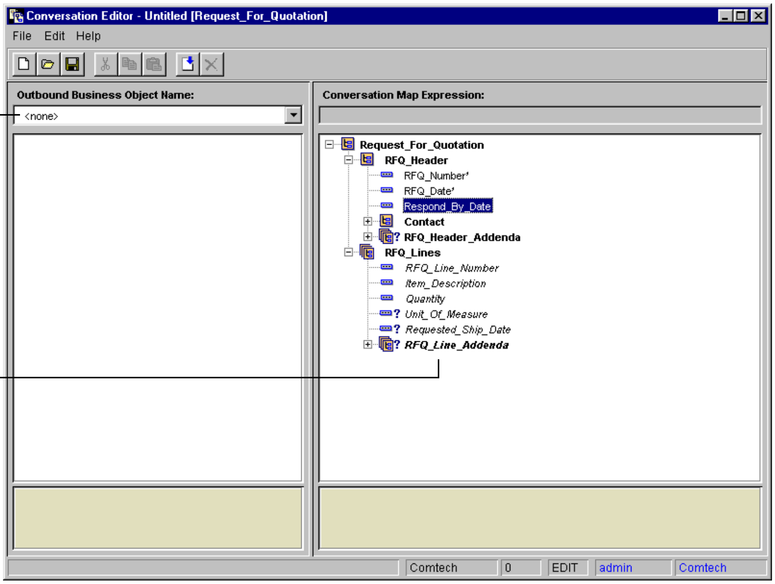
The New Conversation dialog box appears.



Select an inbound business object.

3 Select the inbound business object type that you want associate with this conversation and click OK.

The Conversation editor appears. The business object type you selected appears on the right side of the Conversation editor, in the output area.

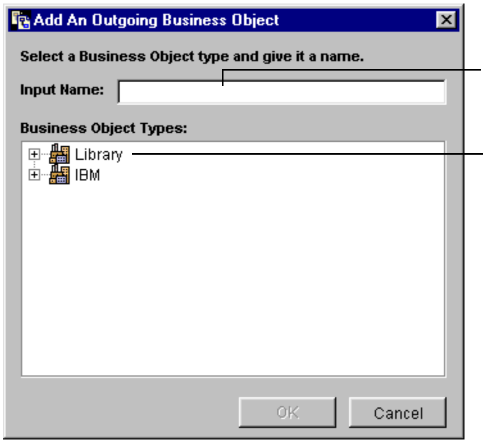


When you have multiple outbound business objects in a conversation, you can view them in this list.

This panel displays the inbound business object type. Double-click a group element to display its contents.

4 To add an outbound business object, click the Add business object button in the Conversation editor's Command toolbar.

Or, you can choose Add Business Object from the Edit menu. The Add Outbound Business Object dialog box appears.


Add An Outgoing Business Object
Select a Business Object type and give it a name.
Input Name:
Business Object Types:
Library
IBM
OK
Cancel

Type a name for the Output.

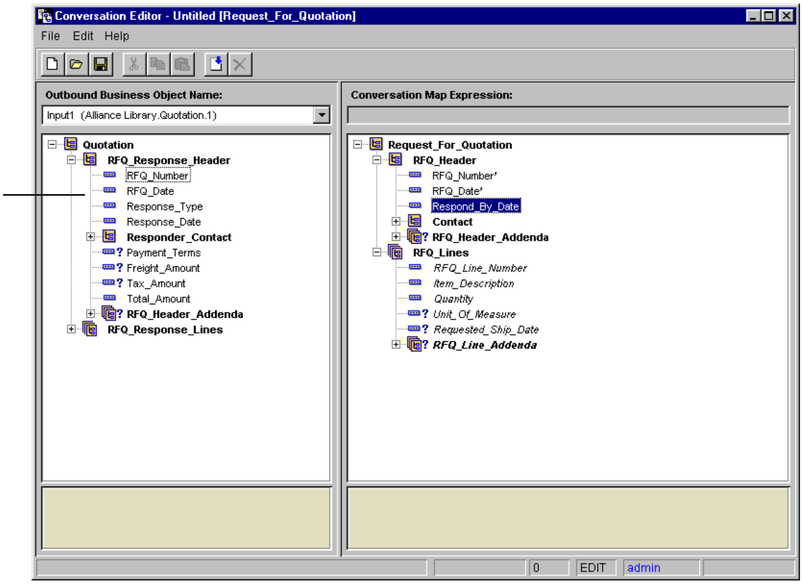
Select an outbound business object.

5 Type in a name for the Outbound business object type.

6 Select an outbound business object type for this conversation. Click OK.

The business object type you select appears in the input panel.

This panel displays the current outbound business object type. Double-click a group element to display its contents.



7 Expand the groups in the outbound and inbound business object types so that you can see their contents.

8 Associate one or more fields in the outbound business object with the inbound business object by clicking on a field in the outbound business object and dragging it to a field in the inbound business object. When you associate an inbound field, an asterisk (*) appears to the right of its name.

NOTE: In this release of the Conversation Editor, you cannot map sequences (repeatable elements).

You need to associate fields that will have identical content in both the outbound and inbound business object. Partner Agreement Manager uses these fields to identify the conversation.

The names of the fields can be different; the important thing is that the data contained in the fields is identical.

NOTE: We recommend that you always use required fields. If you use an optional field and the field is not present in either the outbound or inbound business object instance, Partner Agreement Manager will not be able to complete the conversation.

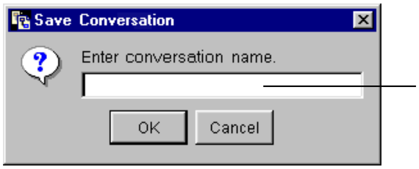
SAVING A CONVERSATION

When you save a conversation, you give it a name; the Channel Manager lists it in the Conversations folder.

To save a conversation:

1 Choose Save from the File menu.

The Save Conversation dialog box appears.



Type a name and click OK.

2 Type a name and click OK.

The conversation appears in the Conversations folder in the Channel Manager.

CHAPTER 6

USING THE CHANNEL API

Read this chapter for a description of the key concepts of the Channel API and examples of how to use the basic features of the Channel API. Sections in this chapter include:

- *Compiling your servlets* on page 60.
- *About Partner Agreement View communication* on page 61.
- *About Partner Agreement View queues* on page 61.
- *About business objects in Partner Agreement View* on page 71.
- *About web.xml* on page 78.

All example code used here is in the sample servlets. These example servlets can be found in:

On this platform:	the files are in:
Windows NT	`<tomcat>\webapps\WebSphere\PAV\WEB-INF\classes\appchannel\sample`
UNIX	`<tomcat>/jakarta-tomcat-3.2.1/webapps/WebSphere/PAV/WEB-INF/classes/appchannel/sample`

Compiling your servlets

Before compiling your servlets, you must perform certain tasks associated with your build environment. The specific tasks depend on whether you are using Windows NT or UNIX.

Compiling on Windows NT

Before compling your custom servlets on Windows NT, edit and then run the appchannelEnv.bat file. This file helps to define your environment (such as JAVA_HOME, path and classpath). In particular, you should change these lines:

```
SET JAVASDK= <JAVA home directory> (for example: c:\jdk1.2.2)
SET WEBSERVER_ROOT=<webserver-root directory> (for example: c:\tomcat)
SET DOCUMENT_ROOT=<document root relative to webserver-root directory>
(for example: \webapps\Extricity\AppChannel)
```

Compiling on UNIX

Before compiling your custom servlets on UNIX, you must ensure that your PATH and CLASSPATH environment variables include your JDK and Partner Agreement View installation.

The following commands illustrate how you can set these variables:

```
export JAVASDK=<full path to jdk home directory>
export WEBSERVER_ROOT=<full path to Tomcat-Jakarta home directory>
export DOCUMENT_ROOT=webapps\WebSphere\PAV
export SERVLETS_ROOT=$WEBSERVER_ROOT/$DOCUMENT_ROOT/WEB-INF/
classes
export CLASSPATH=$JAVASDK/lib/tools.jar:$WEBSERVER_ROOT/lib/
servlet.jar:$WEBSERVER_ROOT/lib/xerces.jar:$WEBSERVER_ROOT/lib/
xml4j.jar:$WEBSERVER_ROOT/lib/mail.jar:$WEBSERVER_ROOT/lib/
activation.jar:$WEBSERVER_ROOT/lib/ibmjsse.jar:$WEBSERVER_ROOT/lib/
AppChannel.jar:$SERVLETS_ROOT:.:$CLASSPATH
export PATH=$JAVASDK\bin:$PATH
```

After setting the PATH and CLASSPATH variables, change the present working directory to your servlets root directory ($WEBSERVER_ROOT/$DOCUMENT_ROOT/WEB-INF/classes). From here, you may compile your servlets.

About Partner Agreement View communication

A key feature of Partner Agreement View is that it provides the infrastructure to allow your Partner Agreement View applications to communicate with Partner Agreement Manager. This communication is not symmetric. Your Partner Agreement View application sends information to Partner Agreement Manager in an synchronous fashion. Partner Agreement Manager sends information to your Partner Agreement View application asynchronously using Partner Agreement View queues. You can also implement the event listener interfaces to allow Partner Agreement Manager to communicate with your Partner Agreement View application on an event-driven basis.

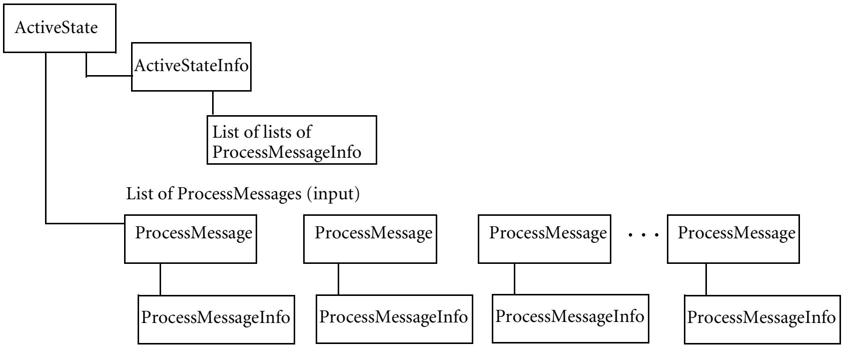
About Partner Agreement View queues

Partner Agreement View queues are the conduits of information from Partner Agreement Manager to your Partner Agreement View application.

About the active state

Each element in a Partner Agreement View queue is called an active state object. Each of these active state objects contains information about a Partner Agreement View node in a Partner Agreement Manager public process. This information is everything your Partner Agreement View application needs to know from Partner Agreement Manager in order to complete that node in the Partner Agreement Manager process. The active state contains data as well as metadata describing the data, the process and the active state itself.

Conceptually, the active state looks like this:



At the top level, the active state contains three things:

- data about this particular instance of the active state such as the active state ID and the process ID.
- an active state info object.
- a list of process messages. Each process message describes one input business object. These are objects that have been sent from Partner Agreement Manager to the Partner Agreement View application. There is one process message for each input business object. Each process message has an associated process message info (PMI) object.

The active state info object contains:

- metadata about the process (process definition ID, the label on this node of the process as it is visible in the Process Designer, Partner ID).
- a list of lists of PMI objects.

  The list of lists of PMI objects is a list of valid response sets that your Partner Agreement View application can return to Partner Agreement Manager to successfully complete this node in the Partner Agreement Manager process. Any one of the lists of PMIs constitutes a completion set for this node in the process.

For example, in response to a purchase order change request there are three possible response sets:

- a purchase order and a sales order
- an invoice and a change order
- a quote

Each of these response sets is a valid response to a purchase order change request. Your Partner Agreement View application would have to send one and only one of these valid response sets back to Partner Agreement Manager to continue the process.

Another way to think of the active state information is in terms of data and metadata:

Data or metadata	about the	specific data
data	input instance	input paths input business objects
	process instance	process ID active state ID
metadata	process	process definition ID partner ID node label
	output	output paths output business object types

The input instance data tells you the input paths to this node in the process and the input business objects. For each input business object, the active state also has a ProcessMessage object that describes the input business object. Associated with each ProcessMessage is a ProcessMessageInfo (PMI) object, which contains methods to access metadata about the process message and input business object.

The process instance data defines this particular active state and indicates which process it is part of. The process ID allows you to select from the queue an active state that is associated with a particular process instance. The active state ID uniquely identifies this active state object instance.

The process metadata tells you a bit more about the process. The active state is associated with one and only one node in a Partner Agreement Manager process. The node label is the label you see in the Process Designer for this node in the Partner Agreement Manager process. The partner ID allows you to select from the queue an active state that is sent to a particular Partner Agreement View Partner ID.

The output metadata tells you what output is expected from this node in the process and what the process output paths are. You can use the output business object types to generate the correct output business objects. For more information, see the Javadocs.

Per-partner vs. shared queues

When you install the Partner Agreement View, the installer configures one of two types of active state queues:

- Shared queues allow all partners to access active state objects on a single queue.
- Per-partner queues allow each partner to view only the active state objects intended for it. This is more secure and more efficient. It is the default for Partner Agreement View, because usually the Channel API is accessed directly by a number of different partners.

