IBM WebSphere Business Integration Adapters

Adapter for Trading Partner Interchange User Guide

Version 3.3.x

Note!
Before using this information and the product it supports, read the information in "Notices" on page 87.

31July2003

This edition of this document applies to connector version 3.2.x and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

Contents

About this document

IBM(R) WebSphere(R) Business Integration Adapters supply integration connectivity for leading e-business technologies and enterprise applications.

This document describes the installation, configuration, and business object development for the IBM WebSphere Business Integration Adapter for Trading Partner Interchange.

Audience

This document is for WebSphere consultants and customers who are implementing the connector as part of a WebSphere business-integration system. To use the information in this document, you should be knowledgeable in the following areas:

- Connector development
- Business object development

Related documents

The WebSphere business integration system documentation describes the features and components common to all installations, and includes reference material on specific collaborations and connectors.

This document contains many references to two other documents: the *System Installation Guide for Windows* or *for UNIX* and the *System Implementation Guide for WebSphere InterChange Server*. If you choose to print this document, you may want to print these documents as well.

To access the documentation, go to the directory where you installed the product and open the documentation subdirectory. If a welcome.html file is present, open it for hyperlinked access to all documentation. If no documentation is present, you can install it or read it directly online at one of the following sites:

- If you are using WebSphere MQ Integrator Broker as your integration broker: http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- If you are using InterChange Server as your integration broker: http://www.ibm.com/websphere/integration/wicserver/infocenter

The documentation set consists primarily of Portable Document Format (PDF) files, with some additional files in HTML format. To read it, you need an HTML browser such as Netscape Navigator or Internet Explorer, and Adobe Acrobat Reader 4.0.5 or higher. For the latest version of Adobe Acrobat Reader for your platform, go to the Adobe website (www.adobe.com).

Typographic and other conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, filename, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.

italic, *italic*	Indicates a variable name or a cross-reference.
blue outline	A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
[ ]	In a syntax line, square brackets surround an optional parameter.
...	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.
< >	In a naming convention, angle brackets surround individual elements of a name to distinguish them from each other, as in `<server_name><`*`connector_name`*`>tmp.log`.
/, \	In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes. All WebSphere business integration system product pathnames are relative to the directory where the product is installed on your system.
%text% and *$text*	Text within percent (%) signs indicates the value of the Windows `text` system variable or user variable. The equivalent notation in a UNIX environment is *$text*, indicating the value of the *text* UNIX environment variable.
ProductDir	Represents the directory where the product is installed.

New in this release

New in Release 3.3.x

There is a new appendix for this user guide, Appendix D, "Trading Partner Interchange adapter sample scenario", on page 83 which has procedures for testing the TPI Adapter.

The adapter can now use WebSphere Application Server as an integration broker. For further information, see"Compatibility" on page 7. The adapter now runs on the following platforms:

- Solaris 7, 8
- AIX 5.x

New in Release 3.2.x

Updated in March, 2003. The "CrossWorlds" name is no longer used to describe an entire system or to modify the names of components or tools, which are otherwise mostly the same as before. For example "CrossWorlds System Manager" is now "System Manager," and "CrossWorlds InterChange Server" is now "WebSphere InterChange Server."

The following changes have been made to the connector in this release:

- The adapter has been updated with general maintenance fixes.
- A new data handler, tpi_rnif_mcd2, has been added that supports the TPI MCD 2.0 format. This data handler is contained in file CwDataHandler.jar (file version 2.3.0).

New in Release 3.1.x

The following changes have been made to the connector in this release:

- New samples have been added to demonstrate how the connector works with the WebSphere MQ Integrator Broker.
- The connector now supports dynamic updates to the trading partner configuration file.
- The connector now supports parallel processing for multiple events. To implement this functionality, four new connector-specific properties have been added:
  - DeliverOnArrival—Specifies whether incoming events are processed one at a time (by using the PollForEvents mode) or simultaneously.
  - DeliverOnArrivalThreads—Specifies the number of threads that can be used for parallel event processing.
  - WaitForController—Specifies the amount of time the connector waits for the controller to become active.
  - ControllerWaitCount—Specifies the number of times the connector attempts to contact an inactive controller.

New in Release 3.0.x

The connector has been internationalized. For more information, see "Processing locale-dependent data" on page 5 and Appendix A, "Standard configuration properties for connectors", on page 47

A new connector-specific configuration property, `DataEncoding` has been added to specify the character encoding used by TPI documents. For more information, see "Connector-specific properties" on page 14

New in Release 2.1.x

The following changes are new in version 2.1.x: The IBM WebSphere Business Integration Adapter for Trading Partner Interchange (TPI) includes the connector for TPI. This adapter supports two integration brokers: InterChange Server (ICS) and WebSphere MQIntegrator. An integration broker is an application that performs integration of heterogeneous sets of applications; it provides services such as data routing. The IBM WebSphere Business Integration Adapter for TPI includes the following:

- An application component specific to TPI
- A sample business object (located in the \connectors\Tpi\Samples directory)
- – IBM WebSphere Adapter Framework, which consists of the following:
  - Connector Framework
  - Development tools (including Business Object Designer and Connector Configurator)
  - APIs (including ODK, JCDK, and CDK)

This manual provides information about using the adapter with both the ICS and WebSphere MQIntegrator integration brokers. Enhancements were made to improve the error reporting of the TPI connector.

Note: Because the connector has not been internationalized, do not run it against InterChange Server version 4.1.1 if you cannot guarantee that only ISO Latin-1 data will be processed.

New in Release 2.0.x

In version 2.0.x of the connector, the following changes were made:

New in Release 1.2.x

In version 1.2.x of the connector, minor changes were made to fix defects and to provide compatibility with IBM CrossWorlds infrastructure version 4.0.0.

New in Release 1.1.x

Version 1.1.x of the IBM CrossWorlds TPI connector now supports TPI version 4.0.

Chapter 1. Overview of the TPI connector

This chapter contains the following sections:

Connectors consist of two parts: the *connector framework* and the *application-specific component*. The connector framework, whose code is common to all connectors, acts as an intermediary between the integration broker and the application-specific component. The application-specific component contains code tailored to a particular application or technology (in this case, Trading Partner Interchange). The connector framework provides the following services between the integration broker and the application-specific component:

- Receives and sends business objects
- Manages the exchange of startup and administrative messages

This chapter describes the connector component of IBM WebSphere Business Integration Trading Partner Interchange (TPI). Note that this document contains information about both the connector framework and the application-specific component. It refers to both of these as the connector. For more information about the relationship of the integration broker to the connector, see the *IBM WebSphere InterChange Server System Administration Guide*, or the *IBM WebSphere Business Integration Implementation Guide for WebSphere MQ Integrator Broker*.

TPI Server overview

This section provides a brief overview of the TPI Server functionality. The connector functionality is described in following sections.

The TPI Server enables the secure exchange of documents among trading partners over the Internet. The application packages documents in secure envelopes that are transmitted among trading partners according to user-configured schedules. The TPI Server supports XML, EDI, and binary data formats.

TPI supports FTP, SMTP, and WebSphere MQ for transporting documents among trading partners. The TPI Server uses a system of directories for delivery and retrieval of documents. The TPI Server maintains six document directories for each trading partner — three for inbound and three for outbound. Each of the directories holds incoming or outgoing documents of a specific format—XML, EDI, or binary.

Sending documents

When a document is placed in one of the outbound directories, the TPI Server packages the document by first creating a digest, which includes the sender and receiver identification. The TPI Server then encrypts the document and digest in a single message. Once the document is packaged, TPI sends the message to the specified trading partner. The trading partner profile in the TPI repository determines which transport method is used to send the message.

Receiving documents

When the TPI Server receives a message from a trading partner, it decrypts the message, verifies the digital signature, and writes the document to the appropriate inbound directory. Once the message is unpacked, the TPI Server returns a Message Disposition Notification (MDN) to the message sender. The MDN verifies only that the message was received and properly unpacked. It does not indicate whether the document was fully processed by the trading partner.

The TPI connector

The connector for TPI enables an integration broker to exchange business objects with TPI Server versions 3.0.3 and higher.

The connector implements business object handling, event polling, and event notification. The application-specific component of the connector generates business objects that it sends to the integration broker; it also responds to business object requests from the integration broker. It generates logging and tracing messages that it writes to a file or the connector console, or sends to the integration broker.

The TPI connector uses the following components:

- WebSphere Business Integration Adapter data handlers — Called by the connector to perform format conversion between the document formats supported by the TPI Server and WebSphere Business Integration Adapter business object format.
- Trading partner configuration file — Stores data handler information for each trading partner.

The TPI connector runs in the same process space, using the same Java Virtual Machine, as the TPI Server.

Note: For this reason it is inadvisable to run your TPI connector with the Parallel Process Degree Resource set to a value greater than 1. For more information on Parallel Process Degree see the *System Administration Guide*.

The connector communicates with the TPI Server by using the DocumentListener interface and the InterchangeEventListener interface. It processes inbound TPI documents to create business objects that it passes to the integration broker. It processes request business objects to create document streams that it passes to the TPI Server. Figure 1 illustrates the TPI connector architecture.

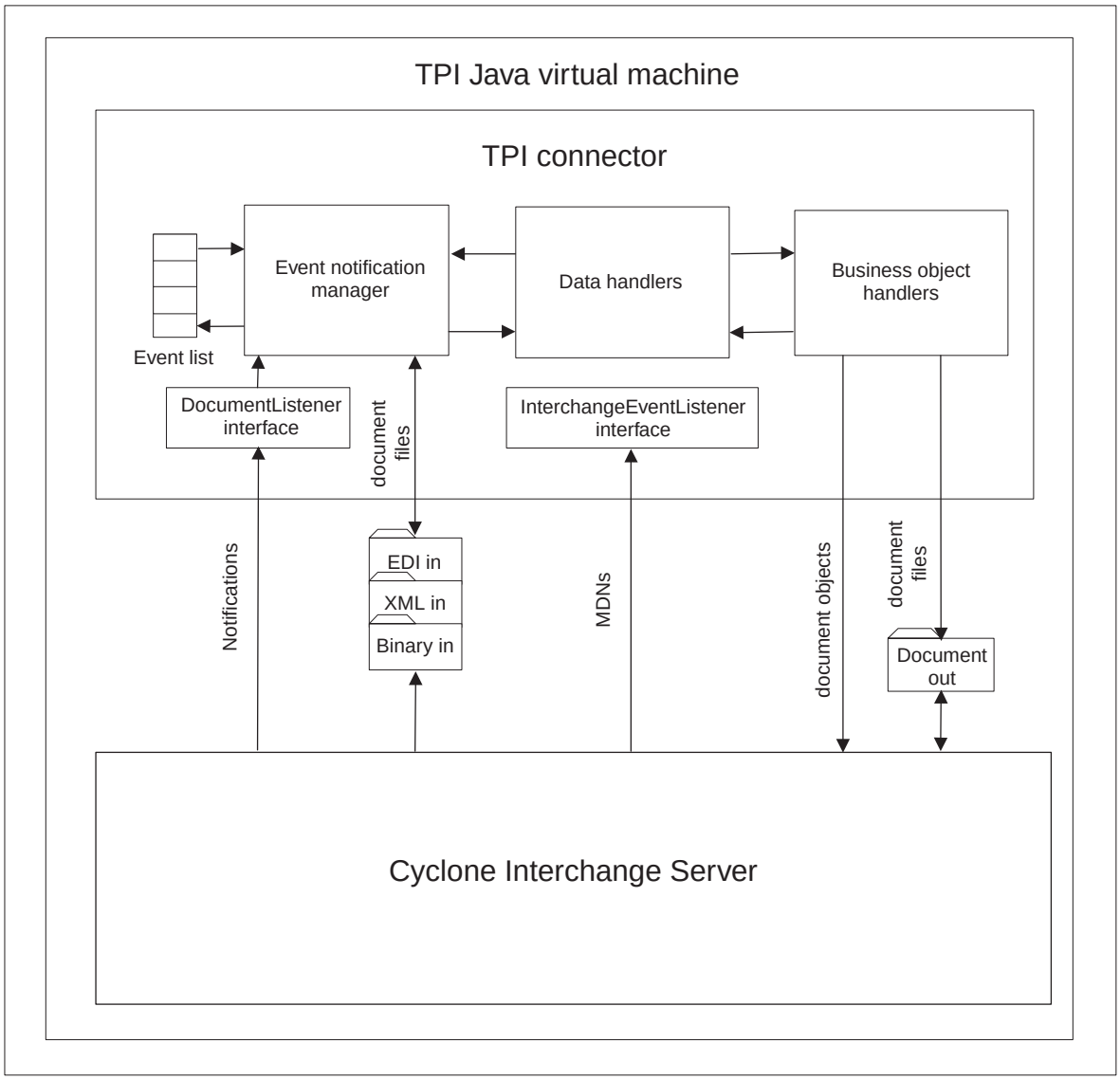


Figure 1. TPI connector architecture

Initialization and termination

In addition to loading the connector configuration properties, the init() method performs the following tasks at startup:

- Reads the trading partner configuration file, and loads data handler information for each trading partner and format combination into memory.
- The connector starts the TPI Server and registers with the DocmentListener, and InterchangeEventListener interfaces for event and request processing.
- If the connector is starting up after a failure, the init() method will perform event recovery tasks as specified by the EventRecovery property.

The terminate() method shuts down the connector and the TPI Server.

Event notification

During event notification, the TPI connector uses the file system to persist meta-data for unprocessed events. It also archives processed events, labeling them with their status.

The connector for TPI uses a callback method, documentArriving(), for event notification. This method is provided through the DocumentListener interface. When the TPI Server receives a document from a trading partner, it calls the documentArriving() method to notify the connector of the inbound document. The notification contains the meta-data necessary for the connector to retrieve and process the document. This includes the following information:

- Sender ID
- Receiver ID
- Document type
- Document filename and path
- Unique document ID

This constitutes the event meta-data. The connector places the event meta-data in an event list in memory, and writes a copy of the event to a file in the event directory with the .event extension for backup and recovery.

Each time the connector polls the event list, it retrieves the oldest event, based on time of arrival to the connector. The connector retrieves the document from the In directory, and uses the trading partner configuration file to determine the document's MIME type.

The connector passes the MIME type to the DataHandler class to create an instance of the appropriate data handler. The connector then calls the data handler to convert the document to a business object. Each data handler uses different criteria to determine the business object name. After the business object is generated, the connector checks for a subscription to the business object. If a subscription exists, the connector passes the object to the subscribing business process. (For subscription information specific to your integration broker, refer to the broker's implementation guide.)

Parallel processing

By default, the TPI connector processes only one event at a time. However, it is capable of processing multiple incoming events simultaneously in order to improve performance. In this situation, each event is allocated to its own thread, and the threads execute simultaneously.

Use connector-specific properties to enable parallel processing; see "Connector-specific properties" on page 14 for more information.

Event recovery

When a connector retrieves an event from the event list, it changes the extension on the corresponding event file to .inprogress until processing is complete, at which time the event file is moved to the archive directory. When restarting after a connector failure, the connector processes all `inprogress` events according to the EventRecovery property setting. If EventRecovery is set to `Reprocess` the connector reads the event directory and restores all `inprogress` events to the event list. Optionally, the connector can be configured to ignore or fail `inprogress` events.

Note: If the connector is terminated unexpectedly during the brief window of time after it receives a `documentArriving` event notification and before it writes the event to a file, the connector cannot recover that event.

Event archiving

The connector maintains an archive directory, where it stores processed events. After an event is processed, the event file is renamed with one of the following extensions:

- archive - the event was successfully processed by the connector.
- unsubscribed - no subscription exists for the event. For subscription information specific to your integration broker, see the broker's implementation guide.
- fail - the connector was unable to process the event and post it to the integration broker.

Business object request processing

All request business objects processed by the TPI connector must contain a configuration child object. This object contains information required by the connector and the TPI Server to process the object. This includes the sender ID, receiver ID, and document type.

When the connector receives a business object, the doVerbFor() method calls a data handler to convert the business object to an appropriately formatted stream. The data handler is called based on the MIME type listed in the trading partner configuration file for the ReceiverID and DocumentType values. TPI parses EDI and XML documents for SenderID, ReceiverID and CycloneID. The data handler outputs a document stream which is written to a file in the document out directory. The connector passes a document object, which references the file, to the TPI Server by calling the sendDocument() method in the TPI Server API. The sendDocument() method returns the unique document ID generated by the TPI Server.

If the WaitForMDN connector property is set to `true`, the connector then waits for the TPI Server to indicate that it has received an MDN from the trading partner. Optionally, the WaitForMDN property can be overridden on a per business object basis by populating the WaitForMDN attribute in the child meta-object. Once an MDN is received, the doVerbFor() method returns a status code (success or error) to the integration broker. If the WaitForMDN property is set to `false`, the connector does not wait for the TPI Server to return an MDN.

Processing locale-dependent data

The connector has been internationalized so that it can support double-byte character sets, and deliver message text in the specified language. When the connector transfers data from a location that uses one character code set to a location that uses a different code set, it performs character conversion to preserve the meaning of the data.

The Java runtime environment within the Java Virtual Machine (JVM) represents data in the Unicode character code set. Unicode contains encodings for characters in most known character code sets (both single-byte and multibyte). Most components in the WebSphere business integration system are written in Java. Therefore, when data is transferred between most WebSphere business integration system components, there is no need for character conversion.

To log error and informational messages in the appropriate language and for the appropriate country or territory, configure the Locale standard configuration property for your environment. For more information on these properties, see Appendix A, “Standard configuration properties for connectors”, on page 47.

Chapter 2. Installing and configuring the connector

This chapter describes how to install and configure the connector component of the IBM WebSphere Business Integration Adapter for Trading Partner Interchange (TPI). It contains the following sections:

Compatibility

The adapter framework that an adapter uses must be compatible with the version of the integration broker (or brokers) with which the adapter is communicating. The 3.3.x version of the adapter for TPI is supported on the following adapter framework and integration brokers:

Adapter framework: WebSphere Business Integration Adapter Framework, version 2.x

Integration brokers:

- WebSphere InterChange Server, version 4.11 and 4.2
- WebSphere MQ Integrator, version 2.1.0
- WebSphere MQ Integrator Broker, version 2.1.0
- WebSphere Application Server Enterprise, version 5.0.1, with WebSphere Studio Application Developer Integration Edition, version 5.0.1

See the Release Notes for any exceptions.

Prerequisites

This section describes the required software components and tasks to be performed before installing the connector.

Installing the TPI Server

The TPI Server must be installed before you install the TPI connector. The connector and the TPI Server must be installed on the same machine. For instructions on installing and configuring the TPI Server, see the *Administrator's Guide* included with the TPI Server.

Setting up accounts and permissions on UNIX

Before installing TPI on a UNIX system, create a user group with both the WebSphere Business Integration Adapter and TPI Server administrators as members. This is required to give the TPI connector access to the TPI Server.

Ensure that both members of the group have read/write permission for the TPI Server installation directory.

Database support

The TPI Server includes a runtime version of Sybase SQL Anywhere as the standard repository. As a runtime version, it provides only limited access for administering and viewing the database. Therefore, it is recommended that you configure either Oracle or Microsoft Server SQL Server as the TPI Repository. For information on how to configure an external database as the TPI repository, see the *Administrator's Guide* included with the TPI Server.

Quick install of the TPI adapter

The following subsections describe how to install the adapter on a UNIX or Windows system.

After your business integration system has been installed, you can install additional adapters from the product CD-ROM at any time. To do this, insert the product CD-ROM, run the installation program, and choose the adapterrs that you want to install.

Note: Unless otherwise indicated, the remaining sections in this chapter apply to both UNIX and Windows installations of the adapter.

Installing on a UNIX system

To install the adapter on a UNIX system, run the Installer for IBM WebSphere Business Integration Adapter and select the IBM WebSphere Business Integration Adapter for TPI.

For more information on installing the connector component, refer to one of the following guides, depending on the integration broker you are using:

- *IBM WebSphere InterChange Server System Installation Guide for UNIX* (when WebSphere ICS is used as the integration broker)
- *IBM WebSphere Business Integration Adapters Implementation Guide for WebSphere MQ Integrator Broker* (when WebSphere MQ Integrator Broker is used as the integration broker)

Table 1 describes the UNIX file structure used by the connector.

