IBM WebSphere Business Integration Adapters

Adapter for Trading Partner Interchange User Guide

Version 34.x

IBM WebSphere Business Integration Adapters

Adapter for Trading Partner Interchange User Guide

Version 34.x

Note!
Before using this information and the product it supports, read the information in "Notices" on page 77.

20February2004

This edition of this document applies to connector version 3.4.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

The IBM(R) WebSphere(R) Business Integration Adapter portfolio supplies integration connectivity for leading e-business technologies, enterprise applications, legacy, and mainframe systems. The product set includes tools and templates for customizing, creating, and managing components for business process integration.

This document describes the installation, configuration, business object development, and troubleshooting 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 complete set of documentation available with this product describes the features and components common to all WebSphere Business Integration Adapters installations, and includes reference material on specific components.

You can install related documentation from the following sites:

For general adapter information; for using adapters with WebSphere message brokers (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, WebSphere Business Integration Message Broker); and for using adapters with WebSphere Application Server:

http://www.ibm.com/websphere/integration/wbiadapters/infocenter

For using adapters with InterChange Server:

http://www.ibm.com/websphere/integration/wicserver/infocenter
http://www.ibm.com/websphere/integration/wbicollaborations/infocenter

For more information about message brokers (WebSphere MQ Integrator Broker, WebSphere MQ Integrator, and WebSphere Business Integration Message Broker):

http://www.ibm.com/software/integration/mqfamily/library/manualsa/

For more information about WebSphere Application Server:

http://www.ibm.com/software/webservers/appserv/library.html

These sites contain simple directions for downloading, installing, and viewing the documentation.

Typographic 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 3.4x

February 2004

A datahandler specific to the RosettaNet Implementation Framework (RNIF) over TPI is provided.

Sample implemetation scenarios are provided.

December 2003

The Adapter for Trading Partner Interchange has been updated with general maintenance fixes.

Beginning with the 3.4 version, the adapter for Trading Partner Interchange is no longer supported on Microsoft Windows NT and IBM AIX.

Adapter installation information has been moved from this guide. See Chapter 2 “Installing the TPI Server” on page 8 for the new location of that information.

Prior versions

Below is a history of changes and new features in prior versions.

Version 3.3.x

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

Version 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).

Version 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.

Version 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 27

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 12

Version 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.

Version 2.0.x

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

Version 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.

Version 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.