For information about configuring the queues, see *Configuring per-partner or shared queues* on page 129.

Getting a queue

Before you perform any queue operations, you must first get an instance of that queue. The entire queue structure is a singleton, so each servlet can simply get an instance of the queue. In the samples, persistent session information including queue information is stored in the `session_data` hash table. The storage of persistent data in the `session_data` structure is standard servlet and server-side application practice. For the purposes of the samples, the `session_data` structure contains a pointer to the queue (the queue field) and a pointer to the current active state in the queue. For more information on using the `session_data` hash table, see reference material on Sun’s site or any book covering servlet development.

For example, the sample GetNotificationServlet gets an instance of the queue, using the channelID. Note that different forms of the **getInstance**() method are required for shared or per-partner queues:

```
// get the queue instance
  AppChannel queue = null;
  if (AppChannelConfig.get().isSharedQueue(channelID))
  {
    queue =  AppChannelFactory.getInstance(channelID, password);
  }
  else
  {
    queue =  AppChannelFactory.getInstance(channelID, partnerID, password);
  }
```

The channelID comes from the configuration data file web.xml. For more on web.xml, see *About web.xml* on page 78.

Getting active states from the queue

Once you have an instance of the queue, you can get active states sent by Partner Agreement Manager. However, the queue could contain many active states. When you get something from the queue, you want to make sure you are getting the active state you are expecting, from the potentially large number of active states in the queue.

To only get the active states you want, you must indicate the constraints that active states must conform to. Examples of common constraints are:

- The active state must be associated with a particular process (i.e., have a given process ID)
- The active state must be from a particular partner (i.e., have a given partner ID)
- The active state must be a reply to a particular request you made. When you send an active state to Partner Agreement Manager, that active state is assigned a conversation ID. You can link the reply with the original request using the conversation ID.

The simplest way to get a particular active state is to use the **getActiveState** method. This method is overloaded and can be used in four ways:

- **getActiveState()** No arguments causes **getActiveState** to return the first available active state posted to you by Partner Agreement Manager.
- **getActiveState(ActiveStateId asid)** Invoking **getActiveState** with an active state ID causes it to return the active state that contains that active state ID.
- **getActiveState(integer i)** Invoking **getActiveState** with an integer argument causes it to return the i'th available active state posted to you by Partner Agreement Manager. For example, if the integer you used was 2, it would return the second available active state in the queue.
- **getActiveState(string id)** Invoking **getActiveState** with an ID string causes it to return the active state that contains the given ID. The ID string is either a conversation ID or a process ID. The conversation ID links responses with requests and is obtained from the process ID. The process ID is returned by **deliverInitialMessage** or **deliverResponseMessage**.

For example, the following code snippet shows how to get an active state with a particular conversation ID (or process ID). This code is from the **GetReplyServlet** sample.

First, the servlet gets a queue:

```
// get an Instance of the queue
   AppChannel queue = null;
   if (AppChannelConfig.get().isSharedQueue(channelID))
   {
      // use the factory for Shared queue mode
      int channelID = 1234;
      queue = AppChannelFactory.getInstance(channelID,
                 "channel1234_password");
   } else {
      long partnerID = 777;
      int channelID = 1102;
      // use the factory for per partner queue mode
      queue = AppChannelFactory.getInstance(channelID,
                 partnerID,
                 "partner1102_password");
   }
```

Then it initializes the active state variable and if the conversation ID isn't null, it calls getActiveState with that conversation ID:

```
ActiveState replyActiveState = null;
if (conversationID != null)
{
   System.out.println(
   "attempting to get the
    ActiveState for conversationID == " + conversationID);
    replyActiveState = queue.getActiveState(conversationID);
}
```

This is the simplest way to get a particular active state from the queue.

Using CQConstraints to get an active state

If your constraints are more complex, you can use a **CQConstraints** structure to define the constraints. Once you've defined your particular set of constraints, you can use that object to get the conforming active states from the queue. The following example is from **GetWithConstraintsServlet**:

The servlet first gets an instance of the queue:

```
// get an Instance of the queue
   AppChannel queue = null;
   if (AppChannelConfig.get().isSharedQueue(channelID))
   {
      // use the factory for Shared queue mode
      int channelID = 1234;
      queue = AppChannelFactory.getInstance(channelID,
                "channel1234_password");
   } else {
      long partnerID = 777;
      int channelID = 1102;
      // use the factory for per partner queue mode
      queue = AppChannelFactory.getInstance(channelID,
                 partnerID,
                 "partner1102_password");
   }
```

Then the servlet instantiates a CQConstraints object and builds the constraints:

```
// BUILD CONSTRAINTS
CQConstraints cqc = new CQConstraints();
```

The first constraint is the partner ID. The servlet sets the PartnerID to the partner ID from session_data.

```
// PartnerID
String strPartnerID = (String)session_data.get("partner.id");
if (strPartnerID != null)
{
   cqc.setPartnerID(Long.parseLong(strPartnerID));
}
```

The next constraint is the label. So you want to get only the active states that are associated with a process node that has a particular label. Recall that this label is the same label that you see next to the node in the Process Designer. Again, the servlet is getting this label from the session_data structure.

```
// Label
String strLabel=(String) session_data.get("label_CONSTRAINT");
if (strLabel != null)
{
   cqc.setLabel(strLabel);
}
```

The last constraint the servlet sets is the process definition ID.

```
// Process Definition ID
String strProcDefID = (String) session_data.get(
   "processName_CONSTRAINT");
if (strProcDefID != null)
{
   cqc.setProcessDefID(strProcDefID);
}
```

Now that the constraints are done, create an iterator to get all the active states. Then get all the matching active states.

```
// fetch all avaliable active states that match
Iterator iterActiveStates = queue.activeStates(cqc);
```

Using event listeners to get active states

Using getActiveState (or getActiveStates with an iterator) relies on the servlet to poll the queue, looking for active states. To use an event-driven approach, you need to use the NAEngineListener methods of the AppChannelQueue object. The NAEngineListener interface is in com.extricity.channels.api. Implement this interface and call AppChannelQueue.addNAEngineListener with your object to have your application listen on the queue for incoming active states.

In order to use a FileSystemEventListener, you must have conversation threading turned on. Conversation threading is turned off by default in Partner Agreement View. See *Setting the conversation aware mode* on page 133 for more information.

Sending messages to Partner Agreement Manager

Once you have an instance of the queue, you can send messages to Partner Agreement Manager. There are four kinds of messages your Partner Agreement View application can send to Partner Agreement Manager: initial, response, abort and terminal. You send an initial message when your Partner Agreement View application is the first node in the process. A response is a response to a request sent from Partner Agreement Manager. An abort message tells Partner Agreement Manager to abort the process. You should use this very rarely. You send a terminal message when your Partner Agreement View application is the final node in the process.

The PostNotification sample posts an initial message and does not expect a response. It already has an instance of the queue. Then it uses the deliverInitialMessage method of the queue object to send a message containing a partner ID and a business object:

```
long partnerID = 1102;
long partnerID = 777;
// first construct the BusinessObject  in variable bo
// Shared Queue vs. Per-Partner Queue use a different API
// ProcessAware vs. Process unaware also use a different API

if (!AppChannelConfig.get().isProcessAware(queue.getChannelID()))
{
  if (AppChannelConfig.get().isSharedQueue(channelID))
  {
    conversation_id = queue.deliverMessage(partnerID,
                bo,
                null);
  }
  else
  {
    conversation_id = queue.deliverMessage(bo,
                null);
  }
}

else
{
  if (AppChannelConfig.get().isSharedQueue(channelID))
  {
    conversation_id = queue.deliverInitialMessage(partnerID,
                null, // process_def_id
                null, // path
                AppChannel.EXEC_MODE_EITHER,//exec_mode
                bo,
                null);
  }
  else
  {
    conversation_id = queue.deliverInitialMessage(
                null, // process_def_id
                null, // path
                AppChannel.EXEC_MODE_EITHER,//exec_mode
                bo,
                null);
  }
}
```

The conversation ID is used to link together requests and replies. It is not used further in this example as there is no expected reply. However in the general case, it is useful for keeping track of a series of requests and replies. If there were an expected reply, you would call AppChannelQueue.getActiveState (conversation_id) to get the expected reply from the queue.

Note that the deliverInitialMessage method is synchronous. It returns immediately. If it fails, it throws an exception indicating why it failed. Likewise, deliverResponseMessage, deliverAbortMessage and deliverTerminalMessage are all synchronous.

About business objects in Partner Agreement View

The Partner Agreement View packages com.extricity.document.apps.bo and com.extricity.document.defs.dom allow you to manipulate your business objects, display them in HTML format and also generate HTML forms based on your business object types.

Creating business objects

If you have a request for which you need to generate a valid response, you'll need to create the appropriate business objects and populate them. To do this, you first get an instance of the BusinessObject factory:

```
// example BOTypeID
bo_type_id = "Invoice!2.Invoice.2";

// Create an instance of WebAppBOFactory
WebAppBOFactory bean = new WebAppBOFactory();

// Create BO
BusinessObject bo = bean.instanceFromDTD(new BOTypeID(bo_type_id));

// Use the PAM defined BO API for setting and getting elements
// and fields
bo.getRootElement().setData("InvoiceDate","1/1/2000 4:00pm"));
```

Now you have a new BusinessObject, ready to populate.

Displaying business objects

The com.extricity.web.document.apps.bo package contains classes to allow you to display your business objects in HTML. The Channel API methods that display business objects use an object called a BusinessObject. A BusinessObject implements BusinessObject. This means that a BusinessObject contains methods that allow you to manipulate and inspect its contents using the Partner Agreement Manager business object API, just like any other Partner Agreement Manager business object. A BusinessObject also has methods that allow you to use the printer objects in the com.extricity.web.bo.printer package to display the BusinessObject contents in HTML format. You can think of the BusinessObject object as a superset of the standard Partner Agreement Manager BusinessObject. For more on the Partner Agreement Manager BO API, see the *Partner Agreement Manager Adapter Developer's Guide*.

Use the WebAppBOFactory class to generate a BusinessObject from a business object type. If you are constructing a response to an request, you can get the correct response business object type from the ActiveStateInfo associated with the request. ActiveStateInfo.getResponseMessageInfos returns a list of lists of ProcessMessageInfo objects that define all valid response sets.

Analogous to the BusinessObject, the com.extricity.document.defs.dom package includes classes called ElementImp and ElementSequenceImp. This is the same implementation of BusinessObject as in Partner Agreement Manager.

Generating HTML forms based on business objects

The com.extricity.web.document.apps.bo package contains classes that you can use to make HTML forms from your business objects. The Text methods (such as TextPrinter) are all used for displaying the business object as a read-only page. The Form methods (such as FormPrinter) are used for creating HTML forms based on the business object types.

For example, PostRequestServlet uses FormHandler to display a BusinessObject in an HTML form.

First get a queue:

```
// get the queue instance
     AppChannel queue = null;
     if (AppChannelConfig.get().isSharedQueue(channelID))
     {
    //*** HARD CODED PASSWORD FOR TEST PURPOSES...
    //***  CHANGE THIS TO YOUR OWN PASSWORD!!
    queue =  AppChannelFactory.getInstance(channelID,
          __HARD_CODED_TEST_CHANNEL_PASSWORD__);
     }
     else
     {
    //*** HARD CODED PASSWORD FOR TEST PURPOSES...
    //***  CHANGE THIS TO YOUR OWN PASSWORD!!
    queue =  AppChannelFactory.getInstance(channelID,
                   partnerID,
          __HARD_CODED_TEST_PARTNER_PASSWORD__);
     }
```

Then create a BusinessObject from the business object type Partner Agreement Manager expects to receive in response:

```
// create an instance of a business object from the
//   business object type ID perscribed in the initialization property
   BusinessObject bo = bean.instanceFromDTD(bo_type_id);
```

Now, take the business object and put it into the session_data structure.

```
session_data.put("botypeid", bo_type_id);
```

Then create a printer and a BOWalker to display the business object:

```
BOTypeID bo_type_id = (BOTypeID) session_data.get("botypeid");
   BusinessObject bo = bean.instanceFromRequest(request);

PrintWriter form_printer = new PrintWriter( new OutputStreamWriter(
   response.getOutputStream(), "UTF-8"), true);
```

```
//create bo printer to print the BusinessObject
FormPrinter bo_printer = new FormPrinter(form_printer, bo);

//create OpenFormTag i.e. <Form>
bo_printer.openFormTag(null);
//print the BO in HTML
 BOWalker walker = new BOWalker(bo.getTypeID(), bo_printer);
 walker.traverse((org.w3c.dom.Node)bo.getRootElement());
//create submit button in HTML
 bo_printer.printSubmitButton();
//create Close form tag i.e. </Form>
 bo_printer.closeFormTag();
```

Mapping business objects

You can use maps to fill business objects fields with default values. For example, you receive a purchase order and need to reply to it. You will want to include the information from the purchase order in your reply. You can use a map to copy the information from the business object you receive to the one you will be sending in reply.