Table 1. Installed UNIX file structure for the TPI adapter

Subdirectory of *$ProductDir*	**Description**
connectors/TPI	Contains the connector CWTPI.jar and the start_TPI.sh files. The start_TPI.sh file is a system startup script for the connector. It is called from the generic connector manager script. When you click Install from Connector Configurator (WebSphere MQ Integrator Broker as the integration broker) or the Connector Configuration screen of System Manager (WebSphere ICS as the integration broker), the installer creates a customized wrapper for this connector manager script. When the connector works with WebSphere ICS, use this customized wrapper to start and stop the connector. When the connector works with WebSphere MQ Integrator Broker, use this customized wrapper only to start the connector. Use the mqsiremotestopadapter command to stop the connector.
connectors/messages	Contains the TPIConnector.txt file, as well as the TPIConnector_*ll_TT*.txt files (message files specific to a language (*ll*) and to a country or territory (*TT*).
connectors/TPI/dependencies	Contains DTD files specific to CIDX and RosettaNet.
repository/TPI	Contains the CN_TPI.txt file.
`/lib`	Contains the WBIA. jar file.
`/bin`	Contains the CWConnEnv.sh file.
connectors/TPI/Samples/ WebSphereICS	Contains the sample files that demonstrate how the connector is used with an WebSphere ICS integration broker. The directory contains a Readme.txt file that describes the sample, as well as the following subdirectories: • /IBM--Contains the sample business objects, collaboration templates and objects, connector definitions, and trading partner profiles. • /IBMTP--Contains trading partner profiles and sample data used for polling.
connectors/TPI/Samples/ WebSphereMQIntegratorBroker	Contains the sample files that demonstrate how the connector is used with WebSphere MQ Integrator Broker integration broker. The directory contains a Readme.txt file that describes the sample, as well as the following subdirectories: • /IBM--Contains connector configuration files, sample business objects, sample data, and trading partner profiles. • /IBMTP--Contains trading partner profiles and sample data used for polling.

After you have installed the connector, you must do the following:

1. Use the Connector Configuration Tool to generate the customized connector wrapper (connector_manager_TPI) required to start the connector. Refer to the System Installation Guide for UNIX or the *IBM WebSphere Business Integration Adapters Implementation Guide for WebSphere MQ Integrator Broker* for more information.
2. Log in as root and run the dbenable.sh script in the CycloneInterchange/bin directory.

Installing on a Windows NT system

To install the adapter on a Windows system, run the Installer for IBM WebSphere Business Integration Adapter and select the IBM WebSphere Business Integration Adapter for TPI. All standard files associated with the adapter are installed.

Table 2 describes the Windows NT file structure used by the TPI adapter.

Table 2. Installed Windows NT file structure for the TPI adapter

Subdirectory of *$ProductDir$*	**Description**
connectors\TPI	Contains the connector CWTPI.jar and the start_TPI.bat files.
connectors\messages	Contains the TPIConnector.txt file, as well as the TPIConnector_*ll_TT*.txt files (message files specific to a language (*ll*) and to a country or territory (*TT*).
connectors\TPI\dependencies	Contains DTD files specific to CIDX and RosettaNet.
repository\TPI	Contains the CN_TPI.txt file.
connectors\TPI\Samples\ WebSphereICS	Contains the sample files that demonstrate how the connector is used with the WebSphere ICS integration broker. The directory contains a Readme.txt file that describes the sample, as well as the following subdirectories: • \IBM--Contains the sample business objects, collaboration templates and objects, connector definitions, and trading partner profiles. • \IBMTP--Contains trading partner profiles and sample data used for polling.
connectors\TPI\Samples\ WebSphereMQIntegratorBroker	Contains the sample files that demonstrate how the connector is used with WebSphere MQ Integrator Broker integration broker. The directory contains a Readme.txt file that describes the sample, as well as the following subdirectories: • \IBM--Contains connector configuration files, sample business objects, sample data, and trading partner profiles. • \IBMTP--Contains trading partner profiles and sample data used for polling.
`\lib`	Contains the WBIA. jar file.
`\bin`	Contains the CWConnEnv.bat file.

Installer adds an icon for the connector file to the IBM WebSphere Business Integration Adapters menu. For a fast way to start the connector, create a shortcut to this file on the desktop.

For more information on installing the connector component, refer to one of the following guides, depending on the integration broker you are using:

- *IBM WebSphere InterChange Server System Installation Guide for Windows* (when WebSphere ICS is used as the integration broker)
- *IBM WebSphere Business Integration Adapters Implementation Guide for WebSphere MQ Integrator Broker* (when WebSphere MQ Integrator Broker is used as the integration broker)

or see the section “Comprehensive install and uninstall instructions for adapters” on page 17.

Configuring the TPI Server event logging feature

The TPI connector uses event messages logged by the TPI Sever to monitor the processing of events. The TPI Server must be configured to run in Debug mode so that it will notify the connector of all server events.

Note: If the TPI Server is not run in Debug mode, the connector hangs while waiting for the TPI Server `init()` method to send an event completion message.

To set the TPI Server to run in debug mode:

1. Start the TPI Server Administrator.
2. From the Administrator menu, select Tools > Preferences.
3. In the General tab, set the Event Logging Level to Alert, Notify, Transaction, Debug.

Configuring the connector

Note: Because the TPI adapter and TPI server run in the same address space, it is inadvisable to run your TPI adapter with the Parallel Process Degree Resource set to a value greater than 1. For more information on Parallel Process Degree see the *System Administration Guide*.

Before starting the connector, you must perform the following configuration tasks:

- Create the trading partner configuration file.
- Configure the Connector to Start the TPI Server.
- Set connector configuration properties.

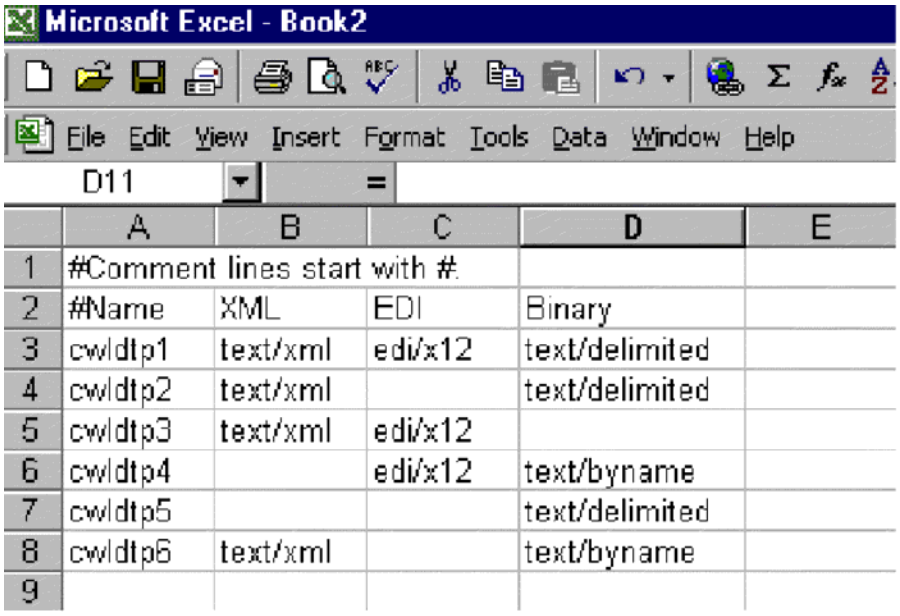
Creating the trading partner configuration file

The trading partner configuration file lists the MIME types used by each trading partner for XML, EDI, and binary formats. Each time the connector processes a document or a business object, it gets the MIME type from the trading partner configuration file. The MIME type is passed to the DataHandler class in order to call the correct data handler for a given document or business object.

The trading partner configuration file is a tab-delimited text file. Each line in the file contains a TPI trading partner ID followed by the MIME types used by that trading partner for XML, EDI and binary formats. The trading partner configuration file format is as follows:

Trading Partner ID <tab> *XML MIME type* <tab> *EDI MIME type* <tab> *binary MIME type*

To ensure proper formatting, create the file as a Microsoft Excel spreadsheet. Then Save the file as Text (Tab delimited). Figure 2 illustrates how to create the file in Excel.



	A	B	C	D	E
1	#Comment lines start with #.				
2	#Name	XML	EDI	Binary	
3	cwldtp1	text/xml	edi/x12	text/delimited	
4	cwldtp2	text/xml		text/delimited	
5	cwldtp3	text/xml	edi/x12		
6	cwldtp4		edi/x12	text/byname	
7	cwldtp5			text/delimited	
8	cwldtp6	text/xml		text/byname	
9					

Figure 2. Create the trading partner configuration file in Excel

The file can be saved to any directory on the machine where TPI is installed. The TradingPartnerConfigurationFile connector property holds the location of the file. For more information see "Connector-specific properties" on page 14.

You can update the trading partner configuration file to add new partners or modify document information while the connector is running. It is not necessary to restart the connector in order to load the updates; the connector retrieves any updates to the configuration file during processing.

Configuring the connector to start the TPI Server

The TPI connector uses the JVM delivered with the TPI Server. At startup, the TPI connector starts the connector launches this JVM. This behavior must be configured after connector installation by editing the connector startup file, start_TPI.bat on Windows NT, and start_TPI.sh on UNIX.

To configure the connector to start up and shut down the TPI Server:

1. Open the start_TPI file located in the \connectors\TPI directory.
2. Change the value of the CYCLONEHOMEDIR attribute to the TPI home directory path.
3. Save and close the file.

In addition to editing the startup file, you must set the following connector properties for the connector to successfully start the TPI Server:

- MetaEventDir
- DocumentOutDir
- TradingPartnerConfigurationFile
- ArchiveProcessedDocDir

See "Connector-specific properties" on page 14 for more information.

Running the connector and TPI Server as a Windows NT service

You can run the TPI connector and server as a Windows NT service. After installing the connector as a service, you must also modify the start_TPI.bat file as follows:

- Replace the *%ProductDir%*\bin\java value with %CYCLONEHOMEDIR%\cijre\bin\java in order to execute the connector from the Cyclone JVM.
- Redirect the output of the JRE process to a file by appending the following to the end of the %CYCLONEHOMEDIR%\cijre\bin\java command line:

```
>"%CONNDIR%"\connectors\TPI\TPITrace.txt
```

Setting connector configuration properties

You must set the connector's standard and connector-specific configuration properties before you can run it. Use one of the following tools to set a connector's configuration properties:

- Connector Configurator (if WebSphere ICS is the integration broker)--Access this tool from the System Manager.
- Connector Configurator (if WebSphere MQ Integrator Broker is the integration broker)--Access this tool from the IBM WebSphere Business Integration Adapter program folder. For more information about Connector Configurator, see Appendix B, "Connector Configurator", on page 63

Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, "Standard configuration properties for connectors", on page 47 for detailed information about these properties.

Important

Because the connector supports both the ICS and WebSphere MQ Integrator Broker integration brokers, configuration properties for both brokers are relevant to the connector.

In addition, refer to Table 3 for configuration information specific to the IBM WebSphere Business Integration Adapter for TPI. The information in this table supplements the information in the appendix

Table 3. Property information specific to this connector

Property	Note
AgentConnections	This connector is single threaded. Therefore, it cannot use the AgentConnections property.
CharacterEncoding	Because this connector is Java-based, it does not use the CharacterEncoding property.

Table 3. Property information specific to this connector (continued)

Property	Note
Locale	Because this connector has been internationalized, you can change the value of the Locale property. See release notes for the connector to determine currently supported locales. **Note:** If you are using WebSphere MQ Integrator BrokerWebSphere MQ Integrator Broker as your broker, you must use the same locale for the adapter, the broker, and any applications.

Connector-specific properties

Connector-specific configuration properties provide information needed by the connector at runtime. Connector-specific properties also provide a way of changing the configuration or logic within the connector without having to recode and rebuild it

Table 4 lists the connector-specific properties for the connector. See the sections that follow for explanations of the properties.

Table 4. Connector-specific configuration properties

Name	Possible values	Default value
ArchiveProcessedDocDir	*<valid directory path>*	
ArchiveProcessedDocInfo	`true` or `false`	true
BackupRequired	`true` or `false`	true
CycloneServerArgs	Any valid TPI Server arguments that do not conflict with other arguments used by the connector.	-appagent; -nogui; -console;
DataEncoding	Any valid encoding name supported by the JVM.	Defaults to the encoding type set by the operating system if no value is specified for this property
DataHandlerDefaultMO		MO_DataHandler_Default
DefaultBinaryMimeType	*<valid MIME type>*	
DefaultEDIMimeType	*<valid MIME type>*	
DefaultVerb	Any verb supported by the connector.	
DefaultXMLMimeType	*<valid DataHandler class name>*	
DeliverOnArrival	`true` or `false`	false
DeliverOnArrivalThreads	Any number between one and ten.	1
DocumentOutDir	*<valid directory path>*	
EventRecovery	`FailOnStartup`, `Reprocess`, `LogError`, or `Ignore`	FailOnStartup
MetaEventDir	*<valid directory path>*	
PollQuantity		25
TradingPartnerConfigurationFile	*<fully qualified filename>*	
WaitForController		500 milliseconds
WaitForMDN	`true` or `false`	true

ArchiveProcessedDocDir: The directory where processed document meta-events are archived. This property is required if the ArchiveProcessedDocInfo property is set to true.

ArchiveProcessedDocInfo: Determines whether the connector retains the meta-event once a document has been successfully processed and sent to the integration broker. If this property is set to false, the meta-events are deleted after processing. The default value is true.

BackupRequired: Determines whether the TPI Server backs up each document after sending it. If set to true, TPI backs up each document after sending. The default value is true.

ControllerWaitCount: Specifies the maximum number of times the connector checks the controller to see if it is active. This property is used in conjunction with the WaitForController property. Each time the connector checks the controller for activity, it waits for the amount of time specified by WaitForController. If the controller does not respond after the maximum number of attempts, event processing fails. The default value is 1.

CycloneServerArgs: A semicolon delimited list of arguments to be passed to the TPI Server at startup. This property is required. The default value is -appagent; -nogui; -console;.

DataEncoding: Specifies the type of data encoding used by the connector. Valid values include any encoding type supported by the JVM. You must set this property whenever non-ASCII character encoding is to be used. If the DataEncoding property is not set, the connector uses the encoding type specified at the operating system level.

DataHandlerDefaultMO: The default name of the meta-object for the connector to use to determine configuration of the data handler for each supported format. This property is required. The default value is MO_DataHandler_Default.

DefaultBinaryMimeType: MIME type for the data handler to use for Binary documents in case the trading partner is not configured in the trading partner configuration file. This property is required if binary MIME type is used in TPI. No default value is set for this property.

DefaultEDIMimeType: MIME type for the data handler to use for EDI documents in case the trading partner is not configured in the trading partner configuration file. This property is required if EDI MIME type is used in TPI. No default value is set for this property.

DefaultVerb: Specifies the verb to be set within an incoming business object, if it has not been set by the data handler during polling.

DefaultXMLMimeType: MIME type for the data handler to use for XML documents in case the trading partner is not configured in the trading partner configuration file. This property is required if XML MIME type is used in TPI. No default value is set for this property.

DeliverOnArrival: Specifies the method of event processing to be used. If DeliverOnArrival is set to false, events are processed one at a time; the PollForEvent thread retrieves an event from memory, processes it, and then retrieves the next event. If DeliverOnArrival is set to true, multiple events can be

processed simultaneously. The number of events that can be processed in parallel is determined by the DeliverOnArrivalThreads property. The default value of DeliverOnArrival is false.

DeliverOnArrivalThreads: Specifies the number of threads allocated to simultaneous event processing. If the DeliverOnArrival property is set to true, set the DeliverOnArrivalThreads property to the number of threads you want to allocate to incoming events. The minimum value for parallel processing is 2; the maximum value is 10. By default, DeliverOnArrivalThreads is set to 1, and parallel processing is not enabled.

DocumentOutDir: The directory location where outbound documents are written temporarily before TPI processes them. In the event of a system failure, the documents can be recovered from this directory. This property is required.

EventRecovery: Determines behavior for event recovery. When restarted, a connector checks the Out Directory for files with the extension .inprogress. If this property is set to FailOnStartup, the connector fails to startup. If set to Reprocess, the connector resubmits these events to the server. If set to LogError, the connector logs an error, but does not shut down. If the set to Ignore, the connector ignores such events. The default value is FailOnStartup.

MetaEventDir: The directory used to persist the TPI event information for recovery purposes. This property is required.

PollQuantity: Specifies the number of events the connector retrieves each time it polls the event list. The default value is 25.

TradingPartnerConfigurationFile: Fully qualified name of the trading partner configuration file. This file contains the MIME types used for documents from each trading partner for Binary, XML, and EDI messages. The MIME type is used to call the correct data handler for each document. This property is required. If the property is not specified the connector will fail to start up.

WaitForController: Specifies the amount of time, in milliseconds, that the connector waits for the controller to become active. This property is used in conjunction with the ControllerWaitCount property. The default value is 500 milliseconds.

WaitForMDN: Determines whether the connector waits for an MDN from the trading partner, or returns from the request thread after passing the document to the TPI Server. The MDN indicates that at the protocol level the send was successfully initiated. The WaitForMDN attribute in the child meta-object can be set to override this property on a per business object basis. The default value is true.

Starting the connector

For information on starting and stopping a connector, see one of the following documents, depending on the integration broker you are using:

- *IBM WebSphere Business Integration Implementation Guide for WebSphere MQ Integrator Broker* (WebSphere MQ Integrator Broker as the integration broker)
- *IBM WebSphere InterChange Server System Administration Guide* (WebSphere ICS as the integration broker)

Special considerations for starting a connector

If you are using Cyclone 4.1 or earlier, you can have difficulty starting the connector for TPI when WebSphere MQ Integrator Broker is used as the broker (WebSphere MQ Integrator Broker integration broker only). If the local connector configuration file contains encrypted property values (for example, a password), the exception java.lang.NullPointerException is thrown during startup and the connector terminates.

To prevent this problem, replace the provider list in the java.security file with the following values:

```
security.provider.1=com.cyclonecommerce.crossworks.provider.Cyclone
security.provider.2=iaik.security.provider.IAIK
security.provider.3=sun.security.provider.Sun
```

On an AIX system, the java.security file is located in the %CYCLONEHOMEDIR%/jre/lib/security directory. On Windows and Solaris systems, the file is located in the %CYCLONEHOMEDIR%/cijre/lib/security directory.

Note: After modifying the security file on an AIX system, you must run the %CYCLONEHOMEDIR%/certloader command.

Comprehensive install and uninstall instructions for adapters

The information in this section describes how to install WebSphere Business Integration Adapters (WBIA).

You install the WBIA product by running a platform-specific executable for the installer. Table 5 lists the installer executable for each operating system. The installer executables are located in the `WebSphereBI` directory on the product CD.

Table 5. Platform-specific executables for WBIA Installer

Operating system	WBIA Installer executable file
Windows	`setupwin32.exe`
AIX	`setupAIX.bin`
Solaris	`setupsolarisSparc.bin`
HP-UX	`setupHP.bin`

You can use the installer executable file to perform the installation in the following ways:

- You can start the graphical installer as described in "Invoking the graphical WBIA Installer" on page 18 and then proceed through the installation wizard to make your selections as described in "Using the graphical WBIA Installer" on page 19.
- You can perform a silent installation as described in "Performing a silent installation" on page 27.

Note: These procedures assume that you are installing from a product CD. If you obtain your software from Passport Advantage, make sure you have downloaded it. Refer to your Passport Advantage information for those downloading instructions.

Note: If you are installing the adapters to communicate with InterChange Server, you must install the broker first. See the installation guide for InterChange Server on the appropriate platform for information on how to install the broker.

Important: Make sure you are logged in as the WebSphere business integration system administrator before you install the adapters. When you install on a UNIX computer, the permissions of the folders and files that are created are set based on the permissions of the user account that performs the installation.

Important: You must not install WBIA as root. The entry that is added to the Object Data Manager (ODM) when installing as root prevents you from using SMIT to uninstall other applications, so you should not install WBIA as root.

Invoking the graphical WBIA Installer

The graphical WBIA installer presents you with a wizard that allows you to make choices about the installation of the WBIA product. Although the installer is Java-based and therefore platform-independent, there are different ways of invoking the installer for each platform. This section describes the approaches for both Windows and UNIX computers.

Invoking Installer in a Windows environment

To invoke Installer in a Windows environment, navigate to the `WebSphereBI` directory of the product CD and execute `setupwin32.exe`.

Invoking Installer in a UNIX environment

The WBIA installer in a UNIX environment is invoked through a `.bin` file specific to the platform, located in the `WebSphereBI` directory. Table 5 on page 17 provides the name of the `.bin` file for each platform.

Follow the steps in one of the following sections to invoke the installer depending on how you are working with the UNIX computer:

- "If you are running CDE on the UNIX computer"
- "If you are connecting to the UNIX computer through X emulation software"

If you are running CDE on the UNIX computer: If you are running the Common Desktop Environment (CDE) and working on the UNIX computer directly then you can navigate to the `WebSphereBI` directory of the product CD and double-click the `.bin` file specific to the operating system.