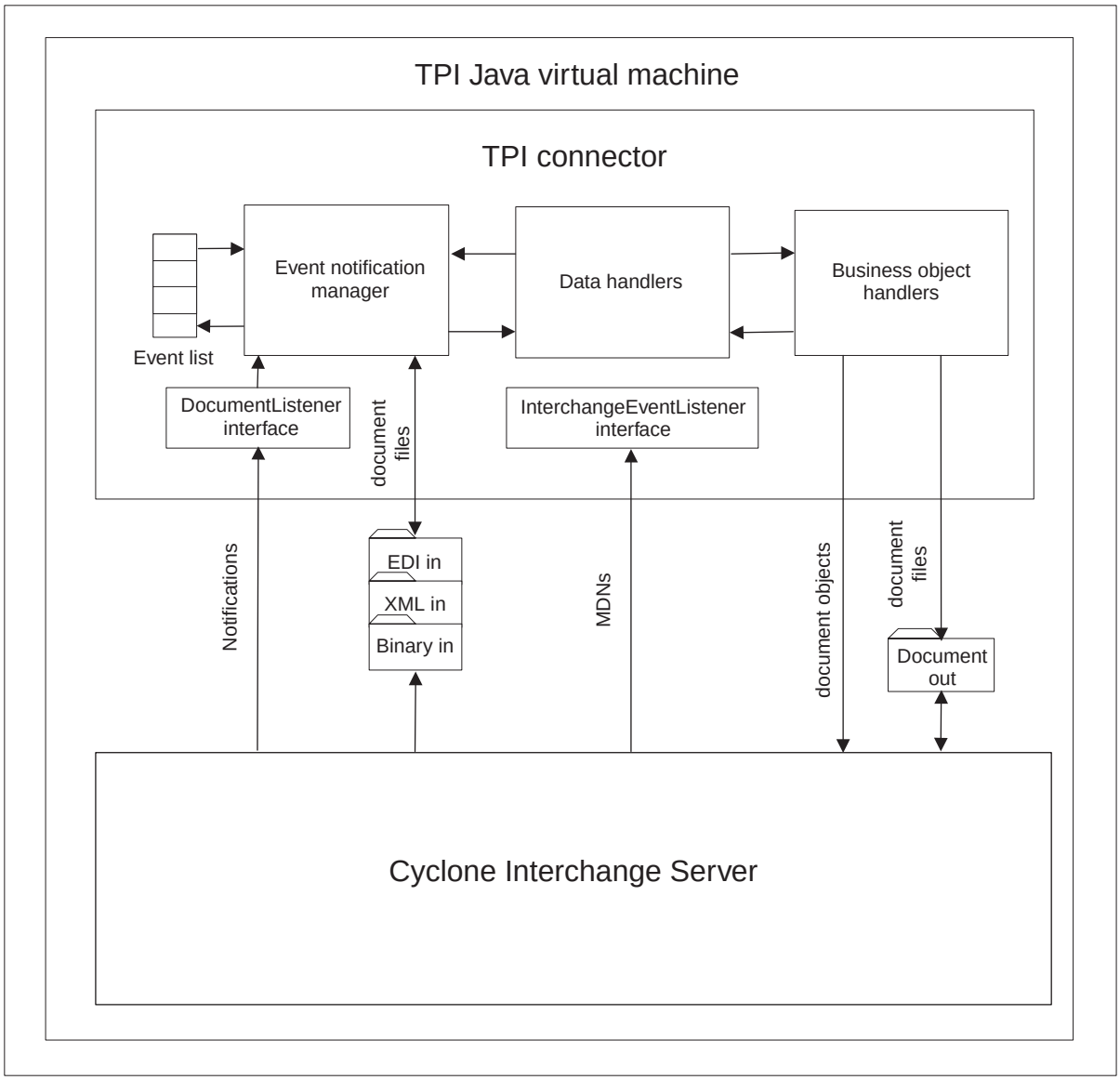


Figure 1. TPI connector architecture

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 12 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 27.

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. Version 3.4 of the adapter for Trading Partner Interchange is supported on the following versions of the adapter framework and with the following integration brokers:

Adapter framework: WebSphere Business Integration Adapter Framework versions 2.1, 2.2, 2.3.x, and 2.4.

Integration brokers:

- WebSphere InterChange Server, versions 4.1.1, 4.2.0, 4.2.1, 4.2.2
- WebSphere MQ Integrator, version 2.1.0
- WebSphere MQ Integrator Broker, version 2.1.0
- WebSphere Business Integration Message Broker, version 5.0

See the Release Notes for any exceptions.

Note: For instructions on installing the integration broker and its prerequisites, see the following documentation. For WebSphere InterChange Server (ICS), see the System Installation Guide for UNIX or for Windows.

For message brokers (WebSphere MQ Integrator Broker, WebSphere MQ Integrator, and WebSphere Business Integration Message Broker), see *Implementing Adapters with WebSphere Message Brokers*, and the installation documentation for the message broker. Some of this can be found at the following Web site:
http://www.ibm.com/software/integration/mqfamily/library/manualsa/.

For WebSphere Application Server, see *Implementing Adapters with WebSphere Application Server* and the documentation at:
http://www.ibm.com/software/webservers/appserv/library.html.

Prerequisites

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

Installing the TPI Server

The Trading Partner Interchange (TPI) Adapter, version 3.4.x, is supported for use with the Trading Partner Interchange Server vsersions 4.1 and 4.2.

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 Solaris

Before installing TPI on a Solaris 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.

Installing the TPI adapter and related files

For information on installing WebSphere Business Integration adapter products, refer to the *Installation Guide for WebSphere Business Integration Adapters*, located in the WebSphere Business Integration Adapters Infocenter at the following site:

http://www.ibm.com/websphere/integration/wbiadapters/infocenter

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 connector and TPI server run in the same address space, 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*.

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.