About map files

Map files have a specific format. An example map file is in:

On this platform:	the file is in:
Windows NT	<tomcat>\webapps\WebSphere\PAV\Web-inf\classes\appchannel\sample\ShipmentNotice.map
UNIX	<tomcat>/jakarta-tomcat-3.2.1/webapps/WebSphere/PAV/WEB-INF/classes/appchannel/sample/ShipmentNotice.map

The first part of the file is a long header of comments describing the format and contents of the file. Below that, there are the following sections:

- **size**—lists the maximum number of elements in repeating groups.
- **readonly**—indicates whether a value can be edited in a form.
- **visible**—indicates whether this field is visible in a form.

- **inputType**—indicates the input method in a form. SELECT indicates that the valid values for this field are subject to a constrained list. If you indicate an inputType of SELECT, you must specify the constrained list of valid values in the dataOptions section. If the valid values of a field are not subject to a constrained list, do not list an inputType for that field. SELECT is the only valid inputType.
- **dataOptions**—goes with inputType. If one or more fields have the inputType SELECT, dataOptions must be set for those fields indicating the values in the constrained list. These values are specified as a comma-delineated list of string values. The specified values will appear in a drop-down menu in a form that uses this map. See the dataOptions in example map file, next.
- **data**—lists the default values for fields.

The format of each field description is the same. Group names are listed in hierarchical order, separated by a forward slash (/). Following the group names is the field name followed by a dot (.). Next is the attribute (size, readOnly, visible, inputType, dataOptions or data) an equals sign and the value. For example, the first line of the .size section of ShipmentNotice.map looks like this:

```
Shipment_Header/Ship_To/Address.size=1
```

This indicates that in the repeating group Shipment_Header, there is a sub-group called Ship_To with a field called Address. The value 1 indicates that there can be only one Address field in Ship_To. That is, the size of the repeating field is 1.

The valid values for the field Shipment_Method are subject to a constrained list in the inputType section:

```
Shipment_Header/Shipment_Method.inputType = "SELECT"
```

Since there is an inputType set to SELECT, there is a corresponding dataOptions entry with the constrained list of valid values:

```
Shipment_Header/Shipment_Method.dataOptions = "UPS, FedEx, U.S.
Postal Service, U.S. Mail (Ground), DHL"
```

The last section contains the default values for fields. You need not list every field, only the ones you want to set default values for. For example, ShipmentNotice.map sets the description field of the second Shipment_Addenda sub-group in the Shipment_Header group:

```
Shipment_Header/Shipment_Addenda[1]/Description.data  = "Top
Secret"
```

The default values need not be hard-coded into the map file. You can use variables to indicate the values as well. To set the default value of a field to a variable you use in your code, use the same name you use in your code and preface it with a $character. For example, ShipmentNotice.map sets the default date to today:

```
Shipment_Header/Shipment_Date.data = $today
```

The dollar sign indicates that what follows is a variable in your servlet code.

Using a map

Once you have a map file, you can use it to fill in the default values for a business object. The following code snippets are from PostRequestServlet. They demonstrate how to use a map to fill in default values for a business object.

First, instantiate a new properties structure:

```
// set the business object map. default values using
//  a property value provider
Properties propsBOMap = new Properties();
```

Then open the map file as a stream and fill in the propsBOMap structure you just instantiated:

```
InputStream stream = this.getClass().getResourceAsStream(boMapName);
propsBOMap.load(stream);
stream.close();
```

Add the date to the properties file:

```
// add current date to the list of properties
propsBOMap.setProperty("today", new java.util.Date().toString());
```

Note that “today” is the name of the variable used in the map file.

Now that you have a complete map, instantiate a new business object with that map. The map will fill in the default information.

```
// if we are here properties have loaded successfully. Now
// all that remains is to initialize the business object ...
new PropertyValueProvider(propsBOMap).initialize(bo);
```

Note that you are using PropertyValueProvider to apply the map to the new business object. PropertyValueProvider is in the com.extricity.web.bo.util package. Note that the constructor for PropertyValueProvider is overloaded. The code snippet above uses PropertyValueProvider with a mapper as an argument. This initializes the business object with the values from the map. Using PropertyValueProvider with a map is useful when you are sending an initial message to Partner Agreement Manager.

You can use PropertyValueProvider with a source BusinessObject and a map as arguments. This will create a map from the source and initialize a business object with the values from that map:

```
new PropertyValueProvider(sourceBusinessObject, propsBOMap).initialize(bo);
```

Using PropertyValueProvider with a source business object is useful when you are sending a reply to Partner Agreement Manager.

If you are using a map file, that map file must be in the same directory as the servlet class file, otherwise your servlet will not be able to find the map.

Making a map

The com.extricity.document.apps.bo.MapWriter class is a utility class that takes a business object and generates a map from that business object.

To use the MapWriter to generate a map file from a business object:

1. Instantiate a FileWriter.
2. Create a MapWriter object, passing FileWriter and BOTypeID.
3. Create BOWalker to traverse the business object.
4. Invoke mapwriter.traverse method on the root element of the business object for which you want to create a map.

Now, all the values in the business object you used as a template are the default values in the map file. You may want to edit the map file and remove some of the default values generated by the MapWriter or set other options, such as inputType. To change the default values, change only the value within the quotes. If you do so, be careful as there is no syntax checking on map files.

For more on using map files, see the example servlet.

On this platform:	the file is in:
Windows NT	<tomcat>\webapps\WebSphere\PAV\Web-inf\classes\appchannel\sample\PostRequestServlet.java
UNIX	<tomcat>/jakarta-tomcat-3.2.1/webapps/WebSphere/PAV/WEB-INF/classes/appchannel/sample/PostRequestServlet.java

About web.xml

The file named web.xml is called a deployment descriptor, meaning that it contains configuration information about the Partner Agreement View application. It is an XML file, containing a DTD which is specified in detail in the Java Servlet API. See www.developer.java.sun.com for more details on the Java Servlet API. For the purposes of Partner Agreement View, the web.xml file translates the full path and name of servlets into something much more readable. For example, appchannel.sample.GetNotificationServlet becomes GetNotificationServlet. The web.xml file also specifies initialization parameters.

On this platform:	the file is in:
Windows NT	<tomcat>\webapps\WebSphere\PAV\Web-inf\web.xml
UNIX	<tomcat>/jakarta-tomcat-3.2.1/webapps/Extricity/PAV/WEB-INF/web.xml

The section of web.xml for GetNotificationServlet is:

```
<servlet>
```

```
<servlet-name>
   GetNotificationServlet
</servlet-name>
<servlet-class>
   appchannel.sample.GetNotificationServlet
</servlet-class>
<init-param>
   <param-name>
      appchannel.partner.id
   </param-name>
   <param-value>
      777
   </param-value>
</init-param>

<init-param>
   <param-name>
      channel.id
   </param-name>
   <param-value>
      1004
   </param-value>
</init-param>
<init-param>
   <param-name>
      business.object.type.id
   </param-name>
   <param-value>
      Invoice!2.Invoice.2
   </param-value>
</init-param>
<init-param>
   <param-name>
      web.admin.email
   </param-name>
   <param-value>
      foo@bar.com
   </param-value>
</init-param>
<init-param>
   <param-name>
      error.image.url
```

```
        </param-name>
        <param-value>
            ../images/execute.gif
        </param-value>
    </init-param>
</servlet>
```

The servlet-name tag indicates the short form of the servlet name and the servlet-class tag indicates the full path and name of the servlet. The init-param blocks list initialization parameters and their values.

You will want to use web.xml for your own Partner Agreement View applications to translate long names and specify the initial parameters for the first instantiation of the servlets.

CHAPTER

7

Sample public processes

Read this chapter to learn more about the sample processes that are provided with Partner Agreement View. The sample public processes are provided as an illustration of using Partner Agreement View in public processes.

Sections in this chapter include:

About the sample public processes

Partner Agreement View 2.2 supports Partner Agreement View partners in public processes. The sample public processes are included with the Partner Agreement View product to demonstrate how to use Partner Agreement View.

There are five sample public processes distributed with Partner Agreement View. The Partner Agreement View installer places the sample processes in your Partner Agreement Manager installation as XML files. You must import the processes under your partner name, as described in *Importing the sample public processes*, next. The sample processes imported are:

- SAMPLE_AppChannel_WAW_OSI_OSIR
- SAMPLE_AppChannel_AWA_Multi_BO
- SAMPLE_AppChannel_AW_Invoice
- SAMPLE_AppChannel_WA_ShipmentNotice
- SAMPLE_AppChannel_AWA_RFQ_Quotation

To use these samples, you must either have a partner ID of 777 and have configured the Partner Agreement View password (see page 86) or edit the web.xml file to change the partner ID and/or password (see *Sample Servlets* on page 86).

Importing the sample public processes

In order to use the sample public processes, you must first edit them and then import them. The steps for importing the sample processes are:

Step 1 Edit the XML files so that your partner name and partner ID owns the processes.

Step 2 Import the sample processes.

Step 3 Distribute the sample processes.

Editing the XML files

You need to perform minor edits on each of the sample process XML files so the processes are owned by your organization. To do this, you will need to know your Partner Agreement Manager Partner name, Partner Agreement Manager Partner ID, Partner Agreement View Partner name, and Partner Agreement View Partner ID.

The sample process XML files are located in `<PAM` *`root`*`>\Alliance\Exports\Processes`. The XML files are named:

- SAMPLE_AppChannel_WAW_OSI_OSIR.xml
- SAMPLE_AppChannel_AWA_Multi_BO.xml
- SAMPLE_AppChannel_AW_Invoice.xml
- SAMPLE_AppChannel_WA_ShipmentNotice.xml
- SAMPLE_AppChannel_AWA_RFQ_Quotation.xml

To edit the XML files:

1 Open one of the XML files in a text editor. You will need to replace four placeholders with your partner information. You can do this with a search and replace.

Replace this placeholder:	With:
__MASKED_ALLIANCE_PARTNER_0_NAME__	Your PAM partner name
__MASKED_ALLIANCE_PARTNER_0_ID__	Your PAM partner ID
__MASKED_APPCHANNEL_PARTNER_1_NAME__	A Partner Agreement View Partner name
__MASKED_APPCHANNEL_PARTNER_1_ID__	A Partner Agreement View Partner ID

2 Save the file. You may wish to save as another file name.

3 Repeat this process with the remaining XML files.

Now that you own the processes, you can import them.

Importing the sample processes

You must import the samples into your Partner Agreement Manager installation using the IBM Partner Agreement Manager window.

To import the sample processes:

1 Open the IBM Partner Agreement Manager window.

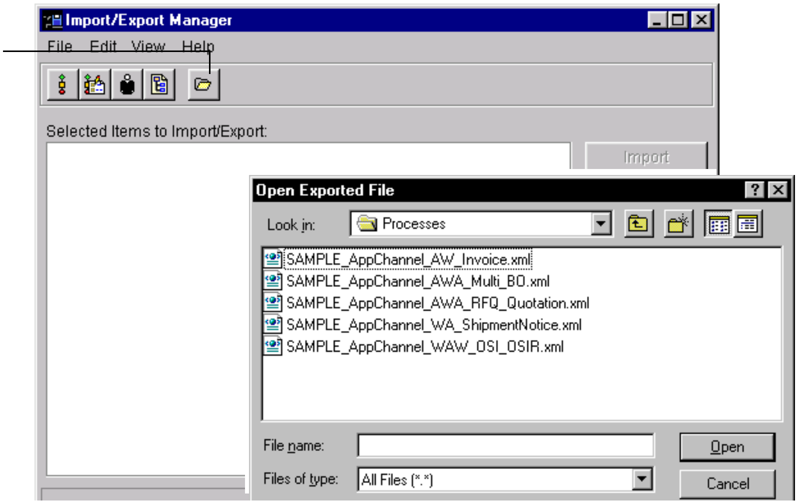
IMPORTANT: If your Process Server is running on UNIX, you must use the UNIX variant-compatible Partner Agreement Manager client.

2 Click the Import/Export Manager button on the Command Toolbar. Or, you can choose Import/Export Manager from the Tools menu. The Import/Export manager window appears.

3 Click on the Open XML File for Import button in the Command Toolbar. Or, you can choose Open for Import from the File menu.

The Open Exported File window appears. Browse to the Processes directory. The sample process XML files are located in `<PAM` *`root`*`>\Alliance\Exports\Processes`.

A list of all of the sample public processes appears.



Click this button to select the file to be imported.