You can also navigate to the `WebSphereBI` directory of the product CD and execute the `.bin` file at the command line. The following example shows how to do so on a Solaris computer:

```
# ./setupsolarisSparc.bin
```

If you are connecting to the UNIX computer through X emulation software: If you are using a Windows computer to connect to the UNIX computer through X emulation software do the following to invoke the installer:

1. Determine the IP address of the Windows computer that you are using to connect to the UNIX computer.

   You can execute the `ipconfig` command at the Windows command line interface to display the IP address of the Windows computer.

2. Set the DISPLAY environment variable on the UNIX computer to the IP address determined in step 1 on page 18.

   You must be sure to follow the IP address with a colon and the identifier for the monitor or display on the Windows client computer. If the Windows client computer only has a single monitor then the display value is 0.0.

   The following example shows the DISPLAY environment variable being set to the single monitor on a Windows computer whose IP address is 9.26.244.30:

   ```
   DISPLAY=9.26.244.30:0.0
   ```

3. Export the DISPLAY environment variable by executing the following command:

   ```
   export DISPLAY
   ```

4. Start the X emulation client on the Windows computer and connect to the UNIX computer.
5. Navigate to the WebSphereBI directory of the product CD at the command line of the X emulation client.
6. Execute the .bin file specific to the operating system. For example, if the UNIX computer was running AIX then you would execute the following command:

   ```
   # ./setupAIX.bin
   ```

   The graphical installer starts on the Windows computer that you are using to connect to the UNIX computer.

Using the graphical WBIA Installer

Once running, the WBIA Installer prompts you to make your installation choices and then performs the installation.

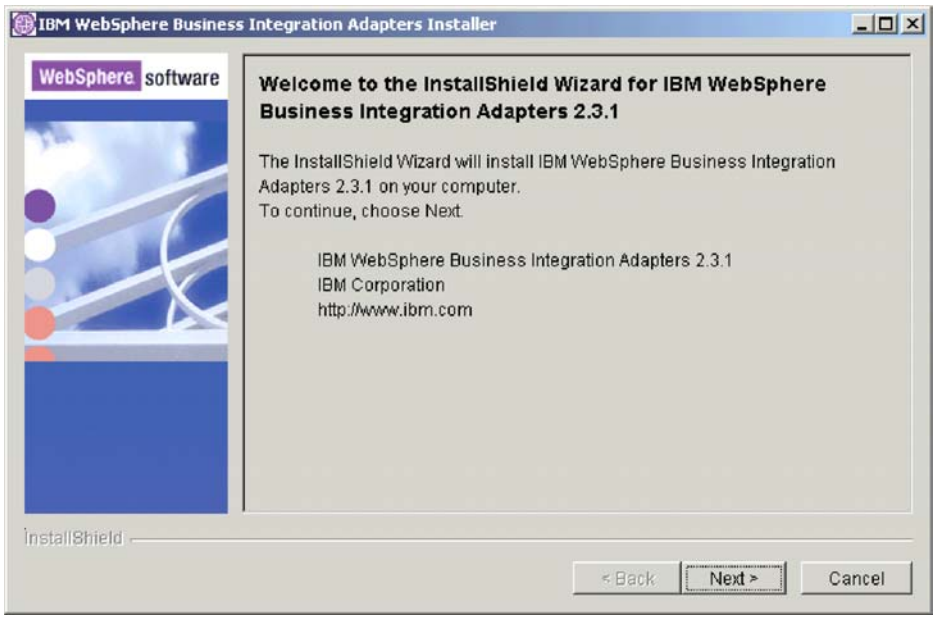
When you select an adapter to install, the runtime components and the data handlers required to support it are automatically selected for installation. However, if Installer determines that current versions of the runtime components are already on the system, then they are not installed.

Note: If you click **Cancel** while the business integration adapters are installing, some files will still appear in the newly created directory. The number of files depends on how far the process had progressed before installation was canceled.

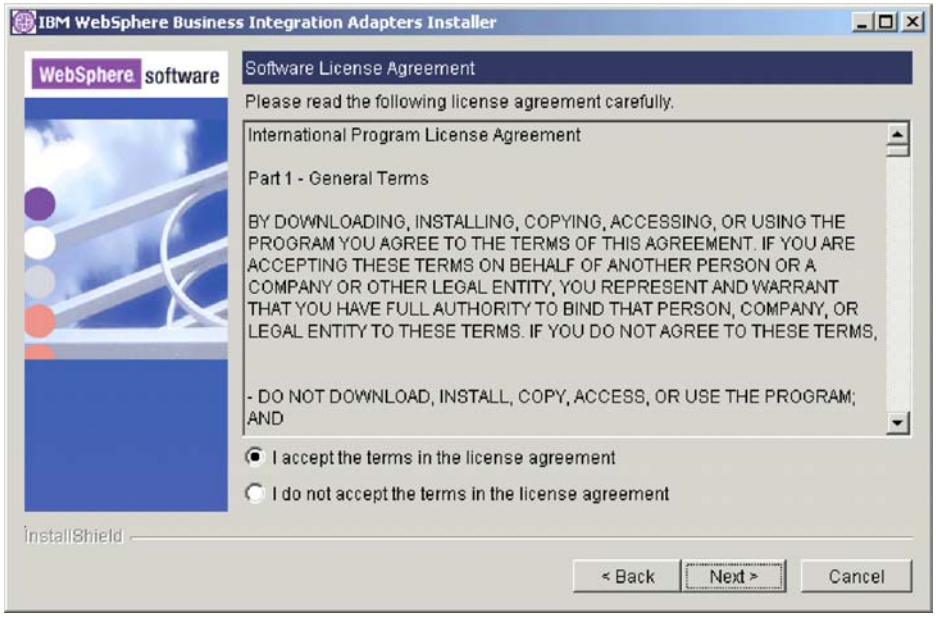
Do the following to proceed through the installer:

1. At the "language selection" prompt, choose the desired language from the drop-down menu and click **OK**.

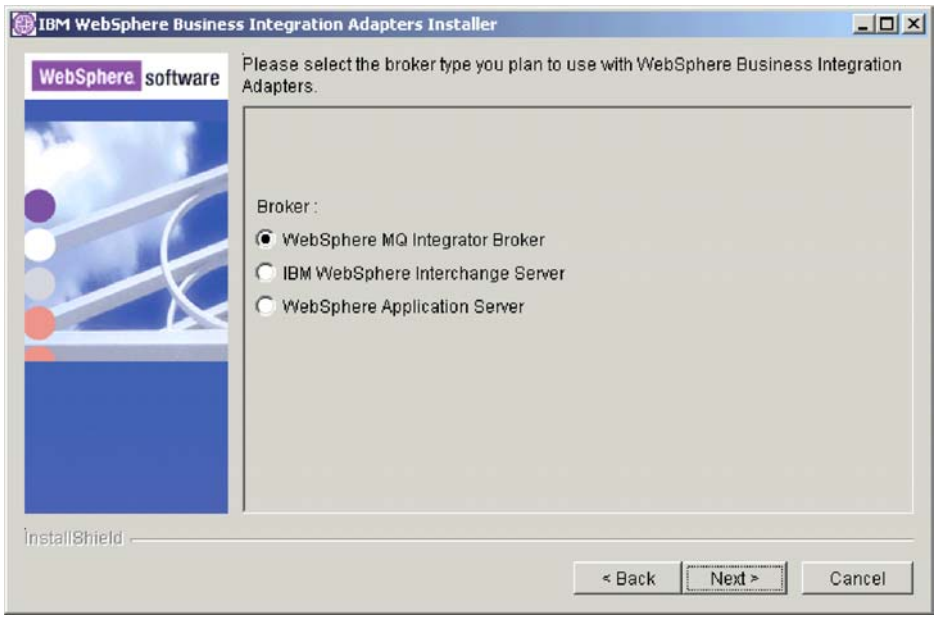
2. At the "Welcome" screen click **Next**.



3. At the "IBM License Acceptance Panel", click **I accept the terms in the license agreement** and then click **OK**.



4. At the "broker selection" screen click the radio button for the type of integration broker with which the adapters will communicate and then click **Next**.



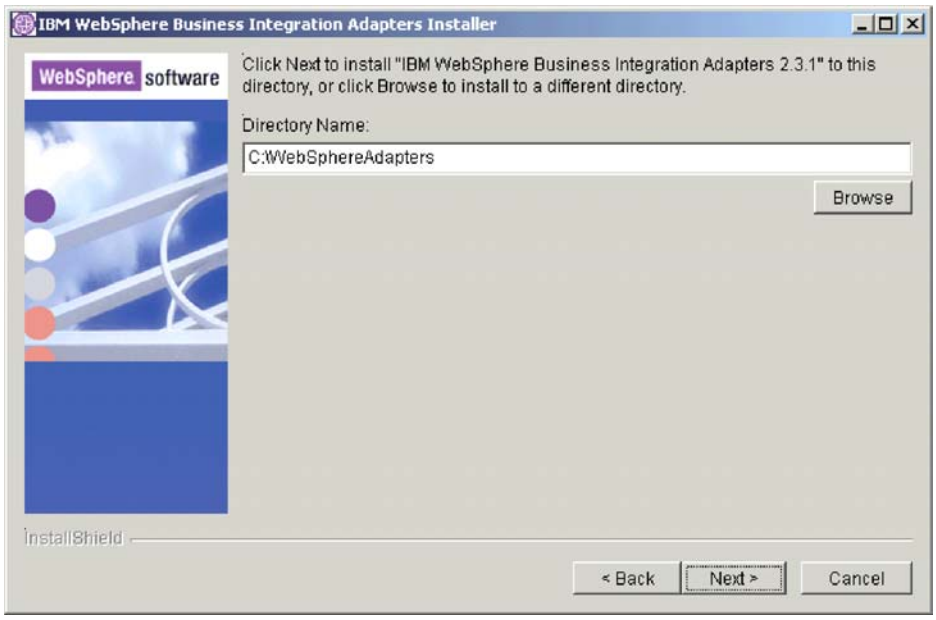
5. At the "installation directory" screen, either type the full path of the directory into which the adapters should be installed, click **Browse** to select a directory, or accept the default path, then click **Next**.

   Important: You must specify an installation directory that does not have spaces in the path.

   Table 6 shows the default adapter installation directories for the different integration brokers on the different supported platforms.

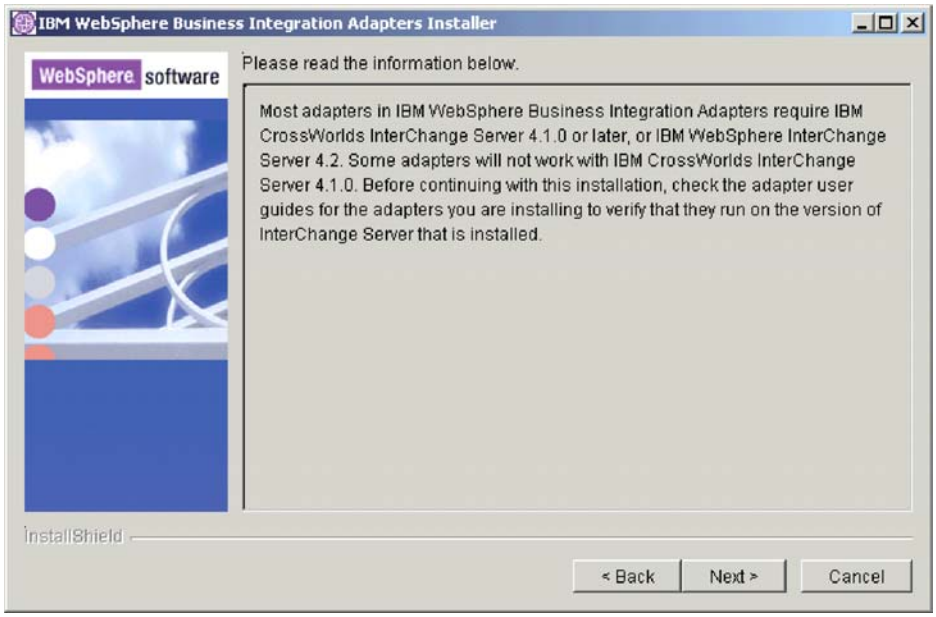
   Table 6. Default adapter installation directories

Broker type	Default Windows directory	Default UNIX directory
WebSphere MQ Integrator Broker	`C:\WebSphereAdapters`	`/$HOME/WebSphereAdapters`
WebSphere InterChange Server	`C:\IBM\WebSphereICS`	`/$HOME/IBM/WebSphereICS`
WebSphere Application Server	`C:\WebSphereAdapters`	`/$HOME/WebSphereAdapters`

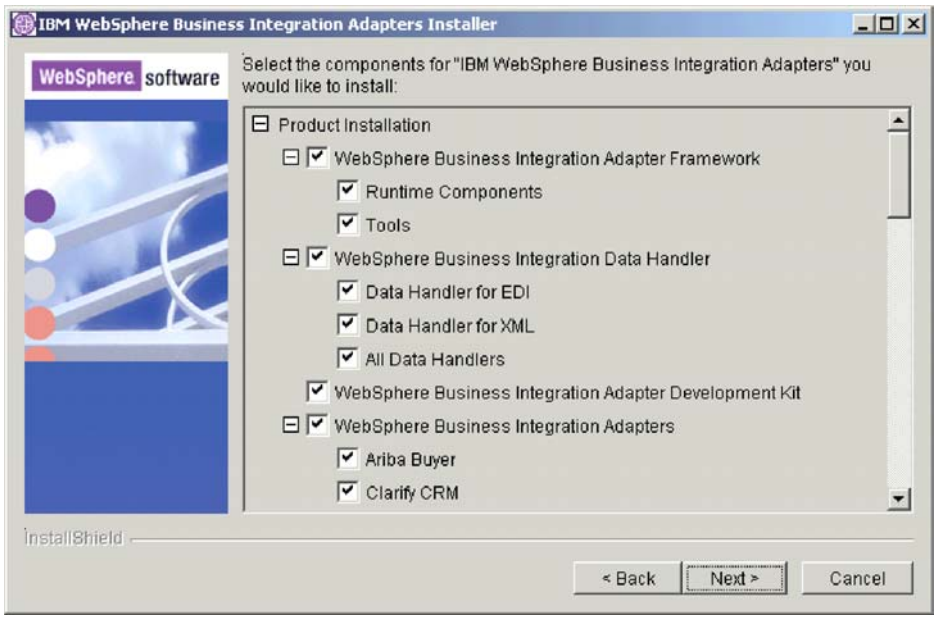


Important: If your broker is WebSphere InterChange Server then you must specify the WebSphere InterChange Server product directory at this screen. If you do not specify the same directory for the adapter installation as for the broker installation then the adapters will not be able to run.

6. If you selected WebSphere InterChange Server as your broker in step 20 then Installer presents an informational screen at this time. Read the information and then click **Next**.



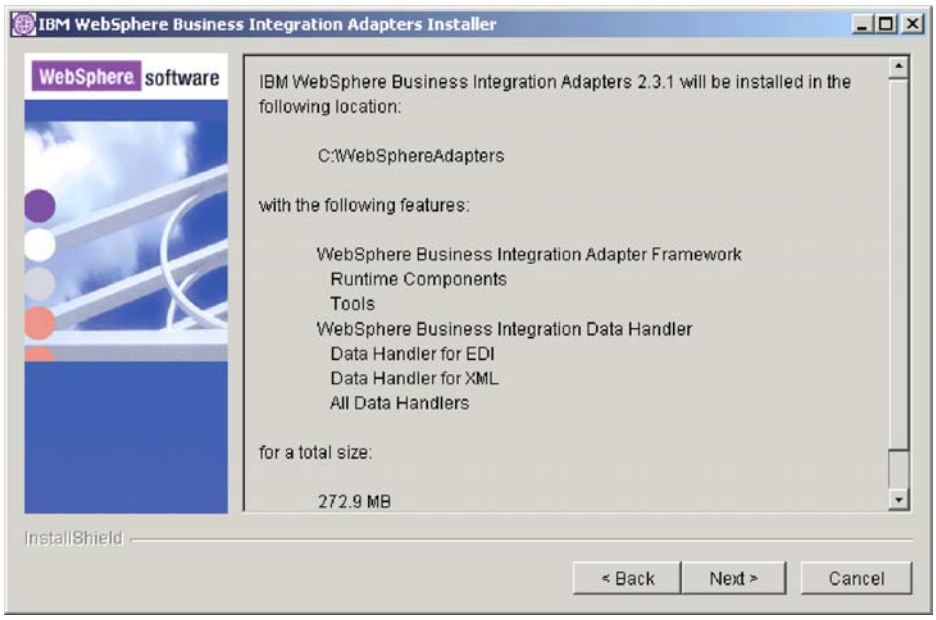
7. At the "component selection" screen, enable checkboxes for the adapters and features you want to install and then click **Next**.



Different features are available for installation depending on the broker. For instance, if you chose WebSphere MQ Integrator Broker or WebSphere Application Server as your broker then the adapter runtime and tools are available for selection. If you chose WebSphere InterChange Server, though, these are not available because the tools and runtime are installed as part of the broker installation.

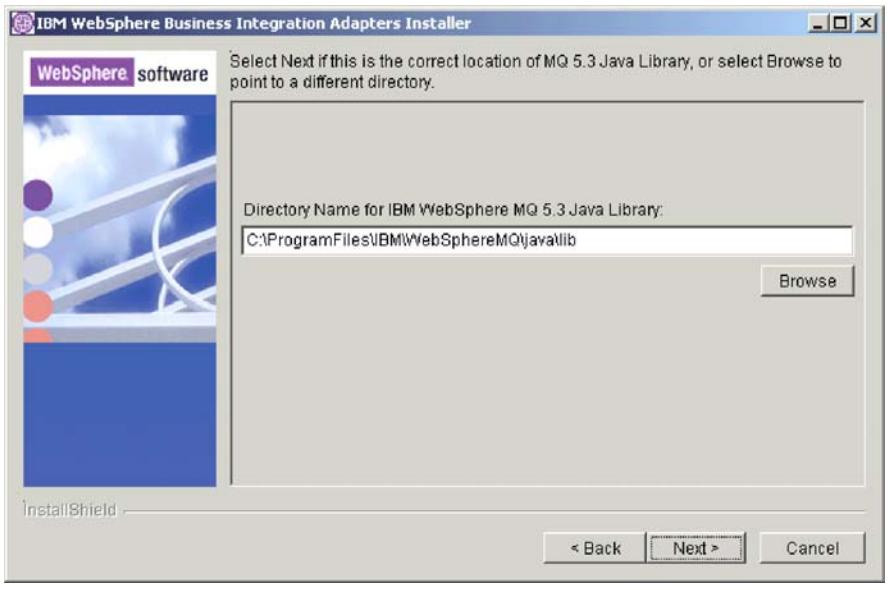
Different features are also available for installation depending on the operating system. For example, the tools are only supported in Windows environments and are therefore only available for selection when you run the installer on a Windows computer.

8. The "summary" screen lists the features selected for installation, the specified product directory, and the amount of disk space required. Read the information to verify it and then click **Next**.



9. Do the following depending on which broker you selected during step 4:
   - If you selected WebSphere MQ Integrator Broker or WebSphere Application Server as your broker in step 4 then Installer presents the "MQ Java library directory" screen.

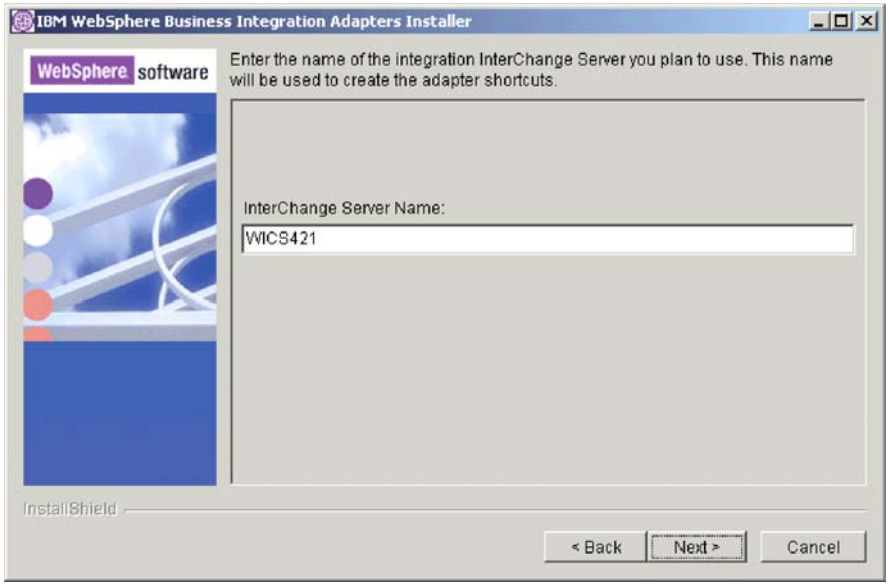
     Either type the directory path to the `java\lib` directory of the WebSphere MQ installation or click **Browse** to select the directory, then click **Next**.