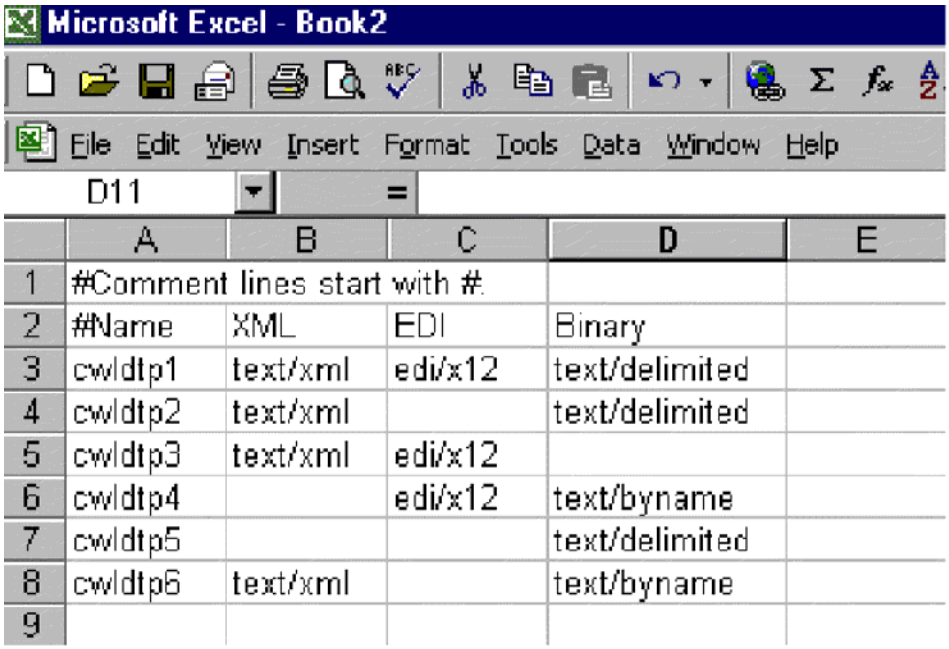
Creating the Trading Partner configuration file

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 12.

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 supplied with the TPI Server to run the TPI Server. Using this JVM, the TPI connector starts as a java process, and from within this process, starts the TPI Server. 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 Solaris. Do the following steps 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.

The JVM Supplied with the TPI Server must also be configured so that the TPI Adapter can successfully instantiate the TPI Server. To configure the TPI supplied JVM:

On Windows:

1. Locate the `\dependencies\win` folder in the TPI Adapter's directory: `%CROSSWORLDS%\connectors\TPI`.
2. Also locate the `\cijre` folder in the TPI home directory

3. Copy the contents of the %CROSSWORLDS%\connectors\TPI\dependencies\win folder into the %CYCLONEHOMEDIR%\cijre folder.
4. The following files should now exist in %CYCLONEHOMEDIR%\cijre
   - \bin\orb.dll
   - \bin\orb.dll
   - \lib\orb.properties
   - \lib\ext\ibmorb.jar
   - \lib\ext\ibmext.jar

On Solaris:

1. Locate the /dependencies/sol folder in the TPI Adapter's directory: ${CROSSWORLDS}/connectors/TPI.
2. Also locate the /cijre folder in the TPI Server's home directory. This will be identified as ${CYCLONEHOMEDIR}.
3. Copy the contents of the ${CROSSWORLDS}/connectors/TPI/dependencies/sol folder into the ${CYCLONEHOMEDIR}/cijre folder.
4. The following files should now exist in ${CYCLONEHOMEDIR}/cijre
   - \lib\orb.properties
   - \lib\ext\ibmorb.jar
   - \lib\ext\ibmext.jar
   - \lib\sparc\liborb.so
   - \lib\sparc\liborb_g.so

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
- ArchiveProcessedDocDirSee

See "Connector-specific properties" on page 12 for more information.

Running the connector and TPI Server as a Windows service

You can run the TPI connector and server as a Windows 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 43

Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, "Standard configuration properties for connectors," on page 27 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 1 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 1. 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.
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 2 lists the connector-specific properties for the connector. See the sections that follow for explanations of the properties.

Table 2. Connector-specific configuration properties

Name	Possible values	Default value
ArchiveProcessedDocDir	*<valid directory path>*	
ArchiveProcessedDocInfo	`true` or `false`	true
BackupRequired	`true` or `false`	true

Table 2. Connector-specific configuration properties (continued)

Name	Possible values	Default value
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.

Creating multiple connector instances

Creating multiple instances of a connector is in many ways the same as creating a custom connector. You can set your system up to create and run multiple instances of a connector by following the steps below. You must:

- Create a new directory for the connector instance
- Make sure you have the requisite business object definitions
- Create a new connector definition file
- Create a new start-up script

Create a new directory

You must create a connector directory for each connector instance. This connector directory should be named:

```
ProductDir\connectors\connectorInstance
```

where `connectorInstance` uniquely identifies the connector instance.

If the connector has any connector-specific meta-objects, you must create a meta-object for the connector instance. If you save the meta-object as a file, create this directory and store the file here:

```
ProductDir\repository\connectorInstance
```

Create business object definitions

If the business object definitions for each connector instance do not already exist within the project, you must create them.

1. If you need to modify business object definitions that are associated with the initial connector, copy the appropriate files and use Business Object Designer to import them. You can copy any of the files for the initial connector. Just rename them if you make changes to them.
2. Files for the initial connector should reside in the following directory:

   ```
   ProductDir\repository\initialConnectorInstance
   ```

Any additional files you create should be in the appropriate `connectorInstance` subdirectory of `ProductDir\repository`.