Look in: Processes

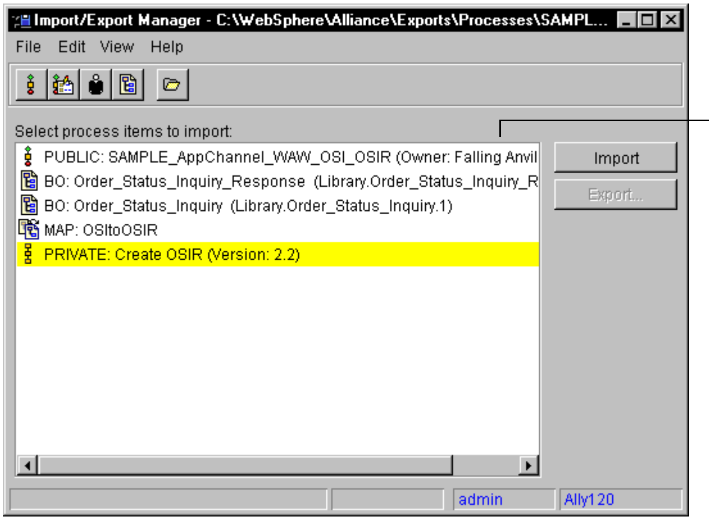
SAMPLE_AppChannel_AW_Invoice.xml
SAMPLE_AppChannel_AWA_Multi_BO.xml
SAMPLE_AppChannel_AWA_RFQ_Quotation.xml
SAMPLE_AppChannel_WA_ShipmentNotice.xml
SAMPLE_AppChannel_WAW_OSI_OSIR.xml

File name:

Files of type: All Files (*.*)

4 Select one of the sample process XML files and click Open.

You should see public processes as well as business objects in the Import dialog. In this example, you will notice that the partner name is Falling Anvil1. When you have edited the XML files and are importing the processes, you'll see your partner name as the owner of the process. If do not properly edit the XML file, you will receive an error message.


Import/Export Manager - C:\WebSphere\Alliance\Exports\Processes\SAMPL...
File
Edit
View
Help
Import
Export...

Select process items to import:

- PUBLIC: SAMPLE_AppChannel_WAW_OSI_OSIR (Owner: Falling Anvil
- BO: Order_Status_Inquiry_Response (Library.Order_Status_Inquiry_R
- BO: Order_Status_Inquiry (Library.Order_Status_Inquiry.1)
- MAP: OSItoOSIR
- PRIVATE: Create OSIR (Version: 2.2)

admin | Ally120

Your partner name appears here.

5 Click Import.

6 Click OK in the confirmation dialog box.

7 Repeat steps 3 through 6 to import the remaining sample XML files.

8 Close the Import/Export Manager when all remaining sample XML files have been imported.

Distributing the sample processes

To run the sample processes, you must distribute them. For more details on distributing processes, see the *Partner Agreement Manager User's Guide*.

To distribute the sample processes:

1 In the IBM Partner Agreement Manager window, right-click on one of the sample processes and select Process Distribution Manager to open the Process Distribution Manager window.

2 Select Review and click Apply.

3 Select Distribution and click Apply.

4 Select Approval and click Apply.

5 Select Test Installation and click Apply.

Sample Servlets

The Partner Agreement View distribution includes a number of sample servlets. You can access these servlets from:

<Channel API computer>:*<port>*:/WebSphere/PAV/jsp/home.jsp.

From this location you select the link to the Samples. The actual files are in:

On this platform:	the files are in:
Windows NT	<*tomcat*>\webapps\WebSphere\PAV\WEB-INF\classes\appchannel\sample
UNIX	<*tomcat*>/jakarta-tomcat-3.2.1/webapps/WebSphere/PAV/WEB-INF/classes/appchannel/sample

Before running the servlets, you must edit the web.xml file. This file is located in:

On this platform:	the files are in:
Windows NT	<tomcat>\webapps\Extricity\PAV\WEB-INF\web.xml
UNIX	<*tomcat*>/jakarta-tomcat-3.2.1/webapps/WebSphere/PAV/WEB-INF/web.xml

The default Partner Agreement View partner ID is 777. You need to replace this with the partner ID of the partner who will be using your Partner Agreement View-enabled Web server applications to participate in the Partner Agreement Manager process. The web.xml file is described in detail in *About web.xml* on page 78.

To run the sample servlets, you must add a password:

1 In the Administration folder of the Process Manager window, click on Passwords.

For Partner Agreement View (per partner queue mode) use,

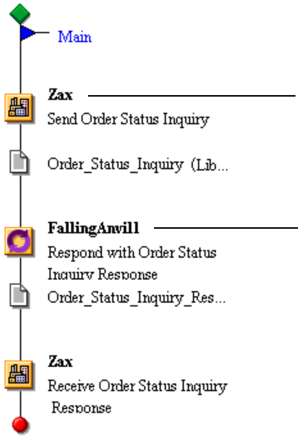
```
Login name: AppChannelInboundPartner<Instance-ID>
password: <partner_password>
```

Where *Instance-ID* is the partner ID (a numeric value). For example, if your partner ID is 777, the login name would be AppChannelInboundPartner777. The password is whatever text you choose; however to use the sample servlet, the password must be “partner_password”.

OSI_OSIR

This process is a two-way communication. The communication starts with Partner Agreement View, goes to Partner Agreement Manager and back to Partner Agreement View.

In these examples, the partner named FallingAnvil1 is the host partner. Recall that the host partner has Partner Agreement Manager installed and a Partner Agreement View interface. The partner named Zax is the Partner Agreement View partner using the PAV application (which uses the Channel API) to participate in the Partner Agreement Manager process. Since you edited the sample XML files to include your partner name and ID as the owner of the process, you will see your partner name in place of FallingAnvil1 in these sample processes. Likewise, you will see the name of the Partner Agreement View partner you specified in the XML files in place of the Zax name. The public process has these actions:


Main
Zax
Send Order Status Inquiry
Order_Status_Inquiry (Lib...
FallingAnvil1
Respond with Order Status Inquiry Response
Order_Status_Inquiry_Res...
Zax
Receive Order Status Inquiry Response

Your Partner Agreement View partner name appears here.

Your Partner Agreement Manager partner name appears here.

It performs these tasks:

1 In the first action, the Partner Agreement View partner creates and populates an Order_Status_Inquiry business object instance. The Partner Agreement View partner initiates the process by sending the Order_Status_Inquiry business object instance to the host partner.

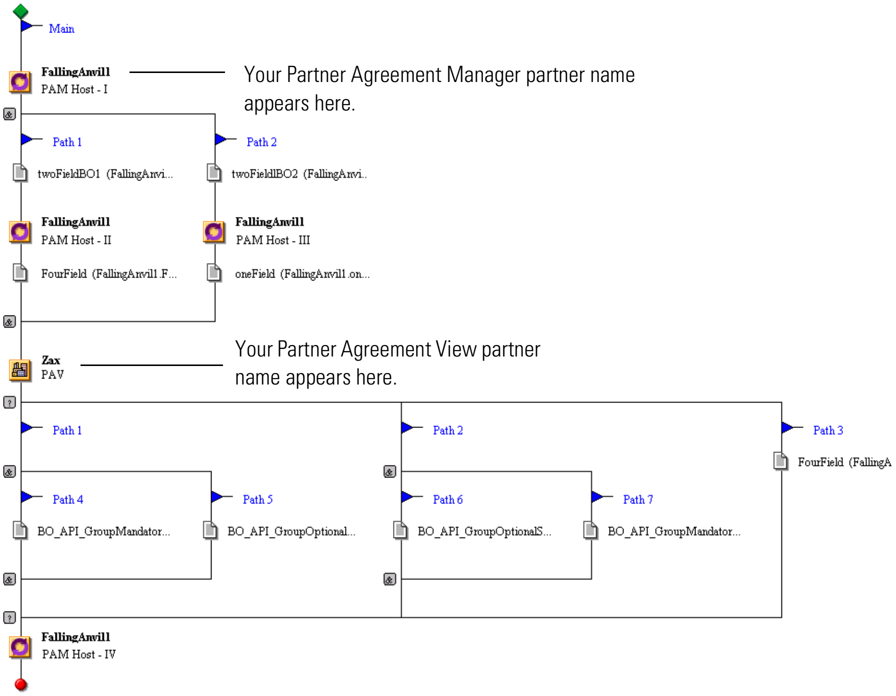
2 In the second action, the host partner receives the Order_Status_Inquiry, generates and populates an Order_Status_Inquiry_Response business object instance and sends it to the Partner Agreement View partner.

3 In the final action, the Partner Agreement View partner receives the Order_Status_Inquiry_Response business object instance, completing the process.

Multi_BO

This process demonstrates how to use multiple business objects and multiple paths in a public process using Partner Agreement View.

The public process has these actions:



Your Partner Agreement Manager partner name appears here.

Your Partner Agreement View partner name appears here.

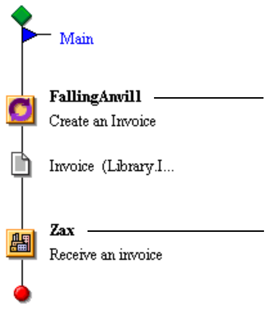
It performs this task:

- The host partner sends two business objects to the Partner Agreement View partner. The host partner expects, in return, one of several possible response sets.

Invoice

This process is a one-way communication. The host partner sends an Invoice business object instance. The Partner Agreement View partner receives the invoice business object instance.

The public process has these actions:



Your Partner Agreement Manager partner name appears here.

Your Partner Agreement View partner name appears here.

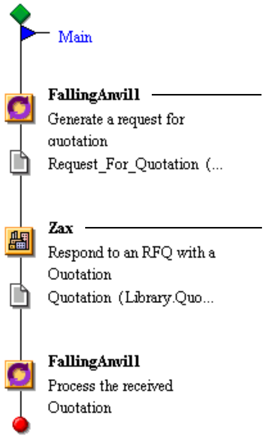
It performs these tasks:

1. In the first action, the host partner creates and populates an Invoice business object instance. This Invoice business object instance is sent to Zax.
2. The Zax partner receives the invoice.

RFQ_Quotation

This process is a two-way communication. The host partner sends a business object instance of the type Request_For_Quotation to the Partner Agreement View partner. The process blocks until it receives a Quotation business object in response. The Partner Agreement View partner via the Web server application, sends a Quotation business object to the host partner. The host partner receives the Quotation business object and processes it, completing the process.

The public process has these actions:


Main
FallingAnwill
Generate a request for ouotation
Request_For_Quotation (...
Zax
Respond to an RFQ with a Ouotation
Quotation (Library.Quo...
FallingAnwill
Process the received Ouotation

Your Partner Agreement Manager partner name appears here.

Your Partner Agreement View partner name appears here.

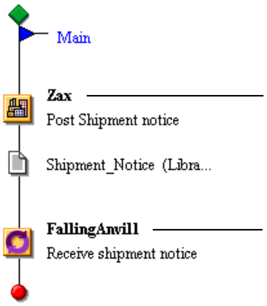
It performs these tasks:

1. In the first action, the host partner creates and populates a Request_for_Quotation business object instance. This business object instance is sent to Zax via the Web server application.
2. In the second action, Zax receives the Request_for_Quotation business object instance and replies with a Quotation business object instance.
3. In the third action, the host partner completes the process by receiving the Quotation business object instance and processing it.

Shipment_Notice

This process is a one-way communication. The Partner Agreement View partner using the Web server application, Zax, sends a shipment notice to the host partner. The host partner then receives the shipment notice.

The public process has these actions:



Your Partner Agreement View partner name appears here.

Your Partner Agreement Manager partner name appears here.

It performs these tasks:

1. In the first action, the Zax partner uses the PAV application to create and populate a Shipment_Notice business object instance.
2. In the second action, the host partner receives the Shipment_Notice business object instance.

APPENDIX

A

Windows NT Quick Install Instructions

These instructions help you get up and running quickly with Partner Agreement View on Windows NT. They assume that you have Partner Agreement Manager up and running. Furthermore, the instructions assume that you are performing a default installation. For detailed information on installing Partner Agreement View, see *Installing WebSphere Partner Agreement View* on page 9.

Sections in this appendix include:

- *Installing Partner Agreement View* on page 96.
- *Configuring Partner Agreement View* on page 97.
- *Running the samples* on page 98.
- *Using the WebProxy* on page 99.

Installing Partner Agreement View

The instructions in this appendix assume that you have three computers: one running Partner Agreement Manager (named Manager), one running the Tomcat Web server (named WebServ), and one running the WebProxy (named WebProx).

To install Partner Agreement View and related software:

1. From the installation CD, run PAV\PAMSide\setup.bat on the machine where you have installed Partner Agreement Manager (Manager).
   - A Accept the defaults.
   - B Say Yes to All when prompted about overwriting existing files.
2. Install the Tomcat web server on a second machine (WebServ) by unzipping Webservers\Tomcat\jakarta-tomcat-3.2.1.zip (on the CD) to your C:\ drive. It will create a jakarta-tomcat-3.2.1 directory with subdirectories. Rename the jakarta-tomcat-3.2.1 directory to tomcat.
3. From the CD, run Partner_Agreement_View\WebServerSide\setup.bat on the machine where you have installed Tomcat (WebServ).
   - A For the Partner Agreement Manager Partner ID, enter the ID for the Partner Agreement Manager installation you have on Manager. (Do not confuse this with the Partner Agreement View installation you just did.)
   - B For the Partner Agreement Manager Host, enter the machine on which Partner Agreement Manager is installed (Manager)
   - C For the Partner Agreement Manager port, enter 80, since that is the port that the IBM HTTP Server (on Manager) listens to by default.

Note that these properties get written to the AppChannel.properties file, located in:

<*tomcat*>\webapps\WebSphere\PAV\conf\AppChannel.properties .

Configuring Partner Agreement View

To configure Partner Agreement View:

1. Copy the contents of Webservers\IBMHttpServer\conf\tomcat.conf from the CD and append it to httpd.conf for the IBM HTTP Server on WebServ.
2. From the CD, copy Webservers\Tomcat\redirector\binary\Apache1.3\Win32-intel\mod_jk.dll to the modules directory of your IBM HTTP Server installation.