   - If you selected WebSphere InterChange Server as your broker in step 20 then Installer presents the "InterChange Server name" screen at this time. Type the name of the InterChange Server instance with which the adapters will communicate in the **InterChange Server Name** field and then click **Next**.

     Important: Be sure to specify the name of the InterChange Server instance accurately. The server name is case-sensitive and if it is not

typed correctly then the connectors will not be able to communicate with the server.



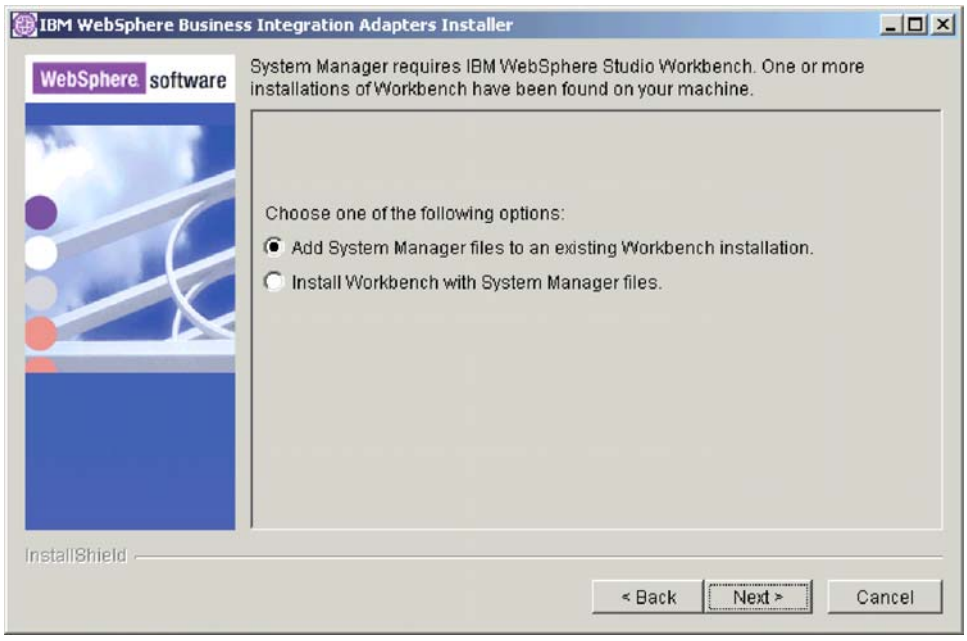
10. If the following are true then Installer next presents the "workbench selection" screen where you specify how you want the tools required to work with the integration components installed:
    - If you are installing on a Windows computer
    - If you selected WebSphere MQ Integrator Broker or WebSphere Application Server as your integration broker
    - If you chose the **Tools** feature to be installed
    - If WebSphere Studio Application Developer Integration Edition version 5.0.1 is installed on the computer

    Choose **Add System Manager files to an existing Workbench installation** if you want to add the plug-ins to an existing installation of WSADIE 5.0.1. You will be required to specify the product directory of the existing installation of WSADIE 5.0.1 at the next Installer screen, as described in step 11 on page 26.

    Important: The plug-ins will not work if there are spaces in the path of the workbench product directory. If there are spaces in the path of your existing WSADIE 5.0.1 installation then you must not make this installation choice.

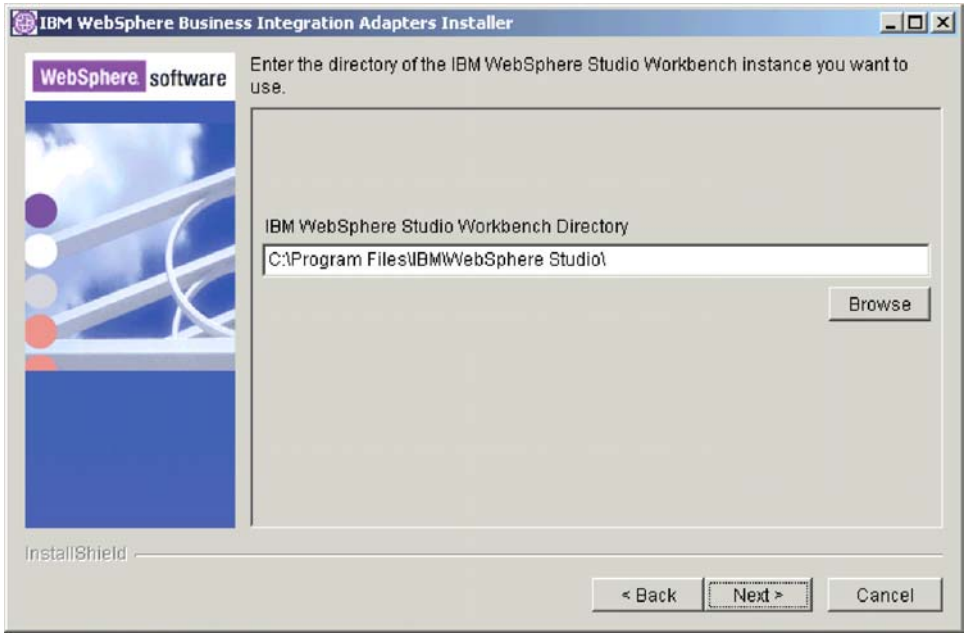
    Choose **Install Workbench with System Manager files** if you want to install WebSphere Studio Workbench 2.0.3 and add the plug-ins to that installation. The workbench is installed in a subdirectory of the adapter installation directory named `Tools\WSWB203` if you make this selection.

    Click **Next** after making your selection.



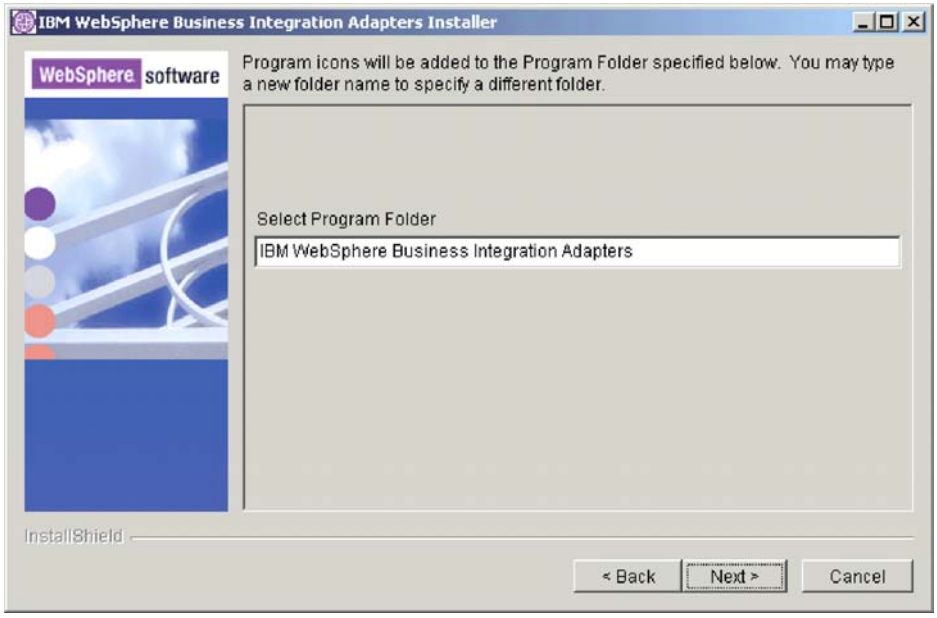
11. If you were presented with the "workbench selection" screen as described in step 10 on page 25 and you chose **Add System Manager files to an existing Workbench installation** then Installer next presents the "workbench installation directory" screen. Type the path to the WSADIE 5.0.1 product directory or click **Browse** to select the directory, then click **Next**.

    Important: The plug-ins will not work if there are spaces in the path of the workbench product directory. If there are spaces in the path of your existing WSADIE 5.0.1 installation then you must click **Back** and select **Install Workbench with System Manager files** instead.



12. If you are installing on a Windows computer then installer next presents the "select program folder" screen. Type the name of the program group in which

shortcuts for the adapters should be created or accept the default program group and then click **Next**.



13. After Installer finishes successfully, click **Finish**.

Performing a silent installation

You can perform a silent installation of WBIA, where you provide your installation choices in a file rather than at the screens of the installation wizard. This is particularly helpful when you have to perform multiple installations that are identical.

To perform a silent installation, first create a file with your installation choices as described in "Preparing the installation response file" and then perform the installation using the file as described in "Executing the silent installation" on page 32.

Preparing the installation response file

When performing a silent installation, you prepare a response file that contains your installation choices. IBM provides four response file templates that contain the options for different combinations of integration broker and platform support. The response file templates are located in the `WebSphereBI` directory on the product CD. Table 7 lists the response file templates for each combination.

Table 7. Response files templates

Integration broker and platform combination	Response file template
WebSphere InterChange Server on Windows:	`settings_WBIAInstaller_ICS.txt`
WebSphere Application Server and WebSphere MQ Integrator Broker on Windows:	`settings_WBIAInstaller_WSMQ.txt`
WebSphere InterChange Server on UNIX:	`settings_WBIAInstaller_ICS_unix.txt`
WebSphere Application Server and WebSphere MQ Integrator Broker on UNIX:	`settings_WBIAInstaller_WSMQ_unix.txt`

Table 8 lists the options that can be used during a silent installation. Be sure to read all the information in the **Option values** column, as it specifies when particular options should be commented out and what the broker and platform compatibilities are.

Table 8. Silent installation options

Option name	Option values
`brokerType.brokerSelection`	Set to 1 for WebSphere MQ Integrator Broker. Set to 2 for WebSphere InterChange Server. Set to 3 for WebSphere Application Server.
`setWINICSDest.value`	Set to the directory path in which InterChange Server is installed. If you leave this option commented, the product installs to the default directory listed in Table 6 on page 21. This option is only relevant for the WICS broker when installed on Windows. Make sure it is commented out when installing for the WMQI or WAS brokers, or when installing on a UNIX computer.
`setWINWMQIDest.value`	Set to a valid directory path, without spaces, to specify the location in which the WBIA product should be installed. If you leave this option commented, the product installs to the default directory listed in Table 6 on page 21. This option is only relevant for WMQI and WAS brokers when installed on Windows. Make sure it is commented out when installing for the WICS broker, or when installing on a UNIX computer.
`setUnixICSDest.value`	Set to the directory path in which InterChange Server is installed. If you leave this option commented, the product installs to the default directory listed in Table 6 on page 21. This option is only relevant for the WICS broker when installed on UNIX. Make sure it is commented out when installing for the WMQI or WAS brokers, or when installing on a Windows computer.
`setUnixWMQIDest.value`	Set to a valid directory path, without spaces, to specify the location in which the WBIA product should be installed. If you leave this option commented, the product installs to the default directory listed in Table 6 on page 21.. This option is only relevant for WMQI and WAS brokers when installed on UNIX. Make sure it is commented out when installing for the WICS broker, or when installing on a Windows computer.

Table 8. Silent installation options (continued)

Option name	Option values
`-P main_product.active`	Set to `true` to install all of the components available in the WBIA product. Set to `false` to install selected components.
`-P f_connectors.active`	Set to `true` to install all adapters. Set to `false` to install specified adapters.
`-P adaptorFrameWork.active`	Set to `true` for adapters to work with WMQI and WAS brokers. Comment out this option when installing for the WICS broker.
`-P WindowEnvSetup.active`	Set to `true` for adapters to work with WMQI and WAS brokers. Comment out this option when installing for the WICS broker or when installing on UNIX.
`-P support.active`	Set to `true` for adapters to work with WMQI and WAS brokers on UNIX. Comment out this option when installing for the WICS broker or when installing for the WMQI or WAS brokers on Windows.
`-P supportFileRequirement.active`	Set to `true` for adapters to work with WMQI and WAS brokers on UNIX. Comment out this option when installing for the WICS broker or when installing for the WMQI or WAS brokers on Windows.
`-P frameworkTOOLS.active`	Set to `true` when WMQI is the broker and Windows is the platform to install the tools used to work with the installed components. Set to `false` if WICS or WAS are the broker, if the platform is UNIX, or if you do not want to install the tools.
`-P adaptorDevelopmentKit.active`	Set to `true` when WMQI is the broker and Windows is the platform to install the Adapter Development Kit, which provides the interfaces necessary to develop custom adapters. Set to `false` if WICS or WAS are the broker, if the platform is UNIX, or if you do not want to install the Adapter Development Kit.

Table 8. Silent installation options (continued)

Option name	Option values
-P cn<*adaptervalue*>.active	Set to `true` to install a specific adapter, where *adaptervalue* is the value that identifies the adapter. For example, `cnJDBC` identifies the Adapter for JDBC. Refer to the response file templates for the available adapters and the values associated with them. Not all adapters are available for all brokers and platforms. Refer to the response file templates to determine if a particular adapter is available for that combination of broker and platform.
-P dataHandler.active	Set to `true` to enable installation of data handlers.
-P <*datahandlervalue*>.active	Set to `true` to install a specific data handler, where *datahandlervalue* is the value that identifies the data handler. For example, `xmlDataHandler` identifies the XML data handler. Refer to the response file templates for the available data handlers and the values associated with them. Installer evaluates any data handler dependencies for selected adapters, so you do not have to specify data handlers to be installed just to satisfy adapter requirements. You would need to select data handlers for installation if you plan to use them for adapters you plan to develop and if the data handlers will not be installed by default.
-P AllDataHandlers.active	Set to `true` to install all data handlers. Set to `false` to install specified data handlers.
-W mqDirectoryUserInput.mqLibraryLocation	Set to the path of the `java\lib` directory within the WebSphere MQ installation on the computer when installing for either the WMQI or WAS brokers. Comment out this option when installing for the WICS broker. Comment out this option if you want to use the default values. On the Windows platform, installer searches the Windows registry by default and uses the value it finds there. On the AIX platform, installer uses the default value `/usr/mqm/java/lib`. On the Solaris and HP-UX platforms, installer uses the default value `/opt/mqm/java/lib`.
-W workbenchChoice.workbenchList	Set to `1` to add the tools plugins to an existing installation of WebSphere Studio Workbench or WebSphere Studio Application Developer Integration Edition. Set to `2` to install WebSphere Studio Workbench.

Table 8. Silent installation options (continued)

Option name	Option values
`-W workbenchLocation.workbenchLocDirectory`	If you specified that WebSphere Studio Workbench be installed by setting the value of the `workbenchChoice.workbenchList` option to `2`, then set this option to the directory in which the workbench should be installed, for example `C:\WebSphereAdapters\Tools`.
`-W inputServer.name`	Set to the name of the InterChange Server instance with which the installed adapters will communicate. The name is case-sensitive and must be specified accurately or the adapters will not be able to communicate with the server. You must make sure this option is not commented and is set to the proper value when installing adapters to communicate with the WICS broker. Comment out the option when installing for WMQI or WAS brokers.
`-W inputShortcuts.folder`	Set to the name of the program group created for the WBIA product, for example `IBM WebSphere Business Integration Adapters`. This option is only relevant on the Windows platform. Comment out this option when installing on UNIX.
`-W createReposFile.active`	Set to `true` to create a file that contains the definitions for the components selected for installation. Set to `false` to not create a file containing the definitions. The definition files for individual components are still copied to the `repository` directory even if you choose not to create this comprehensive file.
`-G replaceExistingResponse`	Set to `yesToAll` or `yes` to replace all files found on the system that have the same name as those being copied by the installer. Set to `noToAll` or `no` to not replace any files found on the system that have the same name as those being copied by the installer.
`-G replaceNewerResponses`	Set to `yesToAll` or `yes` to replace all files found on the system that are newer than those being copied by the installer. Set to `noToAll` or `no` to not replace any files found on the system that are newer than those being copied by the installer.
`-G createDirectoryResponse`	Set to `yes` to create the product directory specified by the option if it does not already exist. Set to `no` to not have the product directory created if it does not exist. You must set this option to `yes` if the specified directory does not exist for the installation to succeed.
`-G removeExistingResponse`	This option is not relevant for any broker on any platform. Comment out this option.

Table 8. Silent installation options (continued)

Option name	Option values
`-G removeModifiedResponse`	This option is not relevant for any broker on any platform. Comment out this option.

You can modify one of the response file templates and use it for the silent installation. In this case you should be sure that any options that are incompatible with others you require are commented out by placing a hash symbol # in front of them. Alternatively, you can create a new response file with the options you require. This approach has the benefit that no unnecessary options or commented description blocks are present to clutter the response file, making it easier to read and edit. If you take this latter approach, it is recommended that you copy a template file and remove the sections and options that are not necessary, rather than typing the necessary options into a new file.

Executing the silent installation

To execute the silent installation, you run the platform-specific installer executable at the command line with several options, including the name of the response file you prepared.

The following example shows how to do so on a Windows computer where the response file template for the WAS and WMQI integration brokers is used, and exists in the `C:\data` directory:

```
D:\WebSphereBI>setupwin32.exe -silent -options
C:\data\settings_WBIAInstaller_WSMQ.txt
```

The following example shows how to do so on an AIX computer where a custom response file named `install.txt` was used, and exists in the `/home/icsadmin` directory:

```
# ./setupAIX.bin -silent -options /home/icsadmin/install.txt
```

Installing additional adapters

To install additional adapters, run the WBIA Installer again. The word `(installed)` is displayed in the component selection screen next to all components that have already been installed on the system.

WBIA directories, files, and environment variables

Installer creates a number of directories, files, and environment variables depending on the selections you made while running Installer.

WBIA directories and files

After the installation is complete, you can view the file system and its contents. Table 9 lists some of the directories you might need to know about. The folders and files created vary depending on the choices made during the installation and on the operating system.

Table 9. WebSphere Business Integration Adapter directories

Directory name	Contents
`_jvm231`	This directory contains the Java Runtime files. **Note:** If you are upgrading from a previous version of WBIA, the existing directory name is retained from that release.

Table 9. WebSphere Business Integration Adapter directories (continued)

Directory name	Contents
`_uninst_WBIA2.3.1`	This directory contains files that are required to uninstall the WebSphere Business Integration Adapters product. For more information on uninstalling WBIA, see "Uninstalling WebSphere Business Integration Adapters" on page 34. **Note:** If you are upgrading from a previous version of WBIA, the existing directory name is retained from that release.
`bin`	This directory contains the executable files and shell scripts that the business integration adapters use.
`connectors`	This directory contains files specific to each adapter in the system. It also contains adapter-specific files that you may need to install in the application that the adapter supports.
`DataHandlers`	This directory contains the `.jar` files for the data handlers.
`DevelopmentKits`	This directory contains the files needed to develop connectors.
`legal`	This directory contains the license files.
`lib`	This directory contains shared libraries and `.jar` files for the system.
`logs`	This directory is provided to contain log and trace files.
`messages`	This directory contains message text files used by the connectors to generate log and trace messages.
`ODA`	This directory contains the `.jar` and `.bat` files for each Object Discovery Agent.
`repository`	This directory contains the connector definition files.
`templates`	This directory contains sample script files for creating and clearing WebSphere MQ queues. If your integration broker is WebSphere MQ Integrator Broker or WebSphere Application Server, see your broker implementation guide for more information on using these scripts. If your integration broker is WebSphere InterChange Server, see the installation guide for InterChange Server on the appropriate platform.
`Tools`	This directory contains the `WSWB203` directory, which in turn contains the installation of WebSphere Studio Workbench if you chose to install it.

Environment variables

If you chose WebSphere MQ Integrator Broker or WebSphere Application Server as your broker, Installer takes the actions described in Table 10 on page 34 to create and update environment variables on the computer. These actions are not taken if you chose WebSphere InterChange Server as your integration broker, because the environment variables required for that broker are created during installation of the broker itself.

Table 10. Actions taken by Installer for environment variables

Environment variable name	Installer action
CROSSWORLDS	Creates this environment variable to reference the WBIA product directory, as specified when using Installer.
MQ_LIB	Creates this environment variable to contain the path to the Java\lib directory within the WebSphere MQ installation, as specified when using Installer.
CLASSPATH	Adds the following entries: *ProductDir*\lib\rt.jar; *ProductDir*\DataHandlers\CwDataHandler.jar;
PATH	Adds the following entries: *ProductDir*\bin\hotspot; *ProductDir*\bin\classic; *ProductDir*\bin;

Uninstalling WebSphere Business Integration Adapters

The method for uninstalling WBIA or selected adapters depends on how you installed the product.

If you installed the WBIA product from CD, follow the directions in "Uninstalling adapters installed from CD".

If you installed the WBIA product using Electronic Software Delivery (ESD), follow the directions in "Uninstalling adapters installed from ESD" on page 37.

Uninstalling adapters installed from CD

You can uninstall adapters that were installed from a CD in the following ways:

- You can start the graphical uninstaller as described in "Invoking the graphical WBIA Uninstaller" and then proceed through the uninstallation wizard to make your selections as described in "Using the graphical WBIA Uninstaller" on page 35.
- You can perform a silent installation as described in "Performing a silent uninstallation" on page 36.