Create a connector definition

You create a configuration file (connector definition) for the connector instance in Connector Configurator. To do so:

1. Copy the initial connector's configuration file (connector definition) and rename it.
2. Make sure each connector instance correctly lists its supported business objects (and any associated meta-objects).
3. Customize any connector properties as appropriate.

Create a start-up script

To create a startup script:

1. Copy the initial connector's startup script and name it to include the name of the connector directory:

   dirname
2. Put this startup script in the connector directory you created in "Create a new directory" on page 15.
3. Create a startup script shortcut (Windows only).
4. Copy the initial connector's shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

You can now run both instances of the connector on your integration server at the same time.

For more information on creating custom connectors, refer to the *Connector Development Guide for C++ or for Java*.

Starting the connector

A connector must be explicitly started using its **connector start-up script**. The startup script should reside in the connector's runtime directory:

ProductDir`\connectors\`*connName*

where *connName* identifies the connector. The name of the startup script depends on the operating-system platform, as Table 3 shows.

Table 3. Startup scripts for a connector

Operating system	Startup script
UNIX-based systems	`connector_manager_`*connName*
Windows	`start_`*connName*`.bat`

You can invoke the connector startup script in any of the following ways:

- On Windows systems, from the **Start** menu

  Select **Programs>IBM WebSphere Business Integration Adapters>Adapters>Connectors**. By default, the program name is "IBM WebSphere Business Integration Adapters". However, it can be customized. Alternatively, you can create a desktop shortcut to your connector.
- From the command line
  - On Windows systems:

```
start_connName connName brokerName [-cconfigFile ]
```

– On UNIX-based systems:

```
connector_manager_connName -start
```

where *connName* is the name of the connector and *brokerName* identifies your integration broker, as follows:

– For WebSphere InterChange Server, specify for *brokerName* the name of the ICS instance.

– For WebSphere message brokers (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, or WebSphere Business Integration Message Broker) or WebSphere Application Server, specify for *brokerName* a string that identifies the broker.

Note: For a WebSphere message broker or WebSphere Application Server on a Windows system, you *must* include the `-c` option followed by the name of the connector configuration file. For ICS, the `-c` is optional.

- From Adapter Monitor (WebSphere Business Integration Adapters product only), which is launched when you start System Manager

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.

- From System Monitor (WebSphere InterChange Server product only)

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.

- On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector starts when the Windows system boots (for an Auto service) or when you start the service through the Windows Services window (for a Manual service).

For more information on how to start a connector, including the command-line startup options, refer to one of the following documents:

- For WebSphere InterChange Server, refer to the *System Administration Guide*.
- For WebSphere message brokers, refer to *Implementing Adapters with WebSphere Message Brokers*.
- For WebSphere Application Server, refer to *Implementing Adapters with WebSphere Application Server*.

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 Windows and Solaris systems, the file is located in the %CYCLONEHOMEDIR%/cijre/lib/security directory.

Stopping the connector

The way to stop a connector depends on the way that the connector was started, as follows:

- If you started the connector from the command line, with its connector startup script:
  - On Windows systems, invoking the startup script creates a separate "console" window for the connector. In this window, type "Q" and press Enter to stop the connector.
  - On UNIX-based systems, connectors run in the background so they have no separate window. Instead, run the following command to stop the connector:

    `connector_manager_`*`connName`* `-stop`

    where *connName* is the name of the connector.
- From Adapter Monitor (WebSphere Business Integration Adapters product only), which is launched when you start System Manager

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- From System Monitor (WebSphere InterChange Server product only)

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector stops when the Windows system shuts down.

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

Note: The adapter for TPI is also delivered with a datahandler for RNIF over TPI. See Appendix D, "Adapter for Trading Partner Interchange sample scenarios," on page 69.

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 4. 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 20 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 5 lists the attributes of the meta-object. All of the listed attributes must

be defined in the business object definition.

Table 5. 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 the connector component of WebSphere Business Integration adapters. The information covers connectors running on the following integration brokers:

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator, WebSphere MQ Integrator Broker, and WebSphere Business Integration Message Broker, collectively referred to as the WebSphere Message Brokers (WMQI).
- WebSphere Application Server (WAS)

Not every connector makes use of all these standard properties. When you select an integration 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 added in this release.

New properties

- XMLNameSpaceFormat

Deleted properties

- RestartCount

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 one of the WebSphere message brokers, 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 6 on page 29 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, as standard property dependencies are based on `RepositoryDirectory`.