IMPORTANT: Be sure to restart the HTTP Server so that the changes take effect.

3. On WebServ go to c:\tomcat\bin\tomcat.bat.

   Under set _CLASSPATH, add the line:

   `set JAVA_HOME=<`*path to your jdk*`>`.

   The documentation contains more info on setting environment variables.
4. Now bring up the Partner Agreement Manager Server and Partner Agreement Manager Client on Manager. From the Partner Agreement Manager Process Manager window do the following:

   A. Open the Partner Agreement View 1001 channel profile from the Channels folder of the Process Manager window. Edit the outbound URL to be http://WebServ.<*domain*>.com/WebSphere/PAV/servlet/AppChannelPOBox, where domain is the Web Server where you installed the Partner Agreement Manager Process Server .

   Note that the outbound URL is case-sensitive.

   B. With the new partner toolbar button, create a new partner (such as PAVPartner) on your Partner Agreement View 1001 channel. Give it partner id 777 (in order to run the sample servlets).

   C. Under the Administration folder, open the Passwords. Add the following password

   ```
   Login AppChannelInboundPartner777
   Password partner_password
   ```

D Edit the processes (found under *<your PAM installation>*\Alliance\Exports\Processes) for your Partner Agreement Manager partner's name and ID and the new Partner Agreement View partner's name and ID (PAV Partner, 777). Import and install these processes.

Running the samples

From a browser check that the Web servers are working correctly as follows:

1 From a browser, connect to the URL: http://WebServ.*<domain>*.com to see that the IBM HTTP Server is running.

2 Start Tomcat by navigating to the c:\tomcat directory from a DOS shell and then running the command:

```
.\bin\tomcat start
```

3 Using the browser, connect to the URL: http://WebServ.*<domain>*.com:8080. You should see the Tomcat page.

4 Now check that the redirection from the IBM HTTP Server to tomcat is working correctly by connecting to http://WebServ/WebSphere/PAV/jsp/home.jsp.   If this works, everything is OK and you can continue.

5 Make sure the Partner Agreement Manager Server is running and try to log in to the Active State Inbox as follows.

A From http://WebServ.*<domain>*.com:8080/Extricity/PAV/jsp/home.jsp and go to Samples> Queue Samples. You will see a page with several servlets, starting with a Hello World servlet. Click on the Active States Inbox. You should see a login screen. Respond to the prompts as shown here:

```
Channel ID:   1001
Partner ID:   777
Password:   partner_password
```

B You should see the Active State Inbox appear. If it does not, look closely at the error message to determine the source of the problem.

c If you logged in successfully, you can start trying out the samples. There are two ways to try out the samples processes. One is through the Active State Inbox; the other is through the sample servlets (same web page as Hello World Servlet and Active State Inbox servlet.)

- The Partner Agreement Manager-initiated processes must be started from the Partner Agreement Manager client. The PAV Partner can respond either via the Active State Inbox or via the corresponding sample servlet.

- PAV Partner-initiated processes can be started either through the Active States Inbox or through the corresponding sample servlets.

Using the WebProxy

To use the WebProxy, you must run the webproxy executable, configure Partner Agreement Manager via the partner.properties file, and configure Partner Agreement View via AppChannel.properties.

Note: If you are using UNIX, you'll have to compile the webproxy code yourself before running the executable.

1 The computer where the webproxy will run is "WebProx".

A On WebProx, create a directory c:\webproxy.

B From the CD, unzip the WebProxy\Webproxy.zip file to the c:\webproxy directory.

C From c:\webproxy\release, run the command:

```
webproxy.exe 6500 6600 6700
```

2 On Manager, in <*your PAM installation*>\Partners\PartnerXXX\Properties\Partner.properties, append the following lines:

```
WebProxy.HTTP.Host=<WebProx's IP address>; mode=server;
   type=string;
WebProxy.HTTP.ControlPort=6500; mode=server; type=int;
WebProxy.HTTP.DataPort=6600; mode=server; type=int;
```

3 On WebServ open the AppChannel.properties file, located in <*tomcat-root*>\webapps\WebSphere\PAV\conf.

A Edit alliance.host.<*channelID*> to be WebProx

B Edit alliance.port.<*channelID*> to be to be 6700.

NOTE: The WebProxy.HTTP.ControlPort used in step 2 (6500 in this example) must match the first parameter in the `webproxy.exe` command in step 1C; the WebProxy.HTTP.Data port used in step 2 (6600) must match the second parameter in the `webproxy.exe` command; and the Partner Agreement Manager Port used in step 3B (6700) must match the third parameter in step 1C. For further information about the WebProxy, please see the WebProxy appendix of *Partner Agreement Manager Administrator's Guide*.

APPENDIX

B

Upgrading Partner Agreement View

This appendix describes how to upgrade Partner Agreement View from version 1.1 to version 2.2. Sections in this appendix include:

- *Software version comparisons* on page 102.
- *Upgrade the Channel Interface computer* on page 102.
- *Upgrade the Channel API computer* on page 103.

These instructions assume that you have already installed Partner Agreement Manager 2.2.

Software version comparisons

The upgrade from Partner Agreement View version 1.1 to version 2.2 includes a number of new or changed versions of software. This table lists the various components and compares the version numbers between 1.1 and 2.2. For a figure that illustrates how these components work together, see *Installing WebSphere Partner Agreement View* on page 9.

Software	version for 1.1	version for 2.2
Partner Agreement Manager	1.1	2.2
Apache Tomcat	3.0	3.2.1
Partner Agreement View	1.1	2.2
IBM HttpServer	1.3.11.1	1.3.12.2
WebProxy	1.0	1.1

Upgrade the Channel Interface computer

To upgrade the Channel Interface computer:

1. Upgrade the IBM HttpServer to 1.3.12.2.
2. Run Partner Agreement View 2.2 Channel Interface installer on the Partner Agreement Manager 2.2 computer (this assumes that you’ve already upgraded to Partner Agreement Manager 2.2).
3. After upgrading Channel API, update the per-channel property Outbound_URL to point to the new Tomcat 3.2.1 installation of Partner Agreement View 2.2 Channel API address. Note that the Outbound URL is case-sensitive.
4. Ensure you’re using the new sample processes, then run them.

Upgrade the Channel API computer

To upgrade the Channel API computer:

1. Upgrade IBM HttpServer to 1.3.12.2.
2. Install a fresh version of Apache Tomcat 3.2.1.
3. Install Partner Agreement View 2.2 on Tomcat 3.2.1
4. Test Partner Agreement View 2.2 in this environment. That is, run our sample example servlets. This ensures you've installed and configured the new Partner Agreement View 2.2 correctly
5. Update your PAV applications, as described in *Upgrading Partner Agreement View applications from 1.1 to 2.2* on page 105.
6. In each of your PAV application, implement the new partner logoff:
   - A Create an instance of AppChannel object **queue**, using channel_id, partner_id, and password.
   - B Use **queue.deliverLogoffMessage**() API to implement logoff in your web application.
7. Edit the `appchannelEnv.bat` file to point to correct JDK, Tomcat root, and virtual root.
8. Compile your PAV application servlets, using Partner Agreement View 2.2. For more information, see *Compiling your servlets* on page 60.
9. Test your migrated PAV application using Partner Agreement View 2.2.

APPENDIX

C

Upgrading Partner Agreement View applications from 1.1 to 2.2

Between Partner Agreement View 1.1 and 2.2 there are a number of API changes that require servlet developers to make code changes. Some classes are now deprecated and might not be supported in future versions. For now, these deprecated classes work in version 2.2.

Sections in this appendix include:

These instructions assume that you have already installed Process Manager 2.2.

Affected areas

For the most part, the API changes affect the Business Objects and Business Object rendering in HTML (known as the “BO Printers”). The BusinessObject implementation and the Printer technology was completely rewritten in order to add support for the full DTD 1.0 specification (particularly Attributes, OR constructs, and ANY constructs). Some additional changes where required to make the appliations DBCS (double-byte character set) compliant, which is is necessary for internationalized web applications.

The following classes/packages have been deprecated:

- All classes in package appchannel.util
- All classes in package com.extricity.web.bo (except XMLUtils)
- All classes in package com.extricity.web.bo.util
- All classes in package com.extricity.web.bo.printer
- All classes in package com.extricity.web.html

The new classes that replace these, are:

- com.extricity.document.defs.dom.BOWalker;
- com.extricity.document.apps.bo.TextPrinter;
- com.extricity.document.apps.bo.FormPrinter;
- com.extricity.document.defs.dom.WebAppBOFactory;
- com.extricity.document.apps.bo.util.PropertyValueProvider;
- com.extricity.document.apps.bo.MapWriter;

The two best sources of information on these changes are:

- The Partner Agreement View Javadocs, which you can access at
- The sample servlet files. The sample files are in appchannel.sample. You can look at your previous installation (Partner Agreement View 1.1) for older sample files.

Remove references to deprecated classes

Remove all references to **DisplayBO**, **DisplayBOFactory**, **TextBlockPrinter**, **TextImagePrinter**, **FormTablePrinter,** and so on, because these classes are deprecated.

Remove any of these import statements, if they occur in your servlets:

- import com.extricity.web.bo.DisplayBO;
- import com.extricity.web.bo.DisplayBOFactory;
- import com.extricity.web.bo.util.PropertyValueProvider;
- import com.extricity.web.bo.printer.TextBlockPrinter;
- import com.extricity.web.bo.printer.TextImageTablePrinter;
- import com.extricity.web.bo.printer.FormTablePrinter;
- import com.extricity.web.html.HtmlColor;
- import appchannel.util.FormHandler;

Add these import statements:

- import java.io.OutputStreamWriter;
- import java.io.Writer;
- import java.io.FileOutputStream; (if you are using file input/output operations)
- import java.io.PrintWriter;
- import com.extricity.document.defs.dom.BOWalker;
- import com.extricity.document.apps.bo.TextPrinter;
- import com.extricity.document.apps.bo.FormPrinter;
- import com.extricity.document.defs.dom.WebAppBOFactory;
- import com.extricity.document.apps.bo.util.PropertyValueProvider;
- import com.extricity.document.apps.bo.MapWriter;

Code changes

The following sections on code changes reflect the change in classes and methods used for business object rendering in HTML (BO Printers).

Getting business objects from ProcessMessages

The displayBO class is replaced with BusinessObject class. Thus, you must change code that gets business objects from ProcessMessages.

Replace this code

```
ProcessMessage msg
DisplayBO  bo = (DisplayBO) msg.getBusinessObject( );
```

With this

```
ProcessMessage msg
BusinessObject  bo = (BusinessObject) msg.getBusinessObject();
```

Getting business objects from DisplayBOFactory

The **DisplayBOFactory** class has been replaced with **WebAppBOFactory**. You must change your code used to get business objects.

Replace this code

```
DisplayBO bo = DisplayBOFactory.getInstance( ).getDisplayBO(
   new BOTypeID(bo_type_id)
);
```

With this

```
// First create an instance of WebAppBOFactory and then use
// instanceFromDTD(bo_type_id) to get the BusinessObject
WebAppBOFactory bean = new WebAppBOFactory( );
BusinessObject bo = bean.instanceFromDTD(bo_type_id);
```

Get DTD for BOTypeID

The **getDTD( )** method for the **HttpDocumentManagementBean** class now requires a definition ID.

Replace this code

```
//get the DTD using HttpDocumentManagementBean
String dtdString = docBean.getDTD(
   msg.getMessageInfo().getBOTypeID()
);
```

WITH THIS

```
//get the DTD using HttpDocumentManagementBean
String dtdString = docBean.getDTD(
  msg.getMessageInfo().getBOTypeID().getDefinitionID()
);
```

USE UTF-8 ENCODING WITH GETBYTES( )

When calling the **getBytes**( ) method, you must specify the UTF-8 encoding constant, otherwise the method will use the system default encoding. This is necessary for DBCS compliance.

REPLACE THIS CODE

```
strMsgFileName.getBytes( )
```

WITH THIS

```
strMsgFileName.getBytes(Constants.ENCODING_UTF_8)
```

WRITING BUSINESS OBJECTS TO A FILE

When writing business objects to a file, you must not write the XML prolog and DOCTYPE declaration. Instead you use the **toStream**(**writer**) method, which does this for you.

REPLACE THIS CODE

```
//write xml prolog
writer.write(Constants.DEFAULT_XML_DECLARATION);

//form the <?DOCTYPE declaration
String docType = XMLUtils.getDocTypeFromBO(bo);

//write DOCTYPE
writer.write(docType);

//write BusinessObject
writer.write(bo.toString());
```

WITH THIS

```
//write BusinessObject along with prolog and DOCTYPE declaration
bo.toStream(writer);
```

SPECIAL PRINTER CLASSES NOT SUPPORTED

The **TextImageTablePrinter** and **TextBlockPrinter** classes are no longer supported; instead you should use the **PrintWriter** class with the **OutputStreamWriter** class. For DBCS support, use the UTF-8 encoding when creating **OutputStreamWriter**.