Important: The WBIA Uninstaller does not remove adapters installed from an ESD download. For information on removing adapters installed from an ESD download, see "Uninstalling adapters installed from ESD" on page 37.

Invoking the graphical WBIA Uninstaller: Follow the steps in one of the following sections to invoke the installer depending on the platform on which the WBIA product is installed:

- "To invoke the uninstaller in a Windows environment"
- "To invoke the uninstaller in a UNIX environment" on page 35

To invoke the uninstaller in a Windows environment: Navigate to the *ProductDir*/_uninst_WBIA2.3.1 directory and execute the WBIA Uninstaller, uninstaller.exe.

To invoke the uninstaller in a UNIX environment: Navigate to the *ProductDir*/_uninst_WBIA2.3.1 directory and execute the WBIA Uninstaller, uninstaller.bin.

If you are running the Common Desktop Environment and are working directly on the UNIX computer then you can double-click the uninstaller.bin file.

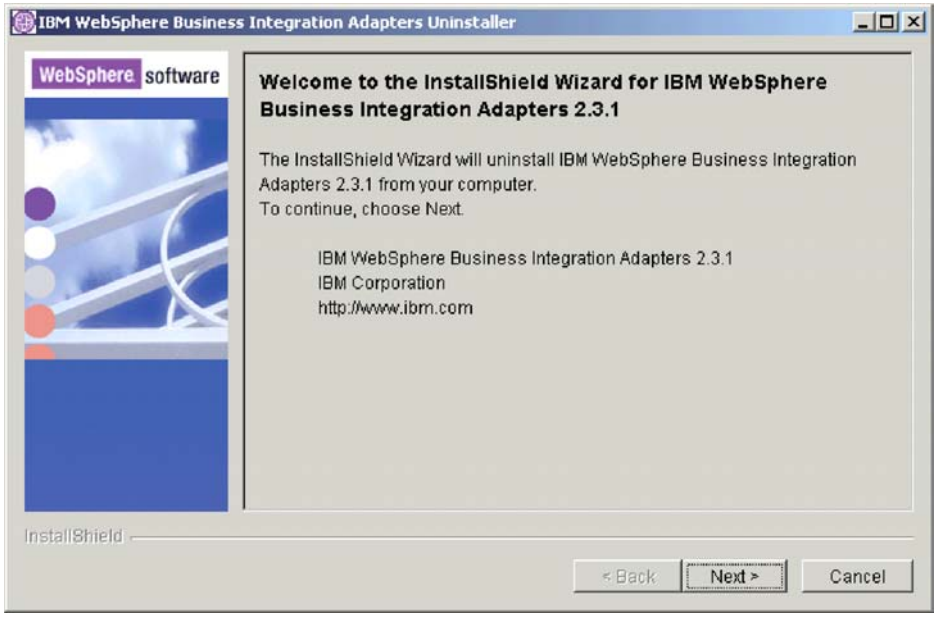
If you are using X emulation software to connect to the UNIX computer from a Windows computer then you must execute the uninstaller.bin file from the command line, as in the following example:

```
# ./uninstaller.bin
```

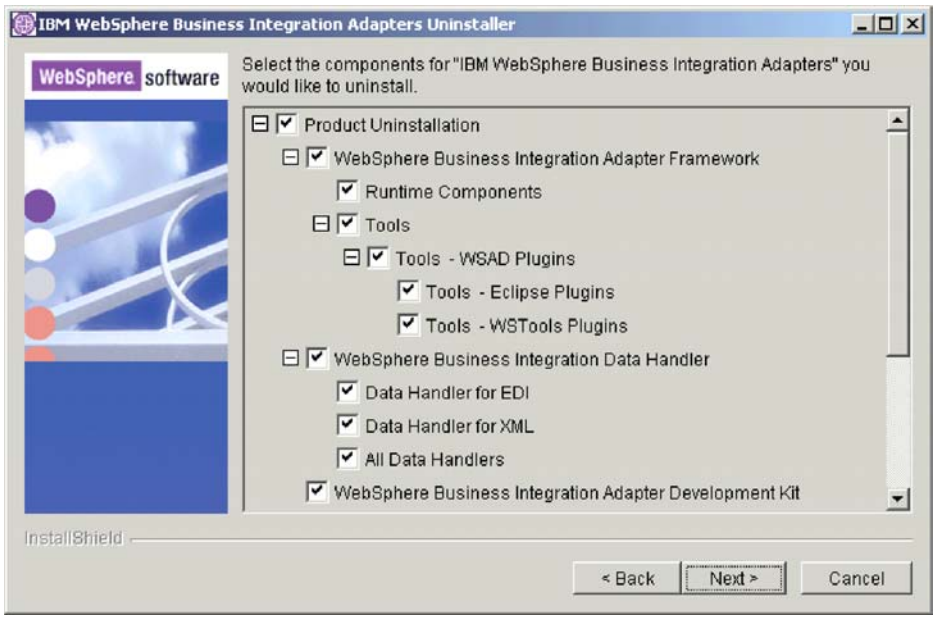
Using the graphical WBIA Uninstaller: Using the WBIA Uninstaller, do the following to uninstall either the entire system or selected components:

1. At the language selection prompt, choose the desired language from the drop-down menu and click **OK**.

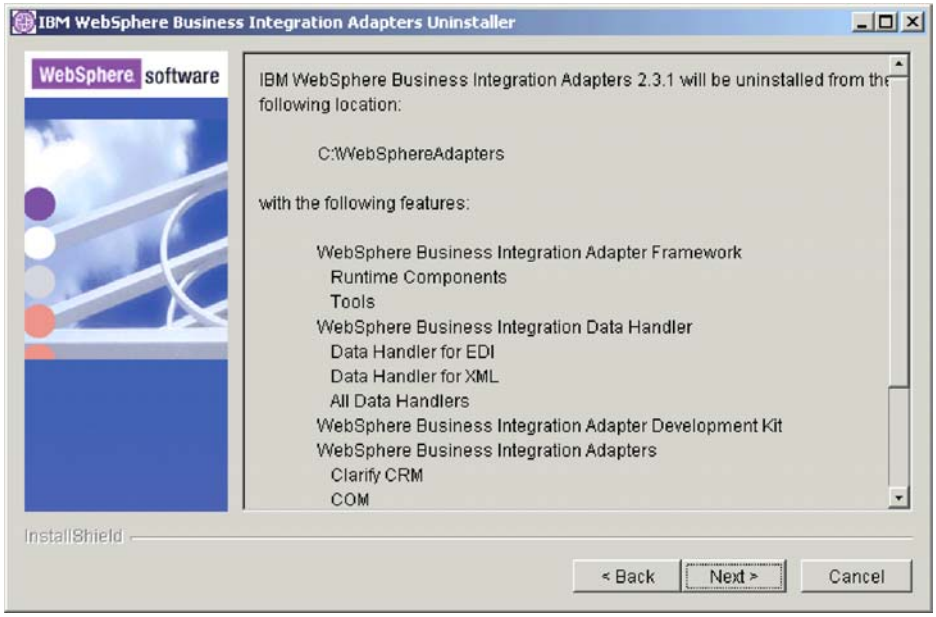
2. At the "Welcome" screen click **Next**.



3. At the "component selection" screen, ensure that the components you want uninstalled have checkboxes next to them. By default all WBIA product components are selected for uninstallation; clear the checkbox for any component you want to leave installed, and then click **Next**.



4. The “summary” screen lists the components that will be uninstalled and the product directory from which they will be removed. Read the information to verify it and then click **Next**.



5. After Uninstaller finishes successfully, click **Finish**.

Performing a silent uninstallation: To perform a silent uninstallation, you run the platform-specific uninstaller executable in the `_uninst_WBIA2.3.1` directory within *`ProductDir`* at the command line with the `-silent` option.

The following example shows how to do so on a Windows computer if the WBIA product is installed in `C:\WebSphereAdapters`:

```
C:\WebSphereAdapters\_uninst_WBIA2.3.1>uninstaller.exe -silent
```

The following example shows how to do so on an AIX computer:

```
# ./uninstaller.bin -silent
```

Note: If you upgraded from a previous version of WBIA, the directory _uninst_WBIA2.3.1, was not created. The old _uninst_WBIA2.x.x uninstallation directory is retained but now contains upgraded contents.

Note: Silent uninstallation does not uninstall adapters installed from an ESD. For information on how to uninstall adapters that were installed from an ESD, see "Uninstalling adapters installed from ESD".

Uninstalling adapters installed from ESD

If you installed a WebSphere Business Integration Adapters adapter using Electronic Software Delivery (ESD), the uninstaller is in a directory labeled _uninstXXX , where XXX identifies the adapter it uninstalls. For example, the uninstaller for the WBIA adapter for JMS is located in a directory named _uninstJMS. To uninstall an adapter that you downloaded using ESD, do the following:

1. From the command line, navigate to the appropriate uninstall directory.
2. To launch the uninstaller for an adapter, execute the following command:

   ```
   java –cp uninstall.jar run
   ```

   Note: The Java Runtime Environment must be installed to run this command.

   Important: To uninstall the entire WebSphere Business Integration Adapters product, navigate to the *ProductDir*/_uninst_WBIA2.3.1 and execute the WBIA Uninstaller named Uninstaller.exe on Windows and uninstaller.bin on Unix.

Chapter 3. Developing business objects for the connector

This chapter covers the elements and structure required for Trading Partner Interchange (TPI) business objects. It specifically discusses the business object requirements of the TPI connector. It contains the following sections:

Data handlers and document formats

The connector supports the exchange of documents in XML, EDI, and binary formats. The TPI connector uses data handlers to convert TPI documents to WebSphere Business Integration Adapter business objects, and to convert WebSphere Business Integration Adapter objects to TPI-supported document formats.

Table 11. WebSphere Business Integration Adapter-delivered data handlers

Document format	WebSphere Business Integration Adapter-delivered data handler
XML	XML
EDI	EDI
Binary	Any

The connector calls the appropriate data handler by passing in the document MIME type to the DataHandler class. Additionally, the connector may pass the BOPrefix value, if it is populated in the incoming business object. When processing an inbound document, the connector gets the MIME type of the document from the trading partner configuration file. When processing an outbound request business object, the connector gets the MIME type from the trading partner configuration file, based on the values of the DocumentType and ReceiverID attributes in the child meta-object. See "Business object structure" on page 40 for more information.

Data handler requirements for business objects

Because the TPI connector calls data handlers to convert from streams to business objects and from business objects to streams, each business object must conform to the specifications of the data handler called to perform the conversion. For information about the specific business object requirements for a particular data handler, see the *Data Handler Guide*.

TPI Server requirements for XML and EDI documents

The TPI Server has specific requirements for the content and formatting of header information in EDI and XML documents. All XML and EDI documents must contain a valid SenderID, RecieverID and a unique CycloneID. If the SenderID or

ReceiverID values are invalid, or if the CycloneID value is not unique in the TPI system, the TPI Server will not process the document. These values must correspond to request business objects so that the data handler places them correctly in the document.

EDI requirements

For EDI documents the placement of these values is mandated by the EDI specification. Consider the following sample EDI header:

```
ISA*00* *01*XXXXXX *L1*2 *L0*0*961106*2106*U*00302*000087875*
```

SenderID: The SenderID is read from two columns, *L1*2 in this example. The first column must contain the two character SenderID qualifier. The second column must contain all remaining characters of the SenderID.

ReceiverID: Like SenderID, the ReceiverID is also read from two columns in the EDI document header, *L0*0 in this example. The first column must contain the two character ReceiverID qualifier. The second column must contain all remaining characters of the ReceiverID.

CycloneID: for EDI documents, the CycloneID corresponds to the EDI Control ID. The CycloneID is represented in a single column, *000087875* in this example. This value corresponds to the EDI Control ID number 87875.

XML requirements

The placement of SenderID, ReceiverID, and CycloneID in XML documents can be customized using the TPI Server Administrator. See the Administrator's Guide included with the TPI Server for more information.

Business object and attribute naming conventions

There are no specific naming conventions for business objects used in request processing. Business objects that are processed in event notification must follow the naming conventions of the data handler that performs the conversion between the incoming document format and the WebSphere Business Integration Adapter business object. See the *Data Handler Guide* for more information.

Business object structure

The TPI connector has two requirements for the business objects it processes:

- Each business object must contain a meta-object that holds the dynamic meta-data required by the connector to format the content and send the document using the TPI Server API.
- Each business object must conform to the business object structure required by the data handler used to convert the object into a data stream.

Child meta-object attributes

Each business object sent from a business process to the connector must contain a single-cardinality meta-object as a cardinality 1 child. The meta-object contains the dynamic meta-data required by the connector to call the appropriate data handler for converting the object, and to call the sendDocument() method in the TPI Server API. Table 12 lists the attributes of the meta-object. All of the listed attributes must be defined in the business object definition.

Table 12. Meta-object attributes for TPI connector

Attribute name	Description	Default value
DocumentExt	Specifies the document's file extension; used during request processing.	
DocumentType	Specifies the format of the document—XML, EDI, or binary.	binary
BOPrefix	Used with the MIME type to create an instance of the XML data handler.	
SenderID	Specifies the unique TPI ID for the trading partner that is sending the document.	Default value must be set by user.
ReceiverID	Specifies the unique TPI ID for the trading partner that is receiving the document.	Default value must be set by user.
UniqueID	Specifies the unique identifier assigned to each document by the TPI Server. This attribute is optional, but may used for processing business object requests.	
OriginalName	Specifies the prefix used to name the document object file that is written to the document out directory.	Must be set by user.
WaitForMDN	Determines whether the connector waits for an MDN from the TPI Server after sending the document.	true
BackupRequired	Determines whether the TPI Server creates a backup copy of the document after sending it to the trading partner.	true

DocumentExt

The DocumentExt attribute enables you to specify file extensions (for example, .xml or .edi) during request processing. It is an optional attribute, and has no default value.

DocumentType

The `DocumentType` attribute is used with the `ReceiverID` attribute to get the document MIME type from the trading partner configuration file. The connector uses the MIME type to invoke the appropriate data handler. The default value is binary.

The DocumentType attribute can also accept the special value CW_RNIF. In this situation, you do not need to specify a value for the ReceiverID attribute; it is not used by the connector.

BOPrefix

The BOPrefix is used with the MIME type by the connector to invoke the appropriate data handler instance. If the data handler implementation handles only one MIME type, the BOPrefix attribute in a child meta-object is optional.

SenderID

The SenderID is the unique partner ID of the document sender. This value is used by the connector to build the DefaultDocument object, which is passed in the sendDocument() call.

ReceiverID

The ReceiverID is the unique ID of the trading partner to whom the document is being sent. This value is used with the DocumentType attribute to get the document MIME type from the trading partner configuration file. It is also used to

build the DefaultDocument object, which is passed in the sendDocument() call. This attribute is optional, but may used for business object processing.

UniqueID
The unique identifier assigned to each document by the TPI Server.

Original Name
The prefix used to name the output file that the connector writes to the DocumentOutDir directory for retrieval by the TPI Server. The ObjectEventId is appended to this name to guarantee uniqueness.

WaitForMDN
This attribute determines whether the connector waits for notification from TPI that the MDN was received for the document. This attribute is optional. If this attribute is populated in the meta-object, it overrides the connector's WaitForMDN property. The default value is true.

BackupRequired
This attribute sets a flag to have the TPI Server back up the document after sending it. This attribute is optional. If this attribute is populated, it overrides the BackupRequired connector property. This value is passed as a parameter in the sendDocument() call. The default value is true.

Data handler requirements for business object structure

Each data handler used by the TPI connector has its own requirements for business object structure. Business objects must conform to the specifications of the data handler called to convert the them. These requirements are documented in the *Data Handler Guide*.

Sample business object with child meta-object

The following is an example of a TPI connector business object definition with the meta-object as a cardinality 1 child. This business object definition was developed for the delimited data handler.

```
[BusinessObjectDefinition]
Name = TPICustBO
Version = 1.0.0.
AppSpecificInfo = cw_mo_cfg=CustBORouteInfo;

[Attribute]
Name = FirstName
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
[end]

[Attribute]
Name = LastName
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
[end]

[Attribute]
Name = Company
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
```

```
[end]

[Attribute]
Name = City
Type = String
IsKey = false
IsRequired = false
AppSpecificInfo =
[end]

[Attribute]
Name = CustBORouteInfo
Type = TPIRouteInfo
ContainedObjectVersion = 1.0.0.
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue =
AppSpecificInfo = type=cw_mo_cfg
[end]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Retrieve
[End]

[End]

[BusinessObjectDefinition]
Name = TPIRouteInfo
Version = 1.0.0.

[Attribute]
Name = SenderId
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
[end]

[Attribute]
Name = ReceiverId
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
[end]

[Attribute]
Name = DocumentType
Type = String
IsKey = true
```

```
IsRequired = true
AppSpecificInfo =

[end]

[Attribute]
Name = BOPrefix
Type = String
IsKey = false
IsRequired = false
AppSpecificInfo =
[end]

[Attribute]
Name = WaitForMDN
Type = String
IsKey = false
IsRequired = false
AppSpecificInfo =
[end]

[Attribute]
Name = BackupRequired
Type = String
IsKey = false
IsRequired = false
AppSpecificInfo =
[end]

[Attribute]
Name = OriginalName
Type = String
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = TPICustomer
[End]

[Attribute]
Name = UniqueId
Type = String
IsKey = false
IsForeignKey = false
IsRequired = false
[end]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
[End]

[Attribute]
Name = OriginalName
Type = String
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = TPICustomer
[End]

[Attribute]
Name = UniqueId
Type = String
```

```
IsKey = false
IsForeignKey = false
IsRequired = false
[end]

[Verb]
Name = Create
[End]

[Verb]
Name = Retrieve
[End]

[End]
```

Mapping considerations (WebSphere ICS integration broker only)

In some TPI implementations, a company may have two or more trading partners receiving documents based on the same business object, Sales_Order, for example, but in different formats. This scenario requires a polymorphic map, which can output business objects in different formats depending on which trading partner is receiving the data.

A polymorphic map is essentially two or more separate submaps, one for each output type, that take the same input object but produce output objects of different types. These submaps are called by a single main map, which tests for a condition, such as an attribute value, to determine which submap to call. Once the submap has generated an output object, the main map returns the object to the connector.

One way of implementing this for TPI is to use the ReceiverID attribute value as the determining condition. Based on the ReceiverID in the input object, the main map calls the appropriate submap. It is the task of the submap to:

- Generate an output object for the appropriate data handler, depending on the document format specified by the trading partner.
- Populate the child meta-object with the appropriate meta-data.

For more information on polymorphic maps, see the *Map Development Guide*.

Business object verb processing

All business objects processed by the TPI connector must have the Create verb set. If the Create verb is not set on a request the connector fails the business object. The connector sets the Create verb on all event business objects if it is not already set by the data handler.

Business object attribute properties

The TPI connector has no specific requirements for business object attribute properties. Business object attribute properties should conform to the requirements of the data handler used to convert the business object. These requirements are documented in the *Data Handler Guide*.

Business object application-specific information

Application-specific information in business object definitions provides the connector and the data handlers with instructions on how to process business objects. Application-specific information for a TPI business object must meet the requirements of both the connector and the data handler used to process the business object.

Connector requirements for application-specific information

The TPI connector uses application-specific information at the business object level to identify the child meta-object. When converting a business object, WebSphere Business Integration Adapter-delivered data handlers do not include as part of the output stream the contents of child meta-objects with the `type = cw_mo_cfg` tag in the application-specific information. The meta-object name must also be included with the `cw_mo_cfg` tag in the application-specific information of the parent object.

In the top-level business object header, the application-specific information specifies the name of the meta-object with the following syntax:

```
cw_mo_cfg=<meta-object attribute name>
```

In the child business object attribute for the meta-object, the application-specific information specifies the type of the meta-object, and follows this syntax:

```
type=cw_mo_cfg
```

Data handler requirements for application-specific information

In addition to the connector's requirements for application-specific information, you must consider the requirements of the specific data handler used to process each business object. See the *Data Handler Guide* for more information on the application-specific information requirements for each data handler.

Appendix A. Standard configuration properties for connectors

This appendix describes the standard configuration properties for WebSphere Business Integration adapter connectors. The information covers connectors running on the following brokers:

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator Broker (WMQI)
- WebSphere Application Server (WAS)

Not every connector makes use of all these standard properties. When you select a broker from Connector Configurator, you will see a list of the standard properties that you need to configure for your adapter running with that broker.

For information about properties specific to the connector, see the relevant adapter user guide.

Note: In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes and follow the conventions for each operating system.

New and deleted properties

These standard properties have been either added or deleted in the 2.3 release of the adapters.