You should also remove any code that sets the colors of optional and required groups and fields on printer, since this is no longer supported in Partner Agreement View 2.2.

REPLACE THIS CODE

```
writer respWriter =
   new OutputStreamWriter(response.getOutputStream(), "UTF-8");
TextImageTablePrinter text_printer = new TextImageTablePrinter(respWriter);
```

or

```
PrintWriter printer = new PrintWriter( new OutputStreamWriter(
   response.getOutputStream(), "UTF-8"),
   true);
TextBlockPrinter text_printer = new TextBlockPrinter(printer);
```

WITH THIS

```
PrintWriter printer = new PrintWriter( new OutputStreamWriter(
   response.getOutputStream(), "UTF-8"), true);
```

When completing creation of HTML code in a servlet, you must now use the **flush**( ) method on the printer object.

ADD THIS CODE

```
printer.flush();
```

PRINTING NON-EDITABLE BO INSTANCE DATA

Rather than using the **TextBlockPrinter** class to print non-editable business object instance data, you now create a **TextPrinter** instance and use the **BOWalker** class to traverse all nodes and/attributes of the business object.

REPLACE THIS CODE

```
PrintWriter printer = new PrintWriter( new OutputStreamWriter(
   response.getOutputStream(), "UTF-8"), true);
```

```
TextBlockPrinter printer = new TextBlockPrinter(response.getWriter());
printer.print((DisplayBO) boReceived);
```

WITH THIS

```
//create "PrintWriter instance
PrintWriter printer = new PrintWriter( new OutputStreamWriter(
   response.getOutputStream(), "UTF-8"), true);
//pass the PrintWriter to "TextPrinter" to print the BO
TextPrinter txt_printer = new TextPrinter(printer);

//create a BOWalker to walk through all nodes and/attributes of the BO
BOWalker walker = new BOWalker(bo.getTypeID(), txt_printer);
walker.traverse((org.w3c.dom.Node)bo.getRootElement());
```

PRINTING EDITABLE BO INSTANCE DATA

Rather than use the **FormHandler** class to print editable business object instance data, you now create a **FormPrinter** object and use the **BOWalker** class to traverse all nodes and/attributes of the business object.

REPLACE THIS CODE

```
FormHandler printer = new FormHandler(request, response);
// Print the BO instance Data
printer.print(bo);
```

WITH THIS

```
FormPrinter form_printer = new FormPrinter(printer, bo);
//prints validation JAVAScript
form_printer.printValidationScript();
//print form tag ...<form name="display_options" onSubmit='return
//verify(this);' method="post" action="">
form_printer.openFormTag(null);
// Print the BO instance Data
BOWalker walker = new BOWalker(bo.getTypeID(), form_printer);
walker.traverse((org.w3c.dom.Node)bo.getRootElement());
// Print Submit button on HTML page
form_printer.printSubmitButton();
//Print form close tag ...</Form>
form_printer.closeFormTag();
```

Retrieving validation errors

The new API allows you to use get validation errors for business objects.

ADD THIS CODE

```
WebAppBOFactory bean = new WebAppBOFactory( );
//get the validation error...
String validation_error = bean.getValidationError(bo);
```

Form handlers not supported

The Partner Agreement View API no longer supports the **FormHandler** class. Remove this code from your servlets.

REPLACE THIS CODE

```
FormHandler printer = new FormHandler(request, response);
printer.setOutputFormat(FormHandler.IMAGE_TABLE_FORMAT);
printer.setServletContext(this.getServletContext());
```

WITH THIS

```
//now we don't support setting different HTML display layout options
PrintWriter printer = new PrintWriter( new OutputStreamWriter(
response.getOutputStream(), "UTF-8"), true);
```

Check for Submit button

Because the FormHandler class is no longer supported, use the **WebAppBOFactory** class to check for when the user clicks the Submit button.

REPLACE THIS CODE

```
FormHandler form_printer = new FormHandler(request, response);
//returns true only if the user clicks "Submit" button
boolean  flag = form_printer.receive(bo);
```

WITH THIS

```
WebAppBOFactory bean = new WebAppBOFactory();
//returns true only if the user clicks "Submit" button
 boolean  flag = bean.isFinalSubmission(request);
```

Get business object type

To get a business object's type, you now use the **getRootElementName**( ) method.

Replace this code

```
String boType = boReceived.getTypeURI()
```

With this

```
String boType = bo.getTypeID().getRootElementName()
```

Setting business object values from a map file

In the earlier versions of the Partner Agreement View API, you could initialize a business object with the **initialize**( ) method of the **PropertyValueProvider** class. In the current implementation, you must create a **FormPrinter** object and use the **setMap**( ) method to initialize the business object.

Replace this code

```
// set the default values using a property value provider
Properties propMapDefaults = new Properties();

// Load the map from the filesystem (map should be in
// the same directory as this class instance)
InputStream stream =  this.getClass().getResourceAsStream(boMapName);
propMapDefaults.load(stream);
stream.close();

// add current date to the list of properties
// This demonstrates how to set a variable property on the default
// value provider at runtime
propMapDefaults.setProperty("today", new java.util.Date().toString());

//initialize the business object ...
new PropertyValueProvider(propMapDefaults).initialize(bo);
```

With this

```
// set the default values using a property value provider
Properties propMapDefaults = null;

// Load the map from the filesystem (map should be in
// the same directory as this class instance)
InputStream stream =  this.getClass().getResourceAsStream(boMapName);
if (stream != null) {
   propMapDefaults = new Properties();
   propMapDefaults.load(stream);
   stream.close();
   // add current date to the list of properties
   // This demonstrates how to set a variable property on the default
   // value provider at runtime
   propMapDefaults.setProperty("today",
   new java.util.Date().toString());
}

PrintWriter printer = new PrintWriter( new OutputStreamWriter(
   response.getOutputStream(), "UTF-8"), true);

FormPrinter bo_printer = new FormPrinter(printer, bo);

//print the business object with the values from map file
bo_printer.setMap(propMapDefaults);
```

For more information about printing business object data, see *Printing non-editable BO instance data* on page 110 and *Printing editable BO instance data* on page 111.

Creating a map file from business object data

In Partner Agreement View 2.2, the **MapPrinter** class has been replaced with a combination of the **MapWriter** and **BOWalker** classes.

There are three things to note in the following code examples:

- The **FileWriter** class has been replaced by the **BufferedWriter** class, which is preferred. The **BufferedWriter** class allows you to specify the UTF-8 encoding for DBCS support.
- The **MapPrinter** class is replaced by the **MapWriter** class and the **BOWalker** class.
- The **traverse**( ) method of BOWalker replaces the **print**( ) method of the MapPrinter class.

Replace this code

```
// Build a Map for this instance of the Business Object
// with a MapPrinter.  This will write the Built map to the
// filesystem in the directory specified by filename.
// The Map will contain the current BusinessObject's instance data
// as the .data fields for the map.

FileWriter file_writer = new FileWriter(filename);
MapPrinter mapPrinter = new MapPrinter(file_writer, true);

mapPrinter.print((DisplayElement)bo.getRootElement());
```

With this

```
// Build a Map for this instance of the Business Object
// with a MapPrinter.  This will write the Built map to the
// filesystem in the directory specified by filename.
// The Map will contain the current BusinessObject's instance data
// as the .data fields for the map.
BufferedWriter file_writer =  new BufferedWriter(new
OutputStreamWriter(new FileOutputStream(filename),
Constants.ENCODING_UTF_8));
//create mapwriter using Botype id and file writer
MapWriter mapwriter = new MapWriter(file_writer, boTypeID);
//create a mapwalker using Botype id and mapwriter
BOWalker mapwalker = new BOWalker(bo_type_id, mapwriter);
mapwalker.traverse((org.w3c.dom.Node)bo.getRootElement());
```

Message map functionality not supported

In previous versions of the Partner Agreement View API, the message map functionality allowed a map file to include syntax in the form:

```
BO2/otherpath/version/two=BO1/path/version/one
```

Which states that when the input business object has a field named BO1/path/version/one and the response business object has a field named BO2/otherpath/version/two, the data from the input business object should be copied into the response business object.

This functionality is not currently supported in Partner Agreement View 2.2.

Replace this code

```
// After loading properties successfully, initialize the new business object
// from the source BO and data from the map file ...
new PropertyValueProvider(sourceBO, propertyMap).initialize(bo);
```

With this

```
//set BO field data
// from Request_For_Quotation/RFQ_Header/RFQ_Number to
// Quotation/RFQ_Response_Header/RFQ_Number
bo.getRootElement().setData("RFQ_Response_Header/RFQ_Number",
  sourceBO.getRootElement().
  getData("RFQ_Header/RFQ_Number"));

//set BO field data having zero or multiple lines
//Request_For_Quotation/RFQ_Lines[*]/RFQ_Line_Number to
// Quotation/RFQ_Response_Lines[*]/RFQ_Line_Number
bo.getRootElement().setData("RFQ_Response_Lines[0]/RFQ_Line_Number",
  sourceBO.getRootElement().
  getData("RFQ_Lines[0]/RFQ_Line_Number"));

//Copy data
//from Request_For_Quotation/RFQ_Header/Contact to
//Quotation/RFQ_Response_Header/Responder_Contact
bo.getRootElement().getElement("RFQ_Response_Header/Responder_Contact").
  copyIn(sourceBO.getRootElement().
  getElement("RFQ_Header/Contact"));
```

APPENDIX

D

Configuring Tomcat and Apache on UNIX

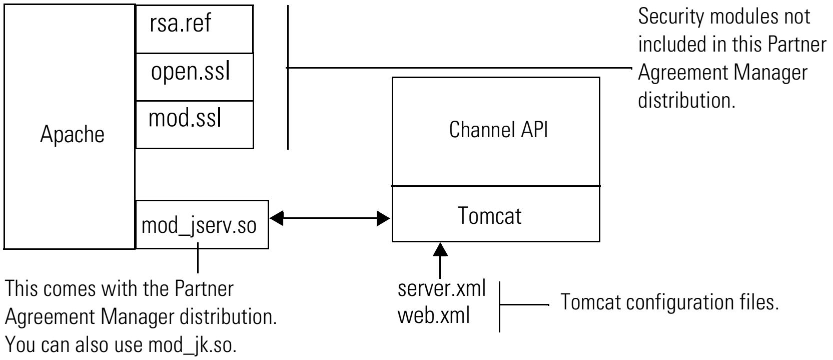
Read this appendix for information on installing and configuring Tomcat with your Apache Web server on UNIX. This section includes:

- *About Apache Tomcat* on page 120.
- *Installing Tomcat with Apache/SSL* on page 121.
- *Installing Tomcat with Apache/SSL* on page 121.
- *Installing Tomcat with Apache/SSL* on page 121.
- *Installing Tomcat with Stronghold* on page 123.
- *Using Tomcat with other Apache-based Web servers* on page 124.

Note: At present time, there is no redirector for Tomcat and Apache for AIX.

About Apache Tomcat

Partner Agreement View uses the Tomcat servlet engine produced by the Apache Jakarta project. You can deploy Tomcat to work as a servlet engine for another Web server. For example, you can configure the Apache Web server to serve HTML pages, using Tomcat strictly as a servlet engine.



Apache
rsa.ref
open.ssl
mod.ssl
mod_jserv.so
Channel API
Tomcat
server.xml
web.xml
Security modules not included in this Partner Agreement Manager distribution.
This comes with the Partner Agreement Manager distribution. You can also use mod_jk.so.
Tomcat configuration files.

In this configuration, a commercial strength Web server, such as Apache, receives URL requests. It then forwards the appropriate requests to Tomcat, where they can be serviced by the Partner Agreement View API. To achieve this configuration, the Web server needs to be configured to redirect to Tomcat. For more information on Tomcat, see the Apache Jakarta web site: jakarta.apache.org.

Apache (www.apache.org) provides redirectors for Apache and iPlanet that allow those Web servers to redirect to Tomcat. As there is a wide variety of possible configurations, the Partner Agreement View installer does not automatically add the Tomcat redirector to the Web server.

Installing Tomcat with Apache/SSL

This section outlines the steps required to install and configure an Apache web server. For more information on the Apache web server, see www.apache.org. For more information on SSL, see www.modssl.org and www.openssl.org.

Before installing Apache/SSL, you must have a C++ compiler installed.

1 Download the Apache and SSL packages.

Download this	From
apache_1.3.12.tar.gz	www.apache.org
mod_ssl-2.6.6-1.3.12.tar.gz	www.modssl.org
openssl-0.9.5a.tar.gz	www.openssl.org
RSA implementation	See the INSTALL file in mod_ssl for more information on RSA implementations.

2 Copy these four components to a directory called ApacheSrc.

3 Go to that directory.

```
cd ApacheSrc
```

4 Uncompress and untar these files.

```
gunzip openssl-0.9.5a.tar.gz
tar xvf openssl-0.9.5a.tar
gunzip mod_ssl-2.6.6-1.3.12.tar gz
tar xvf mod_ssl-2.6.6-1.3.12.tar
gunzip apache_1.3.12.tar.gz
tar xvf apache_1.3.12.tar
```

5 Unpack your RSA implementation.

6 Go to the mod SSL directory.

```
cd mod_ssl-2.6.6-1.3.12
```

Build and configure Apache with SSL, following the instructions outlined in INSTALL file included in this directory.