New properties

- RFH2Message Domain
- ListenerConcurrency
- RestartCount
- WsifSynchronousRequestTimeout

Deleted properties

None

Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties
- Connector-specific configuration properties

This section describes the standard configuration properties. For information on configuration properties specific to a connector, see its adapter user guide.

Using Connector Configurator

You configure connector properties from Connector Configurator, which you access from System Manager. For more information on using Connector Configurator, refer to the Connector Configurator appendix.

Note: Connector Configurator and System Manager run only on the Windows system. If you are running the connector on a UNIX system, you must have a Windows machine with these tools installed. To set connector properties for a connector that runs on UNIX, you must start up System Manager on the Windows machine, connect to the UNIX integration broker, and bring up Connector Configurator for the connector.

Setting and updating property values

The default length of a property field is 255 characters.

The connector uses the following order to determine a property's value (where the highest number overrides other values):

1. Default
2. Repository (only if WebSphere InterChange Server is the integration broker)
3. Local configuration file
4. Command line

A connector obtains its configuration values at startup. If you change the value of one or more connector properties during a run-time session, the property's **Update Method** determines how the change takes effect. There are four different update methods for standard connector properties:

- **Dynamic**
  The change takes effect immediately after it is saved in System Manager. If the connector is working in stand-alone mode (independently of System Manager), for example with the WebSphere MQ Integrator Broker, you can only change properties through the configuration file. In this case, a dynamic update is not possible.
- Component restart
  The change takes effect only after the connector is stopped and then restarted in System Manager. You do not need to stop and restart the application-specific component or the integration broker.
- Server restart
  The change takes effect only after you stop and restart the application-specific component and the integration broker.
- Agent restart (ICS only)
  The change takes effect only after you stop and restart the application-specific component.

To determine how a specific property is updated, refer to the **Update Method** column in the Connector Configurator window, or see the Update Method column in the Property Summary table below.

Summary of standard properties

Table 13 on page 49 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker.

You must set the values of some of these properties before running the connector. See the following section for an explanation of each property.

Table 13. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdminInQueue	*Valid JMS queue name*	*CONNECTORNAME* /ADMININQUEUE		Delivery Transport is JMS
AdminOutQueue	*Valid JMS queue name*	*CONNECTORNAME*/ADMINOUTQUEUE		Delivery Transport is JMS
AgentConnections	1-4	1	Component restart	ICS: Delivery Transport is MQ or IDL
AgentTraceLevel	0-5	0	Dynamic	
ApplicationName	*application name*	*The value that is specified for the connector application name*	Component restart	Value required
BrokerType	ICS, WMQI, WAS			
CharacterEncoding	ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437 **Note:** This is a subset of supported values.	ascii7	Component restart	
ConcurrentEventTriggeredFlows	1 to 32,767	No value	Component restart	
ContainerManagedEvents	No value or JMS	JMS		Guaranteed event delivery
ControllerStoreAndForwardMode	true or false	True	Dynamic	ICS only
ControllerTraceLevel	0-5	0	Dynamic	ICS only
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	Component restart	JMS transport only
DeliveryTransport	MQ, IDL, or JMS	JMS	Component restart	For WAS or WMQI: JMS only
DuplicateEventElimination	True/False	False	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	Component restart	
jms.FactoryClassName	CxCommon.Messaging.jms .IBMMQSeriesFactory or CxCommon.Messaging .jms.SonicMQFactory or any Java class name	CxCommon.Messaging. jms.IBMMQSeriesFactory	Server restart	JMS transport only
jms.MessageBrokerName	If FactoryClassName is IBM, use crossworlds.queue. manager. If FactoryClassName is Sonic, use localhost:2506.	crossworlds.queue.manager	Server restart	JMS transport only
jms.NumConcurrentRequests	Positive integer	10	Component restart	JMS transport only
jms.Password	Any valid password		Server restart	JMS transport only

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
jms.UserName	Any valid name		Server restart	JMS transport only
JvmMaxHeapSize	Heap size in megabytes	128m	Component restart	ICS only
JvmMaxNativeStackSize	Size of stack in kilobytes	128k	Component restart	ICS only
JvmMinHeapSize	Heap size in megabytes	1m	Component restart	ICS only
ListenerConcurrency	1- 100	1	Component restart	ICS only: Delivery Transport must be MQ
Locale	en_US, ja_JP, ko_KR, zh_C, zh_T, fr_F, de_D, it_I, es_E, pt_BR **Note:** This is a subset of the supported locales.	en_US	Component restart	
LogAtInterchangeEnd	True or False	False	Component restart	ICS only
MaxEventCapacity	1-2147483647	2147483647	Dynamic	ICS: Repository Directory must be <REMOTE>
MessageFileName	*path/filename*	*Connectorname*.txt or InterchangeSystem.txt	Component restart	
MonitorQueue	Any valid queue name	*CONNECTORNAME*/MONITORQUEUE	Component restart	JMS transport only: DuplicateEvent Elimination must be True
OADAutoRestartAgent	True or False	False	Dynamic	ICS only: Repository Directory must be <REMOTE>
OADMaxNumRetry	*A positive number*	1000	Dynamic	ICS only: Repository Directory must be <REMOTE>
OADRetryTimeInterval	*A positive number in minutes*	10	Dynamic	ICS only: Repository Directory must be <REMOTE>
PollEndTime	HH:MM	HH:MM	Component restart	
PollFrequency	*a positive integer in milliseconds* no (to disable polling) key (to poll only when the letter p is entered in the connector's Command Prompt window)	10000	Dynamic	

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
PollQuantity	1-500	1	Component restart	JMS transport only: DuplicateEvent Elimination must be True
PollStartTime	HH:MM*(HH is 0-23, MM is 0-59)*	HH:MM	Component restart	
RepositoryDirectory	Location of meta-data repository		Component restart	For ICS: set to <REMOTE> For WMQI and WAS: set to *<local directory>*
RequestQueue	*Valid JMS queue name*	*CONNECTORNAME*/REQUESTQUEUE	Component restart	
ResponseQueue	*Valid JMS queue name*	*CONNECTORNAME*/RESPONSEQUEUE	Component restart	
RestartCount	0-100		Dynamic	Connector must be in polling mode
RestartRetryCount	0-99	3	Dynamic	
RestartRetryInterval	*A sensible positive value in minutes*	1	Dynamic	
RHF2MessageDomain	mrm, xml	mrm	Component restart	Only if Delivery Transport is JMS and WireFormat is CwXML
SourceQueue	*Valid WebSphere MQ name*	*CONNECTORNAME*/SOURCEQUEUE	Component restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	Component restart	
SynchronousRequestTimeout	0 - any number (millisecs)	0	Component restart	
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	Component restart	
WireFormat	CwXML, CwBO	CwXML	Component restart	CwXML for WMQI and WAS; CwBO if Repository Directory is <REMOTE> (ICS)
WisfSynchronousRequest Timeout	0 - any number (millisecs)	0	Component restart	WAS only

Standard configuration properties

This section lists and defines each of the standard connector configuration properties.

AdminInQueue

The queue that is used by the integration broker to send administrative messages to the connector.

The default value is `CONNECTORNAME/ADMININQUEUE`.

AdminOutQueue

The queue that is used by the connector to send administrative messages to the integration broker.

The default value is `CONNECTORNAME/ADMINOUTQUEUE`.

AgentConnections

WebSphere ICS only.

The AgentConnections property controls the number of ORB connections opened by `orb.init[]`.

By default, the value of this property is set to 1. There is no need to change this default.

AgentTraceLevel

Level of trace messages for the application-specific component. The default is `0`. The connector delivers all trace messages applicable at the tracing level set or lower.

ApplicationName

Name that uniquely identifies the connector's application. This name is used by the system administrator to monitor the WebSphere business integration system environment. This property must have a value before you can run the connector.

BrokerType

Identifies the integration broker type that you are using. The options are ICS, WMQI or WAS.

CharacterEncoding

Specifies the character code set used to map from a character (such as a letter of the alphabet, a numeric representation, or a punctuation mark) to a numeric value.

Note: Java-based connectors do not use this property. A C++ connector currently uses the value ASCII for this property. If you previously configured the value of this property to `ascii7` or `ascii8`, you must reconfigure the connector to use either `ASCII` or one of the other supported values.

Important: By default only a subset of supported character encodings display in the drop list. To add other supported values to the drop list, you must manually modify the `\Data\Std\stdConnProps.xml` file in the product directory. For more information, see the appendix on Connector Configurator.

The default value is `ascii`.

ConcurrentEventTriggeredFlows

WebSphere ICS only.

Determines how many business objects can be concurrently processed by the connector for event delivery. Set the value of this attribute to the number of business objects you want concurrently mapped and delivered. For example, set the value of this property to 5 to cause five business objects to be concurrently processed. The default value is 1.

Setting this property to a value greater than 1 allows a connector for a source application to map multiple event business objects at the same time and deliver them to multiple collaboration instances simultaneously. This speeds delivery of business objects to the integration broker, particularly if the business objects use complex maps. Increasing the arrival rate of business objects to collaborations can improve overall performance in the system.

To implement concurrent processing for an entire flow (from a source application to a destination application), you must:

- Configure the collaboration to use multiple threads by setting its `Maximum number of concurrent events` property high enough to use multiple threads.
- Ensure that the destination application's application-specific component can process requests concurrently. That is, it must be multi-threaded, or be able to use connector agent parallelism and be configured for multiple processes. Set the `Parallel Process Degree` configuration property to a value greater than 1.

The `ConcurrentEventTriggeredFlows` property has no effect on connector polling, which is single-threaded and performed serially.

ContainerManagedEvents

This property allows a JMS-enabled connector with a JMS event store to provide guaranteed event delivery, in which an event is removed from the source queue and placed on the destination queue as a single JMS transaction.

The default value is `JMS`. It can also be set to no value.

When ContainerManagedEvents is set to `JMS`, you must configure the following properties to enable guaranteed event delivery:

- PollQuantity = `1` to `500`
- SourceQueue = `SOURCEQUEUE`

You must also configure a data handler with the MimeType, DHClass, and DataHandlerConfigMOName (optional) properties. To set those values, use the **Data Handler** tab in Connector Configurator. The fields for the values under the Data Handler tab will be displayed only if you have set `ContainerManagedEvents` to `JMS`.

Note: When ContainerManagedEvents is set to `JMS`, the connector does *not* call its `pollForEvents()` method, thereby disabling that method's functionality.

This property only appears if the `DeliveryTransport` property is set to the value `JMS`.

ControllerStoreAndForwardMode

WebSphere ICS only.

Sets the behavior of the connector controller after it detects that the destination application-specific component is unavailable.

If this property is set to `true` and the destination application-specific component is unavailable when an event reaches ICS, the connector controller blocks the request to the application-specific component. When the application-specific component becomes operational, the controller forwards the request to it.

However, if the destination application's application-specific component becomes unavailable **after** the connector controller forwards a service call request to it, the connector controller fails the request.

If this property is set to `false`, the connector controller begins failing all service call requests as soon as it detects that the destination application-specific component is unavailable.

The default is `true`.

ControllerTraceLevel

WebSphere ICS only.

Level of trace messages for the connector controller. The default is `0`.

DeliveryQueue

The queue that is used by the connector to send business objects to the integration broker.

The default value is `DELIVERYQUEUE`.

DeliveryTransport

Specifies the transport mechanism for the delivery of events. Possible values are `MQ` for WebSphere MQ, `IDL` for CORBA IIOP, or `JMS` for Java Messaging Service.

- If ICS is the broker type, the value of the `DeliveryTransport` property can be MQ, IDL, or JMS, and the default is IDL.
- If WMQI is the broker type, the value may only be JMS.
- If WAS is the broker type, the value may only be JMS.

The connector sends service call requests and administrative messages over CORBA IIOP if the value configured for the `DeliveryTransport` property is `MQ` or `IDL`.

WebSphere MQ and IDL

Use WebSphere MQ rather than IDL for event delivery transport, unless you must have only one product. WebSphere MQ offers the following advantages over IDL:

- Asynchronous communication:
  WebSphere MQ allows the application-specific component to poll and persistently store events even when the server is not available.
- Server side performance:
  WebSphere MQ provides faster performance on the server side. In optimized mode, WebSphere MQ stores only the pointer to an event in the repository

database, while the actual event remains in the WebSphere MQ queue. This saves having to write potentially large events to the repository database.

- Agent side performance:
  WebSphere MQ provides faster performance on the application-specific component side. Using WebSphere MQ, the connector's polling thread picks up an event, places it in the connector's queue, then picks up the next event. This is faster than IDL, which requires the connector's polling thread to pick up an event, go over the network into the server process, store the event persistently in the repository database, then pick up the next event.

JMS

Enables communication between the connector and client connector framework using Java Messaging Service (JMS).

If you select JMS as the delivery transport, additional JMS properties such as `jms.MessageBrokerName`, `jms.FactoryClassName`, `jms.Password`, and `jms.UserName`, appear in Connector Configurator. The first two of these properties are required for this transport.

Important: There may be a memory limitation if you use the JMS transport mechanism for a connector in the following environment:

- AIX 5.0
- WebSphere MQ 5.3.0.1
- When ICS is the integration broker

In this environment, you may experience difficulty starting both the connector controller (on the server side) and the connector (on the client side) due to memory use within the WebSphere MQ client. If your installation uses less than 768M of process heap size, IBM recommends that you set:

- The `LDR_CNTRL` environment variable in the `CWSharedEnv.sh` script.

  This script resides in the `\bin` directory below the product directory. With a text editor, add the following line as the first line in the `CWSharedEnv.sh` script:

  ```
  export LDR_CNTRL=MAXDATA=0x30000000
  ```

  This line restricts heap memory usage to a maximum of 768 MB (3 segments * 256 MB). If the process memory grows more than this limit, page swapping can occur, which can adversely affect the performance of your system.
- The `IPCCBaseAddress` property to a value of `11` or `12`. For more information on this property, see the *System Installation Guide for UNIX*.

DuplicateEventElimination

When you set this property to `true`, a JMS-enabled connector can ensure that duplicate events are not delivered to the delivery queue. To use this feature, the connector must have a unique event identifier set as the business object's **ObjectEventId** attribute in the application-specific code. This is done during connector development.

This property can also be set to `false`.

Note: When `DuplicateEventElimination` is set to `true`, you must also configure the `MonitorQueue` property to enable guaranteed event delivery.

FaultQueue

If the connector experiences an error while processing a message then the connector moves the message to the queue specified in this property, along with a status indicator and a description of the problem.

The default value is `CONNECTORNAME/FAULTQUEUE`.

JvmMaxHeapSize

The maximum heap size for the agent (in megabytes). This property is applicable only if the `RepositoryDirectory` value is <REMOTE>.

The default value is 128m.

JvmMaxNativeStackSize

The maximum native stack size for the agent (in kilobytes). This property is applicable only if the `RepositoryDirectory` value is <REMOTE>.

The default value is 128k.

JvmMinHeapSize

The minimum heap size for the agent (in megabytes). This property is applicable only if the `RepositoryDirectory` value is <REMOTE>.

The default value is 1m.

jms.FactoryClassName

Specifies the class name to instantiate for a JMS provider. You *must* set this connector property when you choose `JMS` as your delivery transport mechanism (DeliveryTransport).

The default is `CxCommon.Messaging.jms.IBMMQSeriesFactory`.

jms.MessageBrokerName

Specifies the broker name to use for the JMS provider. You *must* set this connector property when you choose `JMS` as your delivery transport mechanism (DeliveryTransport).

The default is `crossworlds.queue.manager`.

jms.NumConcurrentRequests

Specifies the maximum number of concurrent service call requests that can be sent to a connector at the same time. Once that maximum is reached, new service calls block and wait for another request to complete before proceeding.

The default value is 10.

jms.Password

Specifies the password for the JMS provider. A value for this property is optional.

There is no default.

jms.UserName

Specifies the user name for the JMS provider. A value for this property is optional.

There is no default.

ListenerConcurrency

This property supports multi-threading in MQ Listener when ICS is the integration broker. It enables batch writing of multiple events to the database, thus improving system performance. The default value is 1.

This property applies only to connectors using MQ transport. The `DeliveryTransport` property must be set to `MQ`.

Locale

Specifies the language code, country or territory, and, optionally, the associated character code set. The value of this property determines such cultural conventions as collation and sort order of data, date and time formats, and the symbols used in monetary specifications.

A locale name has the following format:

ll_TT.codeset

where:

ll	a two-character language code (usually in lower case)
TT	a two-letter country or territory code (usually in upper case)
codeset	the name of the associated character code set; this portion of the name is often optional.

By default, only a subset of supported locales appears in the drop list. To add other supported values to the drop list, you must manually modify the `\Data\Std\stdConnProps.xml` file in the product directory. For more information, see the appendix on Connector Configurator.

The default value is `en_US`. If the connector has not been globalized, the only valid value for this property is `en_US`. To determine whether a specific connector has been globalized, see the connector version list on these websites:

http://www.ibm.com/software/websphere/wbiadapters/infocenter, or
http://www.ibm.com/websphere/integration/wicserver/infocenter

LogAtInterchangeEnd

Specifies whether to log errors to the integration broker's log destination. Logging to the broker's log destination also turns on e-mail notification, which generates e-mail messages for the `MESSAGE_RECIPIENT` specified in the `InterchangeSystem.cfg` file when errors or fatal errors occur.

For example, when a connector loses its connection to its application, if `LogAtInterChangeEnd` is set to `true`, an e-mail message is sent to the specified message recipient. The default is `false`.

MaxEventCapacity

The maximum number of events in the controller buffer. This property is used by flow control and is applicable only if the value of the `RepositoryDirectory` property is <REMOTE>.

The value can be a positive integer between 1 and 2147483647. The default value is 2147483647.

MessageFileName

The name of the connector message file. The standard location for the message file is `\connectors\messages`. Specify the message filename in an absolute path if the message file is not located in the standard location.

If a connector message file does not exist, the connector uses `InterchangeSystem.txt` as the message file. This file is located in the product directory.

Note: To determine whether a specific connector has its own message file, see the individual adapter user guide.

MonitorQueue

The logical queue that the connector uses to monitor duplicate events. It is used only if the `DeliveryTransport` property value is JMS and `DuplicateEventElimination` is set to `TRUE`.

The default value is `CONNECTORNAME/MONITORQUEUE`

OADAutoRestartAgent

Valid only when the integration broker is ICS and the `Repository Directory` is <REMOTE>.

Specifies whether the Object Activation Daemon (OAD) automatically attempts to restart the application-specific component after an abnormal shutdown. This property is required for automatic restart.

The default value is `false`.

OADMaxNumRetry

Valid only when the integration broker is ICS and the `Repository Directory` is <REMOTE>.

Specifies the maximum number of times that the OAD automatically attempts to restart the application-specific component after an abnormal shutdown.

The default value is `1000`.

OADRetryTimeInterval

Valid only when the integration broker is ICS and the `Repository Directory` is <REMOTE>.

Specifies the number of minutes for the interval during which the OAD automatically attempts to restart the application-specific component after an

abnormal shutdown. If the application-specific component does not start within the specified interval, the OAD repeats the attempt as many times as specified in "OADMaxNumRetry" on page 58.

The default is `10`.

PollEndTime

Time to stop polling the event queue. The format is `HH:MM`, where *HH* represents 0-23 hours, and *MM* represents 0-59 seconds.

You must provide a valid value for this property. The default value is `HH:MM`, but must be changed.

PollFrequency

The amount of time between polling actions. Set `PollFrequency` to one of the following values:

- The number of milliseconds between polling actions.
- The word `key`, which causes the connector to poll only when you type the letter `p` in the connector's Command Prompt window. Enter the word in lowercase.
- The word `no`, which causes the connector not to poll. Enter the word in lowercase.

The default is `10000`.

Important: Some connectors have restrictions on the use of this property. To determine whether a specific connector does, see the installing and configuring chapter of its adapter guide.

PollQuantity

Designates the number of items from the application that the connector should poll for. If the adapter has a connector-specific property for setting the poll quantity, the value set in the connector-specific property will override the standard property value.

PollStartTime

The time to start polling the event queue. The format is *HH:MM*, where *HH* represents 0-23 hours, and *MM* represents 0-59 seconds.

You must provide a valid value for this property. The default value is `HH:MM`, but must be changed.

RequestQueue

The queue that is used by the integration broker to send business objects to the connector.

The default value is `REQUESTQUEUE`.

RepositoryDirectory

The location of the repository from which the connector reads the XML schema documents that store the meta-data for business object definitions.

When the integration broker is ICS, this value must be set to <REMOTE> because the connector obtains this information from the InterChange Server repository.

When the integration broker is WMQI or WAS, this value must be set to *<local directory>*.

ResponseQueue

Designates the JMS response queue, which delivers a response message from the connector framework to the integration broker. When the integration broker is IICS, the server sends the request and waits for a response message in the JMS response queue.