7 Now that Apache with SSL is successfully installed, enable shared modules. Shared modules are required as the Tomcat redirector is a shared module. In the mod_ssl-2.6.6-1.3.12 directory, execute the following commands:

```
./configure --with-apache=../apache_1.3.12 \
  --with-ssl=../openssl-0.9.5a \
  --with-rsa=<your RSA library> \
  --prefix=/path/to/your/Apache/installation \
  --enable-module=so \
  --enable-rule=SHARED_CORE
cd ../apache_1.3.12
make
make install
```

In these commands, replace *your RSA library* with the actual RSA library name. Likewise, replace */path/to/your/Apache/installation* with the actual path to your Apache installation.

You may need to generate a certificate via the command make certificate if you do not have a certificate to use.

IMPORTANT: Verify that your Apache Web server is functioning correctly before proceeding further. To verify your Apache Web server is working, start it. Then open a browser and request a page.

ADDING TOMCAT TO AN APACHE INSTALLATION

Now that you have your Apache Web server installed, you must install the Tomcat servlet engine.

If you have difficulty with the mod_jserv.so that is distributed on the Partner Agreement View CD, you will have to build it yourself for your platform (one symptom of the difficulty might be that Apache crashes after you add mod_jserv.so). For more information on this, see *Using Tomcat with other Apache-based Web servers* on page 124.

1 On the Partner Agreement View CD, find the file named Webservers/Tomcat/redirector/binary/apache1.3/<*webserver*>. Unpack this into your local file system. This directory contains a Tomcat redirector (mod_jk.so) and the source to build mod_jk.so yourself for alternate configurations.

2 Copy the Tomcat redirector into the Apache **libexec** directory. This directory is where all the shared modules are stored.

```
cp mod_jk.so /ApacheRoot/libexec
```

IMPORTANT: The **mod_jserv.so** in this command is not the **mod_jserv.so** that comes with the Apache installation. Tomcat has its own version of **mod_jserv.so** that is supplied on the Partner Agreement View CD.

3 Copy the Tomcat configuration file **<tomcat>/conf/tomcat.conf** to the Apache configuration directory:

```
cp <tomcat>/conf/tomcat.conf /<apache install>/conf
```

Where *tomcat* is the directory into which Tomcat was installed when you installed Partner Agreement View; and *apache install* is the correct path for your Apache installation.

4 Add the Tomcat configuration file to the Apache configuration file.

Add the contents of **tomcat.conf** to the bottom of the **httpd.conf** file.

You have successfully configured Apache with Tomcat.

NOTE: On UNIX, you will need to edit **tomcat.conf** after you finish installing the Channel API, as described in *Installing the Channel API* on page 24.

Installing Tomcat with Stronghold

Stronghold from **www.c2.net** is an Apache Web server with commercial support. Since it is based on Apache, mod_ssl and openssl, you can use the **mod_jserv.so** that ships on the Partner Agreement Manager CD to configure Stronghold with Tomcat, following the same directions as those for Apache. See *Adding Tomcat to an Apache installation* on page 122.

IMPORTANT: Stronghold creates an additional directory **APACHE_ROOT/modules**. The libexec directory is created under this modules directory. This means you must change the LoadModule variable in the alliance-tomcat-apache.conf file, to have the correct path which includes the modules directory. If you do not do this, Apache will not start.

Using Tomcat with other Apache-based Web servers

Other Apache-based Web servers, such as Raven from www.covalent.net, do not use the same set of modules that were used to build the mod_jserv.so that ships on the Partner Agreement Manager CD. Therefore, you need to build your own mod_jserv.so. On the Partner Agreement Manager CD, we have provided the source to the Tomcat redirector.

To create mod_jserv.so:

1 Go to the Tomcat redirector source directory.

```
cd jserv
```

2 Create mod_jserv.so.

```
apxs -o mod_jserv.so -DEAPI -c *.c
```

apxs is from the /bin directory of your Apache-based Web server.

To add Tomcat to your Apache installation, see *Adding Tomcat to an Apache installation* on page 122.

APPENDIX

E

Installing the IBM HTTP Server and configuring SSL

The IBM HTTP Server is the recommended Web server for Partner Agreement View running on Windows NT. Sections in this appendix include:

- *Installing the IBM HTTP Server* on page 125.
- *IBM HTTP Server redirection* on page 127.
- *Importing certificates* on page 127.

Installing the IBM HTTP Server

To install the IBM HTTP Server:

1. Download version 1.3.12.2 of the software from http://www-4.ibm.com/software/webservers/httpservers/download.html.
2. Create a user on the Windows NT server. Name the user ibmhttp, give it the password ibmhttp, and assign it administrative privileges, including the permission to log on as a service. The installation requires this user name and password.
3. Run the installer, which installs and starts both the HTTP Server and the Administration Server.
4. Open a browser and enter the URL http://<yourcomputername>, to verify that the server is running and to begin configuring SSL.

5 Click Configure Server and enter the user name and password that you created in step 2 (ibmhttp).

6 Open another browser and enter the URL http://<yourcomputername>, to see the configuration instructions:

   A. Click View Documentation.

   B. Click How To.

   C. Click Get Started.

   D. Click With Secure connections.

   E. Follow the instructions to configure the HTTP Server and enable SSL.

7 Use the Key Management Utility to create a new key database for storing certificates. Enter and confirm the password for the key database, and click Stash Password to a File. The password for the key database must match the password you use to run the HTTP Server (ibmhttp).

8 Use the Key Management utility to create a self-signed certificate. Creating a self-signed certificate enables you to configure SSL quickly; after configuring SSL, request a certificate from a certificate authority.

IMPORTANT: The Create New Self-Signed Certificate dialog box contains a Common Name field, for the name of the computer where you installed the IBM HTTP Server. You must enter the *fully qualified* name of that computer in the Common Name field.

9 Follow the IBM documentation you opened in step 6 to configure SSL, using the HTTP administration server.

The administration server adds information to the httpd.conf file as you configure SSL. The httpd.conf file is in the conf directory, in the directory where you installed the HTTP Server. It's a good idea to open the httpd.conf file and review the additions as they occur, to ensure that the configuration information is correct.

IMPORTANT: When the HTTP administration server creates a server name, it adds a period (.) to the end of the server name you enter. However, when this server name is used in redirection, the resulting URL is invalid. To ensure that redirections work, edit the httpd.conf file and either remove the trailing period or add a domain name to the server name, so that it is fully qualified (for example foo.domain.com)

10. Use the URL **https://computer_name** to open the main IBM server page and verify that the server is running and that the SSL configuration is correct.

IBM HTTP Server redirection

To redirect web requests from IBM HTTP Server to Tomcat:

1. Copy the **mod_jk.dll** from the **WebServer/IBMHttp_Redirect** directory of your Partner Agreement View CD to the **modules** directory in your IBM HTTP Server installation.
2. Append the **tomcat.conf** file, found in **WebServer\IBMHTTP_Redirect** on the CD, to the end of the **httpd.conf** file for the IBM HTTP Server.

   Notes about the portion of the **httpd.conf** appended from **tomcat.conf**:

   - The **ApJServDefaultPort** value must match the port value for **Ajp12ConnectionHandler** in **tomcat\server.xml**. By default both will have value 8007.
   - In the "Context mapping" section, the line **ApJServMount /WebSphere/PAV /root** is the default virtual root for the PAV Channel API. If you do not accept the default virtual root during your PAV Channel API installation, you must edit this line so that it contains the virtual root that you chose during installation.

Importing certificates

Before the Process server and the Partner Agreement View web server can respond to requests from each other, each must import the SSL certificate from the other.

To enable trust for a self-signed certificate that is not trusted (that is, a certifying authority has not issued the certificate), you use Keytool to import the certificate. You must use the Keytool provided in JDK 1.3. To download JDK 1.3, browse to http://java.sun.com/j2se/1.3/ and find the download page.

Note: The Keytool is also provided in JDK 1.2, but that version cannot generate the RSA keys.

For information about how to use Keytool, see:

- Windows NT: http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html.
- Solaris: http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/keytool.html

To get the certificate from a server:

1. Use a Web browser to connect to the server. For example, from the Process server machine, use a Web browser to connect to the Web server on the Partner Agreement View computer.
2. Because the browser has not yet confirmed receipt of a certificate, you will see a security alert.
3. Click on View Certificate and select the Details tab.
4. Click on Copy to file and save the certificate to a location you can easily remember.
5. To import the certificate on the Process server, change directory to <*JAVA_HOME*>\jre\lib\security directory (where *JAVA_HOME* is the location of the JVM). Then use keytool to import the certificate using this command:

```
keytool -import -alias <certificate alias name>
    -file <full path to the saved certificate>
    -keystore jssecacerts
```

NOTE: To import the certificate on the Partner Agreement View computer, change directory to <*JAVA_HOME*>\jre\lib\security, where *JAVA_HOME* is the location of the JVM (as specified in tomcat.bat in step 2 of *Installing the Tomcat servlet engine* on page 23), and then run keytool as described above.

APPENDIX

F

CONFIGURING PER-PARTNER OR SHARED QUEUES

When you configure the Channel Interface and Channel API, you can choose one of two queue types: per-partner or shared. This appendix describes these queue types and helps you choose the queue type that works best for your application. Sections in this appendix include:

- *Active state queues* on page 129.
- *Altering the queue mode* on page 130.

IMPORTANT: You must specify the same queue type on both the Channel Interface (on the Process Server) and the Channel API (on the Web server).

ACTIVE STATE QUEUES

The computer that runs the channel API can maintain two types of ActiveState queues:

- A shared queue allows all partners to access a single queue.
- Per-partner queues creates a queue for each partner. This is more secure and is more efficient, because each partner can only see the their own active states. This is optimal for Partner Agreement View, where each partner logs in separately to the Partner Agreement View application. Partner Agreement View is installed in per-partner mode.

Altering the queue mode

If you want to alter the mode after installing Partner Agreement View, follow the steps described here.

1 If there are any existing (in-progress) processes, stop them.

2 On the Partner Agreement Manager side, modify the Uses_Shared_Queue boolean property in the Other tab of the Channel Profile. For more information on modifying the Channel Profile, see *Setting Channel Profiles* on page 40.

3 Stop the Tomcat Web server on the channel API computer.

4 On the channel API computer, locate the AppChannel.properties file and set the "uses.shared.queue" property to the desired value (either true or false).

The location of the AppChannel.properties depends on the platform:

On this platform	The file is in
Windows NT	<*tomcat*>\webapps\WebSphere\PAV\conf\AppChannel.properties.
UNIX	<*tomcat*>/jakarta-tomcat-3.2.1/webapps/WebSphere/PAV/conf/AppChannel.properties

The relevant portion of the AppChannel.properties file looks like this:

```
#This determines the channel mode i.e. whether it uses shared queue
#to store active states for all partner or there is a separate
#queue for each of the partner
#   true  - indicates channel uses shared queue mode
#   false - indicates channel uses separate queue to store active states
#
# Template Example:
# --------------------------
## uses.shared.queue.XXXX=true
uses.shared.queue.1001=true
```

IMPORTANT: You must set this property to the same value on both the Partner Agreement Manager side and Partner Agreement View Web server side.

5 Start the Tomcat server on the channel API computer.

6 In the Administration folder of the Process Manager window, click on Passwords.

A For per partner queue mode use,

```
Login name: AppChannelInboundPartner<Instance-ID>
password: <partner_password>
```

Where *Instance-ID* is the partner ID (a numeric value). For example, if your partner ID is 777, the login name would be AppChannelInboundPartner777. The *partner_password* is whatever text you choose; however for the samples, the password is actually "partner_password".

B For shared queue mode, use :

```
Login name: AppChannelInboundChannel<Instance-ID>
password: <channel_password>
```

Where Instance-ID is the channel instance ID (an integer value). For example, if your channel instance ID is 1004, the login name would be AppChannelInboundChannel1004. The password is whatever text you choose; however for the samples, the password is actually "channel_password".

7 Start new processes from Partner Agreement Manager. The new processes should show-up in your queue.

NOTE: If you don’t stop existing processes, you will have to shut down Partner Agreement Manager. After restarting Partner Agreement Manager, those processes should show up in your queue.

APPENDIX

G

Setting the conversation aware mode

When Partner Agreement View is installed, conversation threading is turned off by default. This appendix describes how to turn on conversation threading for Partner Agreement View. Sections in this appendix include:

- *Channel Interface* on page 134.
- *Channel API* on page 135.

Note that there are two separate pieces: you must turn on conversation threading for the Channel Interface (on the Process Server) as well as the Channel API (on the Web server).

Note that you must perform this step for both the Process Server, as well as the Partner Agreement Manager client.

To turn on conversation threading for the Channel Interface, perform these steps:

1 On the Process Server, locate the file <PAM *root*>\Alliance\Properties\Alliance.properties. Set the properties shown here:

```
channel.manager.general.conversations.enable=true;
mode=client;
type=boolean;
desc=whether to display the conversation navigation item in
the general group of the channel manager
```

These properties might not be present; if they aren't, you should add these lines. If they are present, ensure that channel.manager.general.conversations.enable is set to true.

This enables the conversation definition editor to be viewed from within the Channel Manager.

2 On the Process Server, locate the file: <PAM *root*>\Alliance\com\extricity\channels\appchannel\PAVChannelType<*instance ID*>.properties (where *instance ID* is the Channel Instance ID). Add the value 0x00020000 to the constraint list.

```
Constraints=0x00000008 ... 0x00020000
```

This switch turns the conversation definitions requirement on. To pass verification, processes with partners on this channel now require conversations to be defined for outbound and inbound business objects.

3 Using the Conversation editor in the Channel Manager, create a new conversation between the inbound business object type and the outbound business object type. For more information about conversations, see *Creating conversations* on page 47.

The outbound business object is chronologically before the incoming one. You must bind a particular field or fields from one business object type to particular fields of the other business object. (For example, Partner Agreement Manager sends a PO to its Partner Agreement View partner, and the partner sends back a PO Ack. The PO Ack is the incoming business object and the PO is the outbound business object.)

4 Deactivate any processes that you had installed before changing conversation threading. Save the processes as new versions, make sure that conversation definitions are defined in all necessary cases, verify the processes, and install them for execution.

Channel API

To turn on conversation threading for the Channel API:

1 On the Web server computer, locate the AppChannel.properties file.

The location of the AppChannel.properties file is different for each platform:

On this platform	The file is in
Windows NT	<*tomcat*>\webapps\WebSphere\PAV\conf\AppChannel.properties
UNIX	<*tomcat*>/jakarta-tomcat-3.2.1/webapps/WebSphere/PAV/conf/AppChannel.properties

The relevant section of AppChannel.properties looks like this:

```
# This determines whether channel is conversation  aware or not.  If
# the Application (EDI or WebApp) can keep track of the conversation
# identifier (in this case the ActiveStateID) to pair up a given response
# with a given request, then this Application is "conversation aware".
# If it cannot (as in EDI's case), then it is unaware and requires
# use of the Process Server's process/conversation threading functionality.
#    true  - indicates channel is conversation aware   (i.e WebAppChannel)
#    false - indicates channel is conversation unaware (i.e EDIChannel)
#
# Template Example:
# ---------------------------
## conversation.aware.XXXX=false
conversation.aware.1001=false
```

Note: Both the Process Server and Tomcat must be restarted before these changes take effect.

APPENDIX H

NOTICES

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information 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 information. The furnishing of this information 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 information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information 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 Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

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 Programming 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.

Trademarks

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

AIX
DB2
IBM
MQSeries
SupportPac
WebSphere

Pentium is a registered trademark of Intel Corporation in the United States and/or other countries.

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

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

Glossary

action—a task performed as part of a private process. A private process action is the equivalent of a step in a public process. See the following terms in this glossary for more information about the action types you can include in a private process:

- approval action
- extension action
- mapping action
- notification action
- output object action
- script action
- subprocess action
- termination action
- timer action

See also *private process*.

adapter—the software bridge between Partner Agreement Manager processes and specific end-system and business-application interfaces. Adapters manage interactions between business applications and the Adapter Server. They allow private processes to interact with external business applications while a process is running, and they allow PAM to start public processes based on events that occur in external business applications. See also *adapter implementation*, *adapter instance*, *adapter type*.

adapter implementation—the implementation declaration for an adapter type. It specifies the name and location of the Java source file that defines the application logic used to communicate with a specific end system through that end system's interface. The application logic is specified in the form of properties. See also *adapter*, *adapter instance*, *adapter type*.

adapter instance—an instance of an adapter implementation. The adapter instance is used in a private process extension action and provides the specific values to be used for the properties declared in the adapter implementation. See also *adapter*, *adapter implementation*, *adapter type, extension action*.

adapter type—a definition that is stored in XML format and specifies the adapter's properties as well as the operations and events it supports. A single adapter type can have multiple implementations, and each implementation can have multiple instances. See also *adapter*, *adapter implementation*, *adapter instance*.

approval action—a private process action that you use to ask for a response from a user before letting the process continue to run. You can use an approval action, for example, to ask for an OK when a purchase order exceeds a predetermined amount. See also *private process*.

business object—a message transmitted as part of a public process. Business objects take the form of purchase orders, acknowledgments, requests for clarification, and so on. See also *business object type*.

business object type—a definition that determines the types of information a message can contain. It has three properties: the top-level element in its element definition set, its key field, and whether instances of it return audit information for non-repudiation purposes. The name of the business object type is the name of the element you select as its top-level element. See also *business object*, *element definition set*, *non-repudiation*.

business object variable—one of the two types of variables used in Partner Agreement Manager to store information within a process. Business object variables create an instance of a business object type. They can be used to store, for example, the outputs from extension actions, the inputs for map actions, or the inputs and outputs for subprocesses. See also *business object*, *business object type*, *extension action*, *variant variable.*

CA—see *certificate authority*.

certificate—a security document that binds a public encryption key to an entity (an individual or organization) known as the principal. The security document (a digital certificate) is signed by another entity known as the issuer. A digital certificate for which both the principal and issuer are the same entity is known as a self-signed certificate. A certificate for which the principal and issuer are different entities is issued by a certificate authority (CA) like VeriSign and is known as a CA-issued (or third-party-signed) certificate. Partner Agreement Manager supports both self-signed and CA-issued certificates. PAM also supports the binding of certificates to be used for signature authentication, message encryption, and SSL authentication for channels other than Partner Agreement Manager. See also *certificate authority*, *SSL*.

certificate authority—a trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The role of the certificate authority, or CA, is to authenticate the entities (individuals or organizations) involved in electronic transactions. CAs are a critical component in data security and electronic commerce because they guarantee that the two parties exchanging information are really who they claim to be. See also *certificate*.

channel—a communications mechanism that encapsulates all the processing information needed to send messages to a partner's system, as well as to translate data received from a partner into Partner Agreement Manager messages. PAM provides channels for RosettaNet, EDI, cXML, and other systems and protocols. See also *message*.

digital certificate—see *certificate*.

DTD—Document Type Definition. A type of file associated with SGML and XML documents that defines how the formatting tags should be interpreted by the application presenting the document. In Partner Agreement Manager, a DTD file contains the complete description of a business object type's element definition set. See also *business object*, *business object type*, *element definition set*.

element definition set—a collection of data fields (or elements) or groups of data fields that defines the structure and meaning of a business object type. See also *business object*, *business object type*.

encryption certificate—see *certificate*.

event—a piece of information that comes into Partner Agreement Manager as a message from another source (an enterprise system or business application, for example) and triggers a public process. See also *message*.

event push—a method that uses the HTTP POST mechanism to push events into Partner Agreement Manager as a way to trigger processes. A port on the Process Server is set to listen for events in the form of HTTP POST messages. When a message is detected, PAM uses the information in the message to generate an event. See also *event*.

extended enterprise—a business model under which companies that work together as partners function as efficiently as a single organization through the implementation of automated communication technologies.

extension action—a private process action that communicates via an adapter with an external application that is registered with Partner Agreement Manager. You can use an extension action, for example, to launch a spreadsheet application, perform calculations, and update the enterprise system, or to get information from an enterprise system or listen for an event in the enterprise system. See also *adapter*, *private process*.

LDAP—Lightweight Directory Access Protocol. LDAP provides a standard method for accessing information from a central directory. After user authentication is set up in the LDAP directory, applications that use the LDAP protocol can retrieve the information from that directory. An authenticated user can log in to any application that supports the LDAP protocol with the same user name and password.

linked certificate—see *certificate*.

map—a Java Script or VBScript that inserts data into fields in an output business object type generated by a private process. The map specifies which fields in the output business object type receive data, and it identifies the information source.

map method—a reusable logical block of code that inserts data into a particular type of element or element sequence in a business object type. Within a map method, you can write the expressions that map individual input and output fields in the sequence. Or you can create a submap and drag input fields to output fields and have Partner Agreement Manager create the appropriate mapping expressions. See also *map*, *submap*.

mapping action—a private process action that you use to call a map. The map specifies the fields in a business object type that will receive data extracted from another source. You use a mapping action when you want to extract data from one business object type and insert it in a different business object type. For example, you use a mapping action to transform a purchase order generated by your inventory system into a sales order in a format that your partner expects. See also *map*, *private process*.

message—a structured communication used to pass information and control to another partner in a public process. The action in the process passes to the partner who receives the message. The content of a message is determined by its business object type. A message can be transmitted via synchronous or asynchronous methods, as determined by its communication service type. See *business object type*.

non-repudiation—a business object security feature that authenticates instances of a business object type and maintains an audit record to verify that they were received by the intended recipient. For business object instances that you receive, Partner Agreement Manager authenticates each instance and maintains an audit record to verify that the instance actually originated with the stated originator. If you disable auditing for a business object type, non-repudiation support is disabled for all messages that contain instances of that business object type.

notification action—a private process action that you use to send an e-mail, fax, or pager message to addressees that you specify. You use a notification action to inform someone inside or outside your organization that an event has occurred. For example, you can use a notification action to alert the order entry department when a purchase order arrives from a customer. See also *private process*.

output object action—a private process action that you use to bind a business object to the expected output object and path in a public process. You use an output object action at the point in a private process when you are ready to send a business object to the associated public process. This is typically the last action in the private process. See also *private process*.

partner group—a group of partners that perform the same role in a process at different times. Instead of duplicating a public process and substituting a different partner name, you can set up a partner group for the public process and then designate a specific partner as the participant when you start an instance of the process. For example, you might design a generic purchasing process that works equally well with any of your suppliers and then designate the appropriate partner when you start the process.

partner profile—information that identifies an organization, specifies a contact person in that organization, lists the communication services the organization supports, and defines the organization's security profile. When partners agree to participate in a public process, they must exchange profile information as a way to ensure authenticity before they can proceed.

PIP—Partner Interface Process. RosettaNet PIPs are specialized system-to-system XML-based dialogs that define business processes between supply-chain partners and provide models and documents for the implementation of e-commerce standards. Each PIP includes a technical specification based on the RosettaNet Implementation Framework (RNIF), a message guideline document with a PIP-specific version of the business dictionary, and an XML message guideline document. See also *RosettaNet*.

post method—the last block of code that is executed when a mapping action runs. Its only parameter is the output business object. You use the post method when you need to perform post-processing on the output business object. For example, you might use the post method to set the value of a summary field based on the number of line items in the output business object, or to examine a range of dates in a repeated group, extract the most recent date, and post that date in a header field. See also *mapping action*, *pre method*.

pre method—the first block of code that is executed when a mapping action runs. The pre method's parameters are the map inputs. You use the pre method to access a map's inputs and set global variables based on their content. See also *mapping action*, *post method*.

private process—a task or set of tasks that business partners participating in a public process perform at points where they need to take action internally. Partners participating in a public process must implement a private process for each public process step that they own. A private process begins with input from the public process and ends with output that feeds back into the public process. The input can be the receipt of a business object from a partner, or it can be a triggering event from an internal system. The output is the business object that transfers control back to the public process. See also *action*, *process*, *public process*.

private process action—see *action*.

process—the flow of actions and the exchange of business information between partners in an extended enterprise. A process operates on two levels, public and private. See *extended enterprise*, *private process*, *public process*.

public process—the step-by-step flow of messages, events, and actions between two or more business partners. Public processes are set up by agreement between partners, and each step in a public process has a private process associated with it. A public process is developed by one partner, and all the partners who participate in it must review and approve it before it can be implemented. The partner who designs a public process is its owner. See also *private process*, *process*.

RosettaNet—a consortium of major information technology, electronic components, and semiconductor manufacturing companies that is working to create and implement industry-wide, open e-business process standards. See also *PIP*.

script action—a private process action that consists of a script written in VBScript or JavaScript and is designed to manipulate information or set up conditional actions based on input. You use a script to establish decision-making criteria for branches or loops, to set variables, or to calculate values that are used elsewhere in the private process. See also *private process*.

security certificate—see *certificate*.

self-signed certificate—see *certificate*.

signature certificate—see *certificate*.

SSL—Secure Sockets Layer. The SSL protocol is a security protocol that provides for communications privacy and reliability over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

submap—a secondary level map that is called by a map method to insert data into an output element other than the top-level element. See *map*, *map method*.

subprocess action—a private process action you use to call an existing public process. You can call any public process in which your organization owns the first partner action. For example, you can use a subprocess to get a quote approved by a third-party supplier before responding to a customer. See also *private process*.

termination action—a private process action that you use to stop a process at a predetermined point for a reason that you specify. You can use a termination action to deal with errors in data that might prevent a process from completing successfully. For example, you might want to stop a process in cases where an enterprise system passes incomplete or corrupted information to it. See also *private process*.

third-party-signed certificate—another name for a CA-issued certificate. See *certificate*.

timer action—a private process action that you use to insert a pause. You can use a timer action to specify the period of time you want to elapse before the next action in the process starts. See also *private process*.

variant variable—single field variables. Variant variables store text strings—the type of information contained in a single field element. You can use variant variables to store the input for actions, to set flags (such as the time-out flag for an approval action), to move information within scripts, or to store the results of an approval action. See also *business object variable*.

Index

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC34-5965-00