RestartCount

Causes the connector to shut down and restart automatically after it has processed a set number of events. You set the number of events in `RestartCount.` The connector must be in polling mode (set `PollFrequency` to "p") for this property to take effect.

Once the set number of events has passed through request processing, the connector is shut down and restarted the next time it polls.

RestartRetryCount

Specifies the number of times the connector attempts to restart itself. When used for a parallel connector, specifies the number of times the master connector application-specific component attempts to restart the slave connector application-specific component.

The default is `3`.

RestartRetryInterval

Specifies the interval in minutes at which the connector attempts to restart itself. When used for a parallel connector, specifies the interval at which the master connector application-specific component attempts to restart the slave connector application-specific component.

The default is `1`.

RHF2MessageDomain

WebSphere MQ Integrator Broker only.

This property allows you to configure the value of the field domain name in the JMS header. When data is sent to WebSphere MQ Integrator Broker over JMS transport, the connector framework writes JMS header information, with a domain name and a fixed value of `mrm`. A configurable domain name enables users to track how WebSphere MQ Integrator Broker processes the message data.

A sample header would look like this:

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

The default value is `mrm`, but it may also be set to `xml`. This property only appears when `DeliveryTransport` is set to `JMS` and `WireFormat` is set to `CwXML`.

SourceQueue

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see "ContainerManagedEvents" on page 53.

The default value is `SOURCEQUEUE`.

SynchronousRequestQueue

Delivers request messages that require a synchronous response from the connector framework to the broker. This queue is necessary only if the connector uses synchronous execution. With synchronous execution, the connector framework sends a message to the `SynchronousRequestQueue` and waits for a response back from the broker on the `SynchronousResponseQueue`. The response message sent to the connector bears a correlation ID that matches the ID of the original message.

SynchronousResponseQueue

Delivers response messages sent in reply to a synchronous request from the broker to the connector framework. This queue is necessary only if the connector uses synchronous execution.

SynchronousRequestTimeout

Specifies the time in minutes that the connector waits for a response to a synchronous request. If the response is not received within the specified time, then the connector moves the original synchronous request message into the fault queue along with an error message.

The default value is 0.

WireFormat

Message format on the transport.

- If the integration broker is WMQI or WAS, the setting is `CwXML.`
- if the integration broker is ICS and the value of `RepositoryDirectory` is <REMOTE>, the setting is`CwBO`.

WisfSynchronousRequest Timeout

WAS integration broker only.

Specifies the time in minutes that the connector waits for a response to a synchronous request. If the response is not received within the specified, time then the connector moves the original synchronous request message into the fault queue along with an error message.

The default value is 0.

Appendix B. Connector Configurator

This appendix describes how to use Connector Configurator to set configuration property values for your adapter.

You use Connector Configurator to:

- Create a connector-specific property template for configuring your connector
- Create a configuration file
- Set properties in a configuration file

Note:

In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes and follow the conventions for each operating system.

The topics covered in this appendix are:

Overview of Connector Configurator

Connector Configurator allows you to configure the connector component of your adapter for use with these integration brokers:

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator Broker (WMQI)
- WebSphere Application Server (WAS)

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see "Running Configurator in stand-alone mode" on page 64).

Each time you install a new adapter, you need to set up a **configuration file** for the connector. This file:

- Sets the standard and application-specific properties for the connector
- Designates which business objects and meta-objects it supports
- Sets the logging and tracing values that the connector will use at run time
- Sets the property values used by messaging and data handlers in the adapter
- Allows you to modify connector properties for an existing connector

You use Connector Configurator to create this configuration file and to modify its settings.

Connector configuration properties include both standard configuration properties (the properties that all connectors have) and connector-specific properties (properties that are needed by the connector for a specific application or technology).

Because **standard properties** are used by all connectors, you do not need to define those properties from scratch; Connector Configurator incorporates them into your configuration file as soon as you create the file. However, you do need to set the value of each standard property in Connector Configurator.

The range of standard properties may not be the same for all brokers and all configurations. Some properties are available only if other properties are given a specific value. The Standard Properties window in Connector Configurator will show the properties available for your particular configuration.

For **connector-specific properties**, however, you need first to define the properties and then set their values. You do this by creating a connector-specific property template for your particular adapter. There may already be a template set up in your system, in which case, you simply use that. If not, follow the steps in "Creating a new template" on page 65 to set up a new one.

Note: Connector Configurator runs only in a Windows environment. If you are running the connector in a UNIX environment, use Connector Configurator in Windows to modify the configuration file and then copy the file to your UNIX environment.

Starting Connector Configurator

You can start and run Connector Configurator in either of two modes:

- Independently, in stand-alone mode (all brokers).
- From System Manager (ICS and WAS only).

Running Configurator in stand-alone mode

You can run Connector Configurator independently and work with connector configuration files, irrespective of your broker. However, if WMQI is your integration broker, you can only use Connector Configurator in stand-alone mode.

To do so:

- From **Start>Programs**, click **IBM WebSphere InterChange Server>IBM WebSphere Business Integration Toolset>Development>Connector Configurator**.
- Select **File>New>Configuration File**.
- When you click the pull-down menu next to **System Connectivity: Integration Broker**, you can select ICS Connectivity, WMQI Connectivity, or WAS Connectivity, depending on your broker.

If you are creating a configuration file for use with ICS or WAS as the broker, you may prefer to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see "Completing a configuration file" on page 69.)

Running Configurator from System Manager

If ICS or WAS is your integration broker, you can run Connector Configurator from System Manager.

To run Connector Configurator:

1. Open the System Manager.
2. In the System Manager window, expand the **Integration Component Libraries i**con and highlight **Connectors**.
3. From the System Manager menu bar, click **Tools>Connector Configurator**. The Connector Configurator window opens and displays a **New Connector** dialog box.
4. When you click the pull-down menu next to **System Connectivity: Integration Broker**, you can select ICS Connectivity or WAS Connectivity, depending on your broker.

To edit an existing configuration file:

1. In the System Manager window, select any of the configuration files listed in the Connector folder and right-click on it. Connector Configurator opens and displays the configuration file with the integration broker type and file name at the top.
2. Click the Standard Properties tab to see which properties are included in this configuration file.

Creating a connector-specific property template

To create a configuration file for your connector, you need a connector-specific property template as well as the system-supplied standard properties.

You can create a brand-new template for the connector-specific properties of your connector, or you can use an existing file as the template.

- To create a new template, see "Creating a new template" on page 65.
- To use an existing file, simply modify an existing template and save it under the new name.

Creating a new template

This section describes how you create properties in the template, define general characteristics and values for those properties, and specify any dependencies between the properties. Then you save the template and use it as the base for creating a new connector configuration file.

To create a template:

1. Click **File>New>Connector-Specific Property Template**.
2. The **Connector-Specific Property Template** dialog box appears, with the following fields:
   - **New Template**, and **Name**

     Enter a unique name that identifies the connector, or type of connector, for which this template will be used. You will see this name again when you open the dialog box for creating a new configuration file from a template.
   - **Old Template**, and **Select the existing template to modify**

     The names of all currently available templates are displayed in the Template Name display.

- To see the connector-specific property definitions in any template, select that template's name in the **Template Name** display. A list of the property definitions contained in that template will appear in the **Template Preview** display. You can use an existing template whose property definitions are similar to those required by your connector as a starting point for your template.

3. Select a template from the **Template Name** display, enter that template name in the **Find Name** field (or highlight your selection in **Template Name**), and click **Next**.

If you do not see any template that displays the connector-specific properties used by your connector, you will need to create one. Connector Configurator provides a template named **None**, containing no property definitions, as a default choice.

Specifying general characteristics

When you click **Next** to select a template, the **Properties - Connector-Specific Property Template** dialog box appears. The dialog box has tabs for General characteristics of the defined properties and for Value restrictions. The General display has the following fields:

- **Edit properties**

  Use the buttons provided (or right-click within the **Edit properties** display) to add a new property to the template, to edit or delete an existing property, or to add a child property to an existing property.

  A child property is an attribute of another property, the parent property. The parent property can obtain simple values, or child properties, or both. These property relationships are hierarchical. When you create a configuration file from these properties, Connector Configurator will identify hierarchical property sets with a plus sign in a box at the left of any parent property.

- **Property type**

  Choose one of these property types: Boolean, String, Integer, or Time.

- Flags

  You can set **Standard Flags** (IsRequired, IsDeprecated, IsOverridden) or **Custom Flag**s (for Boolean operators) to apply to this property.

After you have made selections for the general characteristics of the property, click the **Value** tab.

Specifying values

The **Value** tab enables you to set the maximum length, the maximum multiple values, a default value, or a value range for the property. To do so:

1. Click the **Value** tab. The display panel for Value replaces the display panel for General.
2. Select the name of the property in the **Edit properties** display.
3. In the fields for **Max Length** and **Max Multiple Values**, make any changes. The changes will not be accepted unless you also open the **Property Value** dialog box for the property, described in the next step.
4. Right-click the box in the left-hand corner of the adapter display panel. A **Property Value** dialog box appears. Depending on the property type, the dialog box allows you to enter either a value, or both a value and range. Enter the appropriate value or range, and click **OK**.
5. The **Value** panel refreshes to display any changes you made in **Max Length** and **Max Multiple Values**. It displays a table with three columns:

The **Value** column shows the value that you entered in the **Property Value** dialog box, and any previous values that you created.

The **Default Value** column allows you to designate any of the values as the default.

The **Value Range** shows the range that you entered in the **Property Value** dialog box.

After a value has been created and appears in the grid, it can be edited from within the table display. To make a change in an existing value in the table, select an entire row by clicking on the row number. Then right-click in the **Value** field and click **Edit Value**.

Setting dependencies

When you have made your changes to the **General** and **Value** tabs, click **Next**. The **Dependencies** dialog box appears.

A dependent property is a property that is included in the template and used in the configuration file *only if* the value of another property meets a specific condition. For example, `PollQuantity` appears in the template only if JMS is the transport mechanism and `DuplicateEventElimination` is set to `True`.
To designate a property as dependent and to set the condition upon which it depends, do this:

1. In the **Available Properties** display, select the property that will be made dependent.
2. In the **Select Property** field, use the drop-down menu to select the property that will hold the conditional value.
3. In the **Condition Operator** field, select one of the following:

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

   <=(less than or equal to)
4. In the **Conditional Value** field, enter the value that is required in order for the dependent property to be included in the template.
5. With the dependent property highlighted in the **Available Properties** display, click an arrow to move it to the **Dependent Property** display.
6. Click **Finish**. Connector Configurator stores the information you have entered as an XML document, under `\data\app` in the`\bin` directory where you have installed Connector Configurator.

Creating a new configuration file

When you create a new configuration file, your first step is to select an integration broker. The broker you select determines the properties that will appear in the configuration file.

To select a broker:

- In the Connector Configurator home menu, click **File>New>Connector Configuration**. The **New Connector** dialog box appears.
- In the I**ntegration Broker** field, select ICS, WMQI or WAS connectivity.

Note: To get the choice of WMQI, Connector Configurator must be launched from the Start menu, not from System Manager.

- Complete the remaining fields in the **New Connector** window, as described later in this chapter.

To create a new file for ICS or WAS, you can also do this:

- In the System Manager window, right-click on the **Connectors** folder and select **Create New Connector**. Connector Configurator opens and displays the **New Connector** dialog box.

Creating a configuration file from a connector-specific template

Once a connector-specific template has been created, you can use it to create a configuration file:

1. Click **File>New>Connector Configuration**.
2. The **New Connector** dialog box appears, with the following fields:
   - **Name**

     Enter the name of the connector. Names are case-sensitive. The name you enter must be unique, must end with the word "connector", and must be consistent with the file name for a connector that is installed on the system. For example, enter `PeopleSoftConnector` if the connector file name is `PeopleSoft.jar`.

     Important: Connector Configurator does not check the spelling of the name that you enter. You must ensure that the name is correct.
   - **System Connectivity**

     Click ICS or WMQI or WAS connectivity.
   - **Select Connector-Specific Property Template**

     Type the name of the template that has been designed for your connector. The available templates are shown in the **Template Name** display. When you select a name in the Template Name display, the **Property Template Preview** display shows the connector-specific properties that have been defined in that template.

     Select the template you want to use and click **OK**.
3. A configuration screen appears for the connector that you are configuring. The title bar shows the integration broker and connector names. You can fill in all the field values to complete the definition now, or you can save the file and complete the fields later.
4. To save the file, click **File>Save>to File** or **File>Save>Save to the project**. To save to a project, you must be using ICS or WAS as the broker, and System Manager must be running.
   If you save as a file, the **Save File Connector** dialog box appears. Choose `*.cfg` as the file type, verify in the File Name field that the name is spelled correctly and has the correct case, navigate to the directory where you want to locate the file, and click **Save**. The status display in the message panel of Connector Configurator indicates that the configuration file was successfully created.

   Important: The directory path and name that you establish here must match the connector configuration file path and name that you supply in the startup file for the connector.
5. To complete the connector definition, enter values in the fields for each of the tabs of the Connector Configurator window, as described later in this chapter.

Using an existing file

You may have an existing file available in one or more of the following formats:

- A connector definition file.
  This is a text file that lists properties and applicable default values for a specific connector. Some connectors include such a file in a `\repository` directory in their delivery package (the file typically has the extension `.txt`; for example, CN_XML.txt for the XML connector).
- An ICS repository file.
  Definitions used in a previous ICS implementation of the connector may be available to you in a repository file that was used in the configuration of that connector. Such a file typically has the extension `.in` or `.out`.
- A previous configuration file for the connector.
  Such a file typically has the extension *.cfg.

Although any of these file sources may contain most or all of the connector-specific properties for your connector, the connector configuration file will not be complete until you have opened the file and set properties, as described later in this chapter.

To use an existing file to configure a connector, you must open the file in Connector Configurator, revise the configuration, and then save the file as a configuration file (`*.cfg` file).

Follow these steps to open a *.txt, *.cfg, or *.in file from a directory:

1. In Connector Configurator, click **File>Open>From File**.
2. In the **Open File Connector** dialog box, select one of the following file types to see the available files:
   - Configuration (`*.cfg`)
   - ICS Repository (`*.in`, `*.out`)

     Choose this option if a repository file was used to configure the connector in an ICS environment. A repository file may include multiple connector definitions, all of which will appear when you open the file.
   - All files (*.*)

     Choose this option if a `*.txt` file was delivered in the adapter package for the connector, or if a definition file is available under another extension.
3. In the directory display, navigate to the appropriate connector definition file, select it, and click **Open**.

Follow these steps to open a connector configuration from a System Manager project:

1. Start System Manager. A configuration can be opened from or saved to System Manager only if System Manager has been started.
2. Start Connector Configurator.
3. Click **File>Open>From Project**.

Completing a configuration file

When you open a configuration file or a connector from a project, the Connector Configurator window displays the configuration screen, with the current attributes and values.

The title of the configuration screen displays the integration broker and connector name as specified in the file. Make sure you have the correct broker. If not, change the broker value before you configure the connector. To do so:

1. Under the **Standard Properties** tab, select the value field for the `BrokerType` property. In the drop-down menu, select the value `ICS`, `WMQI`, or `WAS`.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in "Specifying supported business object definitions" on page 72..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

   If you are saving to file, select `*.cfg` as the extension, select the correct location for the file and click **Save**.

   If multiple connector configurations are open, click **Save All to File** to save all of the configurations to file, or click **Save All to Project** to save all connector configurations to a System Manager project.

   Before it saves the file, Connector Configurator checks that values have been set for all required standard properties. If a required standard property is missing a value, Connector Configurator displays a message that the validation failed. You must supply a value for the property in order to save the configuration file.

Setting the configuration file properties

When you create and name a new connector configuration file, or when you open an existing connector configuration file, Connector Configurator displays a configuration screen with tabs for the categories of required configuration values.

Connector Configurator requires values for properties in these categories for connectors running on all brokers:

- Standard Properties
- Connector-specific Properties
- Supported Business Objects
- Trace/Log File values
- Data handlers (applicable for connectors that use JMS messaging with guaranteed event delivery)

Note: For connectors that use JMS messaging, an additional category may display, for configuration of data handlers that convert the data to business objects.

For connectors running on **ICS**, values for these properties are also required:

- Associated Maps
- Resources
- Messaging (where applicable)

Important: Connector Configurator accepts property values in either English or non-English character sets. However, the names of both standard and connector-specific properties, and the names of supported business objects, must use the English character set only.

Standard properties differ from connector-specific properties as follows:

- Standard properties of a connector are shared by both the application-specific component of a connector and its broker component. All connectors have the same set of standard properties. These properties are described in Appendix A of each adapter guide. You can change some but not all of these values.
- Application-specific properties apply only to the application-specific component of a connector, that is, the component that interacts directly with the application. Each connector has application-specific properties that are unique to its application. Some of these properties provide default values and some do not; you can modify some of the default values. The installation and configuration chapters of each adapter guide describe the application-specific properties and the recommended values.

The fields for **Standard Properties** and **Connector-Specific Properties** are color-coded to show which are configurable:

- A field with a grey background indicates a standard property. You can change the value but cannot change the name or remove the property.
- A field with a white background indicates an application-specific property. These properties vary according to the specific needs of the application or connector. You can change the value and delete these properties.
- Value fields are configurable.
- The **Update Method** field is informational and not configurable. This field specifies the action required to activate a property whose value has changed.

Setting standard connector properties

To change the value of a standard property:

1. Click in the field whose value you want to set.
2. Either enter a value, or select one from the drop-down menu if it appears.
3. After entering all the values for the standard properties, you can do one of the following:
   - To discard the changes, preserve the original values, and exit Connector Configurator, click **File>Exit** (or close the window), and click **No** when prompted to save changes.
   - To enter values for other categories in Connector Configurator, select the tab for the category. The values you enter for **Standard Properties** (or any other category) are retained when you move to the next category. When you close the window, you are prompted to either save or discard the values that you entered in all the categories as a whole.
   - To save the revised values, click **File>Exit** (or close the window) and click **Yes** when prompted to save changes. Alternatively, click **Save>To File** from either the File menu or the toolbar.

Setting application-specific configuration properties

For application-specific configuration properties, you can add or change property names, configure values, delete a property, and encrypt a property. The default property length is 255 characters.

1. Right-click in the top left portion of the grid. A pop-up menu bar will appear. Click **Add** to add a property or **Add Child** to add a child property to a property.
2. Enter a value for the property or child property.
3. To encrypt a property, select the **Encrypt** box.

4. Choose to save or discard changes, as described for "Setting standard connector properties" on page 71.

The Update Method displayed for each property indicates whether a component or agent restart is necessary to activate changed values.

Important: Changing a preset application-specific connector property name may cause a connector to fail. Certain property names may be needed by the connector to connect to an application or to run properly.

Encryption for connector properties

Application-specific properties can be encrypted by selecting the **Encrypt** check box in the **Edit Property** window. To decrypt a value, click to clear the **Encrypt** check box, enter the correct value in the **Verification** dialog box, and click **OK**. If the entered value is correct, the value is decrypted and displays.

The adapter user guide for each connector contains a list and description of each property and its default value.

If a property has multiple values, the **Encrypt** check box will appear for the first value of the property. When you select **Encrypt**, all values of the property will be encrypted. To decrypt multiple values of a property, click to clear the **Encrypt** check box for the first value of the property, and then enter the new value in the **Verification** dialog box. If the input value is a match, all multiple values will decrypt.

Update method

Connector properties are almost all static and the **Update Method** is `COmponent restart`. For changes to take effect, you must restart the connector after saving the revised connector configuration file.

Specifying supported business object definitions

Use the **Supported Business Objects** tab in Connector Configurator to specify the business objects that the connector will use. You must specify both generic business objects and application-specific business objects, and you must specify associations for the maps between the business objects.

For you to specify a supported business object, the business objects and their maps must exist in the system.

- Business object definitions and map definitions should be saved into System Manager projects if ICS or WAS is your integration broker.
- Business object definitions and MQ message set files should exist if WMQI is your integration broker.

Note: Some connectors require that certain business objects be specified as supported in order to perform event notification or additional configuration (using meta-objects) with their applications. For more information, see the *Connector Development Guide for C++* or the *Connector Development Guide for Java*.

If ICS is your broker

To specify that a business object definition is supported by the connector, or to change the support settings for an existing business object definition, click the **Supported Business Objects** tab and use the following fields.

Business object name: To designate that a business object definition is supported by the connector, with System Manager running:

1. Click an empty field in the **Business Object Name** list. A drop-down list displays, showing all the business object definitions that exist in the System Manager project.
2. Click on a business object to add it.
3. Set the **Agent Support** (described below) for the business object.
4. In the File menu of the Connector Configurator window, click **Save to Project**. The revised connector definition, including designated support for the added business object definition, is saved to the project in System Manager.

To delete a business object from the supported list:

1. To select a business object field, click the number to the left of the business object.
2. From the **Edit** menu of the Connector Configurator window, click **Delete Row**. The business object is removed from the list display.
3. From the **File** menu, click **Save to Project**.

Deleting a business object from the supported list changes the connector definition and makes the deleted business object unavailable for use in this implementation of this connector. It does not affect the connector code, nor does it remove the business object definition itself from System Manager.

Agent support: If a business object has Agent Support, the system will attempt to use that business object for delivering data to an application via the connector agent.

Typically, application-specific business objects for a connector are supported by that connector's agent, but generic business objects are not.

To indicate that the business object is supported by the connector agent, check the **Agent Support** box. The Connector Configurator window does not validate your Agent Support selections.

Maximum transaction level: The maximum transaction level for a connector is the highest transaction level that the connector supports.

For most connectors, `Best Effort` is the only possible choice, because most application APIs do not support the `Stringent` level.

You must restart the server for changes in transaction level to take effect.

If WMQI is your broker

The MQ message set files (`*.set` files) contain message set IDs that Connector Configurator requires for designating the connector's supported business objects. See *Implementing Adapters with WebSphere MQ Integrator Broker* for information about creating the MQ message set files.

Each time that you add business object definitions to the system, you must use Connector Configurator to designate those business objects as supported by the connector.

Important: If the connector requires meta-objects, you must create message set files for each of them and load them into Connector Configurator, in the same manner as for business objects.

To specify supported business objects:

1. Select the **Supported Business Objects** tab and click **Load**. The **Open Message Set ID File(s)** dialog box displays.
2. Navigate to the directory where you have placed the message set file for the connector and select the appropriate message set file (`*.set`) or files.
3. Click **Open**. The **Business Object Name** field displays the business object names contained in the `*.set` file. The numeric message set ID for each business object is listed in its corresponding Message Set ID field.
   Do not change the message set IDs. These names and numeric IDs are saved when you save the configuration file.
4. When you add business objects to the configuration, you must load their message set files. If you attempt to load a message set that contains a business object name that already exists in the configuration, or if you attempt to load a message set file that contains a duplicate business object name, Connector Configurator detects the duplicate and displays the **Load Results** dialog box. The dialog box shows the business object name or names for which there are duplicates. For each duplicate name shown, click in the **Message Set ID** field, and select the Message Set ID that you wish to use.

If WAS is your broker

When WebSphere Application Server is selected as your broker type, Connector Configurator does not require message set IDs. The **Supported Business Objects** tab shows a **Business Object Name** column only for supported business objects.

If you are working in stand-alone mode (not connected to System Manager), you must enter the business object name manually.

If you have System Manager running, you can select the empty box under the Business Object Name column in the Supported Business Objects tab. A combo box appears with a list of the business objects available from the Integration Component Library project to which the connector belongs. Select the business object you want from this list.

Associated maps (ICS only)

Each connector supports a list of business object definitions and their associated maps that are currently active in WebSphere InterChange Server. This list appears when you select the **Associated Maps** tab.

The list of business objects contains the application-specific business object which the agent supports and the corresponding generic object that the controller sends to the subscribing collaboration. The association of a map determines which map will be used to transform the application-specific business object to the generic business object or the generic business object to the application-specific business object.

If you are using maps that are uniquely defined for specific source and destination business objects, the maps will already be associated with their appropriate business objects when you open the display, and you will not need (or be able) to change them.

If more than one map is available for use by a supported business object, you will need to explicitly bind the business object with the map that it should use.

The **Associated Maps** tab displays the following fields:

- **Business Object Name**

  These are the business objects supported by this connector, as designated in the **Supported Business Objects** tab. If you designate additional business objects under the Supported Business Objects tab, they will be reflected in this list after you save the changes by choosing **Save to Project** from the **File** menu of the Connector Configurator window.
- **Associated Maps**

  The display shows all the maps that have been installed to the system for use with the supported business objects of the connector. The source business object for each map is shown to the left of the map name, in the **Business Object Name** display.
- **Explicit**

  In some cases, you may need to explicitly bind an associated map.

  Explicit binding is required only when more than one map exists for a particular supported business object. When ICS boots, it tries to automatically bind a map to each supported business object for each connector. If more than one map takes as its input the same business object, the server attempts to locate and bind one map that is the superset of the others.

  If there is no map that is the superset of the others, the server will not be able to bind the business object to a single map, and you will need to set the binding explicitly.

  To explicitly bind a map:

  1. In the **Explicit** column, place a check in the check box for the map you want to bind.
  2. Select the map that you intend to associate with the business object.
  3. In the **File** menu of the Connector Configurator window, click **Save to Project**.
  4. Deploy the project to ICS.
  5. Reboot the server for the changes to take effect.

Resources (ICS)

The **Resource** tab allows you to set a value that determines whether and to what extent the connector agent will handle multiple processes concurrently, using connector agent parallelism.
Not all connectors support this feature. If you are running a connector agent that was designed in Java to be multi-threaded, you are advised not to use this feature, since it is usually more efficient to use multiple threads than multiple processes.

Configuring messaging (ICS)

The messaging properties are available only if you have set MQ as the value of the `DeliveryTransport` standard property and ICS as the broker type. These properties affect how your connector will use queues.

Setting trace/log file values

When you open a connector configuration file or a connector definition file, Connector Configurator uses the logging and tracing values of that file as default values. You can change those values in Connector Configurator.

To change the logging and tracing values:

1. Click the **Trace/Log Files** tab.

2. For either logging or tracing, you can choose to write messages to one or both of the following:
   - To console (STDOUT):
     Writes logging or tracing messages to the STDOUT display.

     Note: You can only use the STDOUT option from the **Trace/Log Files** tab for connectors running on the Windows platform.
   - To File:
     Writes logging or tracing messages to a file that you specify. To specify the file, click the directory button (ellipsis), navigate to the preferred location, provide a file name, and click **Save**. Logging or tracing message are written to the file and location that you specify.

     Note: Both logging and tracing files are simple text files. You can use the file extension that you prefer when you set their file names. For tracing files, however, it is advisable to use the extension `.trace` rather than `.trc`, to avoid confusion with other files that might reside on the system. For logging files, `.log` and `.txt` are typical file extensions.

Data handlers

The data handlers section is available for configuration only if you have designated a value of JMS for DeliveryTransport and a value of JMS for ContainerManagedEvents. Not all adapters make use of data handlers.

See the descriptions under `ContainerManagedEvents` in Appendix A, Standard Properties, for values to use for these properties. For additional details, see the *Connector Development Guide for C++* or the *Connector Development Guide for Java*.

Saving your configuration file

When you have finished configuring your connector, you save the connector configuration file. Connector Configurator will save it in the broker mode that you selected during configuration. The title bar of Connector Configurator always displays the broker mode (ICS, WMQI or WAS) that it is currently using.

The file is saved as an XML document. You can save the XML document in three ways:

For ICS:

- From System Manager, as a file with a `*.con` extension in a an ICS User Project, or
- In a directory that you specify.
- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder, if you are using the file as a local configuration file.

For WMQI:

- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder.

For WAS:

- From System Manager, as a file with a `*.con` extension in a WAS User Project, or
- In a directory that you specify.
- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder.

After you have created the configuration file and set its properties, you need to deploy it to the correct location for your connector.

- If you are using ICS as your integration broker, save the configuration in a System Manager project, and use System Manager to load the file into ICS.
- If you are using WMQI as your integration broker, copy the configuration file to the correct location, which must match exactly the configuration file location specified in the startup file for your connector.
- If you are using WAS as your integration broker, save the file in a WAS user project. Use **File>Export** to create `.wsdl` files that you can then import into WSAD-IE.
  You can also export the configuration file as a `.jar` file to a specified directory.

For details about using projects in System Manager, and for further information about deployment, see the following implementation guides:

- For ICS: *Implementation Guide for WebSphere InterChange Server*
- For WMQI: *Implementing Adapters with WebSphere MQ Integrator Broker*
- For WAS: *Implementing Adapters with WebSphere Application Server*

Changing a configuration file

You can change the integration broker setting for an existing configuration file. This enables you to use the file as a template for creating a new configuration file, which can be used with a different broker.

Note: You will need to change other configuration properties as well as the broker mode property if you switch integration brokers.

To change your broker selection within an existing configuration file (optional):

- Open the existing configuration file in Connector Configurator.
- Select the **Standard Properties** tab.
- In the **BrokerType** field of the Standard Properties tab, select the value that is appropriate for your broker.
  When you change the current value, the available tabs and field selections on the properties screen will immediately change, to show only those tabs and fields that pertain to the new broker you have selected.

Completing the configuration

After you have created a configuration file for a connector and modified it, make sure that the connector can locate the configuration file when the connector starts up.

To do so, open the startup file used for the connector, and verify that the location and file name used for the connector configuration file match exactly the name you have given the file and the directory or path where you have placed it.

Using Connector Configurator in a globalized environment

Connector Configurator is globalized and can handle character conversion between the configuration file and the integration broker. Connector Configurator uses native encoding. When it writes to the configuration file, it uses UTF-8 encoding.

Connector Configurator supports non-English characters in:

- All value fields
- Log file and trace file path (specified in the **Trace/Log files** tab)

The drop list for the `CharacterEncoding` and `Locale` standard configuration properties displays only a subset of supported values. To add other values to the drop list, you must manually modify the `\Data\Std\stdConnProps.xml` file in the product directory.

For example, to add the locale `en_GB` to the list of values for the `Locale` property, open the `stdConnProps.xml` file and add the line in boldface type below:

```
<Property name="Locale"
isRequired="true"
updateMethod="component restart">
    <ValidType>String</ValidType>
    <ValidValues>
        <Value>ja_JP</Value>
        <Value>ko_KR</Value>
        <Value>zh_CN</Value>
        <Value>zh_TW</Value>
        <Value>fr_FR</Value>
        <Value>de_DE</Value>
        <Value>it_IT</Value>
        <Value>es_ES</Value>
        <Value>pt_BR</Value>
        <Value>en_US</Value>
        <Value>en_GB</Value>
        <DefaultValue>en_US</DefaultValue>
    </ValidValues>
</Property>
```

Appendix C. Connector feature list

This appendix lists the features supported by the connector. For descriptions of these features, see "Appendix A: Connector Feature Checklist" in the *Connector Development Guide for Java*.

Business object request handling features

Table 14 details the business object request handling features supported by the connector.

Table 14. Business object request handling features

Category	Feature	Support	Notes
Create	Create verb	Full	
Delete	Delete verb	No	
	Logical delete	No	
Exist	Exist verb	No	
Misc	Attribute names	Full	
	Business object names	Partial	The name of the business object is determined based on the rules for the particular data handler being used.
Retrieve	Ignore missing child object	No	
RetrieveByContent	Ignore missing child object	No	
	Multiple results	No	
	RetrieveByContent verb	No	
Update	After-image support	No	The connector just sends a document created based on the data in the business object.
	Delta support	No	
	KeepRelations	No	
Verbs	Retrieve verb	No	
	Subverb support	No	
	Verb stability	Full	The connector does no different processing based on the verb, only sends a document out to a trading partner.

Event notification features

Table 15 details the event notification features supported by the connector.

Table 15. Event notification features

Category	Feature	Support	Notes
Connector Properties	Event distribution	No	Event notification is done via callback, and the TPI Server is running in the connector process space.
	PollQuantity	Full	
Event Table	Event status values	Partial	Event status is maintained via the location of the event information and the extension on the event file.

Table 15. Event notification features (continued)

Category	Feature	Support	Notes
	Object key	N/A	
	Object name	N/A	
	Priority	N/A	
Misc.	Archiving	Partial	The connector does not have an event table per se. It stores information about incoming documents and their status on the file system.
	CDK method gotApplEvent	Full	
	Delta event notification	No	
	Event sequence	Partial	Maintained based on how they are received from TPI Server, not based on priority.
	Future event processing	No	
	In-Progress event recovery	Full	
	Physical delete event	No	
	RetrieveAll	No	
	Smart filtering	N/A	Because we don't have an event table, we don't have any way of filtering out these 'duplicates'. TPI Server will partially take care of this functionality.
	Verb stability	Partial	

General features

Table 16 details the general features supported by the connector.

Table 16. General features

Category	Feature	Support	Notes
Business Object Attributes	Foreign key	No	
	Foreign Key attribute property	N/A	
	Key	No	
	Max Length	Partial	May be used by the data handler. This depends on configuration and what type of data is being processed.
	Meta-data-driven design	Full	
	Required	No	
Connection Lost	Connection lost on poll	Full	
	Connection lost on request processing	Full	
	Connection lost while idle	No	
Connector Properties	ApplicationPassword	No	
	ApplicationUserName	No	
	UseDefaults	Full	
Message Tracing	General messaging	Full	
	generateMsg()	Full	
	Trace level 0	Full	

Table 16. General features (continued)

Category	Feature	Support	Notes
	Trace level 1	Full	
	Trace level 2	Full	
	Trace level 3	Full	
	Trace level 4	Full	
	Trace level 5	Full	
Misc.	CDK method LogMsg	Full	
	Java Package Names	Full	
	Logging messages	Full	
	NT service compliance	Full	
	Transaction support	Full	There is no rollback. The document is never sent if there is a problem.
Special Value	CxBlank processing	Partial	Depends on the data handler used.
	CxIgnore processing	Partial	Depends on the data handler used.

Appendix D. Trading Partner Interchange adapter sample scenario

This sample scenario demonstrates how the TPI adapter can be used to send and receive business objects. The sample can also be used to understand the adapter and its behavior. Samples are only provided for understanding how the adapter works. There may be various ways of implementing the solution using the TPI adapter, but the sample only addresses one of these methods. The sample may not reflect any real life scenario.

This sample walks thru both polling and request processing with the TPI adapter. The Electronic Data Interchange (EDI) document, shipped with the samples, is polled by the adapter. The adapter then does a request process for this document, converts it to a business object, then delivers it to the Cyclone server. This BO will be handled by the Visual Test Connector (VTC) as a request process. This BO will be converted by the adapter to an EDI document. The adapter will transport this document using the Cyclone's Interchange server. Also the adapter will populate the Cyclone's unique Id in the BO returned in the request processing call. The results will be visible in the VTC.

Pre-installation notes and assumptions

1. You have installed and are experienced with IBM WebSphere InterChange Server.
2. You have installed the IBM WebSphere Business Integration Adapter for Trading Partner Interchange.
3. You are familiar with the WebSphere Business Integration System Manager.
4. You are familiar with the Visual Test Connector.
5. IBM WebSphere Integration Server is installed and configured on your network.
6. All paths in this document are written for Solaris. For using samples on Windows NT/2000 replace "/" with "\" in the paths.
7. You have read the documentation for Trading Partner Interchange Adapter version 3.2.x
8. You are familiar with the functionality of Cyclone's Interchange Server and its administration.

Note: Whenever `${PRODUCT_DIR}` is mentioned in this document, it refers to the folder containing your WebSphere Business Integration installation. All environment variables and file separators are specified in the UNIX format. Please make the appropriate changes if running on Windows NT/2000 platforms. (ex. `${PRODUCT_DIR}/connectors` would be `%PRODUCT_DIR%\connectors`)

Initial setup prior to running the sample business object

1. **Install the IBM WebSphere Interchange Server then start the server.**
2. **Load the TPI sample repository:**

   Load the `TPISampleRepos.txt` file into `$CROSSWORLDS/bin/repos_copy -i $CROSSWORLDS/TPI/samples/TPISampleRepos.txt -s <ServerName>`
3. **Load the classes needed:**

Copy $CROSSWORLDS/connectors/TPI/samples/TPIExample.class to $CROSSWORLDS/collaborations/classes/UserCollaborations/TPIExample.class

4. **Restart the IBM WebSphere Interchange Server.**
5. **Open System Manager and connect to WebSphere Interchange Server.**
6. **Set TPI Adapter properties:**
   - Make sure the`MetaEventDir` directory exists. Also make sure that the user running the TPI Adapter has read and write permissions in this directory.
   - Make sure the`ArchiveProcessedDocDir` directory exists. Also make sure that the user running the TPI Adapter has read and write permissions in this directory.
   - Make sure the`DocumentOutDir` directory exists. Also make sure that the user running the TPI Adapter has read and write permissions in this directory.
   - Set the connector property `TradingPartnerConfigurationFile` to`$CROSSWORLDS/connectors/TPI/samples/tpcfg.in`

   If any of the listed directories or paths does not exist, create it and set the correct read and write permissions for the user.
7. **Install the the first Cyclone Interchange Server:**
   - Install the Cyclone Interchange Server on the same machine where you installed the IBM WebSphere Interchange Server.
   - Make sure you have configured this server to run in debug mode.
   - On this Cyclone Interchange Server, create a company with a Profile ID of "`cwld`".
8. **Install the the second Cyclone Interchange Server:**

   Install a second instance of a Cyclone Interchange Server on another machine. On this server create a company with a Profile ID of "`cwldtp3`"
9. **Configure Cyclone Interchange Servers:**

   Using Cyclone's Administrator, set up the company `cwldtp3` to be a trading partner of the company `cwld`. Likewise, setup `cwld` to be a trading partner of the company `cwldtp3`. Make sure that you can exchange binary documents between the two companies.
10. **Stop the Cyclone Interchange Server located on the machine where IBM WebSphere InterChange Server is installed.**
11. **Configure the meta object for EDI datahandler:**

    Open the meta object `MO_DataHandler_DefaultX12Config_846`. Double click on the `NameHandlerFile` attribute of the meta object. Set the default value of this attribute to `$CROSSWORLDS/connectors/TPI/samples/dbfile.txt`.
12. **Start the TPI Adapter:**

    Make sure that adapter comes up correctly.
13. **Start the Visual Test Connector**

    Create a profile for the `PortConnector`. Bind the Visual Test Connector to the `PortConnector`then start it.

Running a polling scenario

1. **Start**

   Copy `Sample_X12_846.edi` in the Cyclone's `ediout` directory of the trading partner `cwldtp3`. Cyclone will pick this file and transfer it to `cwld`
2. **Sending test data:**

- The Cyclone server running the company cwld, will get the document and it will notify the TPI Adapter.
- The TPI Adapter will read the document and convert it into a X12_846 BO and deliver it to the IBM WebSphere Interchange Server.
- The IBM WebSphere Interchange Server will send the X12_846 BO to the VTC as part of a subscription delivery.

  Note: This may take a while.

3. **Accepting transfer of the BO:**

   In the Visual Test Connector, click on **Request->Accept Request**. You will see a BO in the window on the right hand side of the VTC. Open the BO and you will see the attributes that were transferred

4. **Save BO:**

   In the VTC, select the receivedBO. Now click on **Edit->Save BO**. Save this BO into a file. This file will be used at the time of request processing

Running a request processing scenario

1. **Start**
   - In the VTC click on the business object X12_846. Now here select the file which you saved at the time of poll.
   - Click on **Edit->Load BO**.
   - Select the file which you saved at the time of polling.
2. **Examine the file:**

   Now double click on the BO which you loaded from the file. You will see the attributes.
3. **Create a child BO:**

   Create an instance of the child business object attribute CycloneRouteInfo
   -
4. **Create a child BO:**
   - Create an instance of the child business object attribute CycloneRouteInfo
   - Go to the attribute SenderId of CycloneRouteInfo. Set it to cwld
   - Go to the attribute ReceiverId of CycloneRouteInfo. Set it to cwldtp3.
   - Go to the attribute DocumentType of CycloneRouteInfo. Set it to edi.
5. **Set attributes for child BO:**

   Expand the attribute header. Now you will see a child BO ISA.
   - Expand the ISA BO.
   - Go the Interchange_Sender_ID attribute of ISA. Set it to ld.
   - Go to the Interchange_Receiver_Id attribute of ISA. Set it to ldtp3.
   - Go to the Interchange_Control_Number attribute of ISA. Increment the value. This is the EDI control number. It uniquely identifies an EDI transaction.
6. **Click OK.**
7. **Send test data**

   Click on **Request->Send**. The BO will be delivered to ICS. ICS will deliver it to the TPI Adapter as part of a subscription delivery. The TPI Adapter will convert the BO to an EDI document. The TPI Adapter will pass this document to Cyclone's Interchange Server, which in turn will deliver it to the company cwldtp3.
8. **Verify theBO was received**

Look in Cyclone's `ediin` directory of the company `cwldtp3`. You will see a document from `cwld`.

Summary:

If you've performed all the above steps successfully, you should have a working sample scenario that uses the TPI Adapter to exchange business objects between IBM WebSphere ICS andthe Cyclone Interchange Server.

Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

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

Trademarks and service marks

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

IBM
the IBM logo
AIX
CrossWorlds
DB2
DB2 Universal Database
Domino
Lotus
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

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

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

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

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

IBM WebSphere InterChange Server V4.2.1, IBM WebSphere Business Integration Toolset V4.2.1, IBM WebSphere Business Integration Adapters V2.3.1, IBM WebSphere Business Integration Collaborations V4.2.