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 6. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdminInQueue	Valid JMS queue name	*CONNECTORNAME* /ADMININQUEUE	Component restart	Delivery Transport is JMS
AdminOutQueue	Valid JMS queue name	*CONNECTORNAME*/ADMINOUTQUEUE	Component restart	Delivery Transport is JMS
AgentConnections	1-4	1	Component restart	Delivery Transport is MQ or IDL: Repository directory is <REMOTE>
AgentTraceLevel	0-5	0	Dynamic	
ApplicationName	Application name	Value specified for the connector application name	Component restart	
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	1	Component restart	Repository directory is <REMOTE>
ContainerManagedEvents	No value or JMS	No value	Component restart	Delivery Transport is JMS
ControllerStoreAndForwardMode	true or false	True	Dynamic	Repository directory is <REMOTE>
ControllerTraceLevel	0-5	0	Dynamic	Repository directory is <REMOTE>
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	Component restart	JMS transport only
DeliveryTransport	MQ, IDL, or JMS	JMS	Component restart	If Repository directory is local, then value is JMS only
DuplicateEventElimination	True or False	False	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	Component restart	JMS transport only

Table 6. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
jms.FactoryClassName	CxCommon.Messaging.jms .IBMMQSeriesFactory or CxCommon.Messaging .jms.SonicMQFactory or any Java class name	CxCommon.Messaging. jms.IBMMQSeriesFactory	Component restart	JMS transport only
jms.MessageBrokerName	If FactoryClassName is IBM, use crossworlds.queue. manager. If FactoryClassName is Sonic, use localhost:2506.	crossworlds.queue.manager	Component restart	JMS transport only
jms.NumConcurrentRequests	Positive integer	10	Component restart	JMS transport only
jms.Password	Any valid password		Component restart	JMS transport only
jms.UserName	Any valid name		Component restart	JMS transport only
JvmMaxHeapSize	Heap size in megabytes	128m	Component restart	Repository directory is <REMOTE>
JvmMaxNativeStackSize	Size of stack in kilobytes	128k	Component restart	Repository directory is <REMOTE>
JvmMinHeapSize	Heap size in megabytes	1m	Component restart	Repository directory is <REMOTE>
ListenerConcurrency	1- 100	1	Component restart	Delivery Transport must be MQ
Locale	en_US, ja_JP, ko_KR, zh_CN, zh_TW, fr_FR, de_DE, it_IT, es_ES, pt_BR **Note:** This is a subset of the supported locales.	en_US	Component restart	
LogAtInterchangeEnd	True or False	False	Component restart	Repository Directory must be <REMOTE>
MaxEventCapacity	1-2147483647	2147483647	Dynamic	Repository Directory must be <REMOTE>
MessageFileName	Path or filename	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	Repository Directory must be <REMOTE>

Table 6. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
OADMaxNumRetry	A positive number	1000	Dynamic	Repository Directory must be <REMOTE>
OADRetryTimeInterval	A positive number in minutes	10	Dynamic	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	
PollQuantity	1-500	1	Agent restart	JMS transport only: Container Managed Events is specified
PollStartTime	HH:MM(HH is 0-23, MM is 0-59)	HH:MM	Component restart	
RepositoryDirectory	Location of metadata repository		Agent restart	For ICS: set to <REMOTE> For WebSphere MQ message brokers and WAS: set to C:\crossworlds\ repository
RequestQueue	Valid JMS queue name	*CONNECTORNAME*/REQUESTQUEUE	Component restart	Delivery Transport is JMS
ResponseQueue	Valid JMS queue name	*CONNECTORNAME*/RESPONSEQUEUE	Component restart	Delivery Transport is JMS: required only if Repository directory is <REMOTE>
RestartRetryCount	0-99	3	Dynamic	
RestartRetryInterval	A sensible positive value in minutes: 1 - 2147483547	1	Dynamic	
RHF2MessageDomain	mrm, xml	mrm	Component restart	Only if Delivery Transport is JMS and WireFormat is CwXML.

Table 6. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
SourceQueue	Valid WebSphere MQ name	*CONNECTORNAME*/SOURCEQUEUE	Agent restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	Component restart	Delivery Transport is JMS
SynchronousRequestTimeout	0 - any number (millisecs)	0	Component restart	Delivery Transport is JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	Component restart	Delivery Transport is JMS
WireFormat	CwXML, CwBO	CwXML	Agent restart	CwXML if Repository Directory is not <REMOTE>: CwBO if Repository Directory is <REMOTE>
WsifSynchronousRequest Timeout	0 - any number (millisecs)	0	Component restart	WAS only
XMLNameSpaceFormat	short, long	short	Agent restart	WebSphere MQ message brokers and 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

Applicable only if RepositoryDirectory is <REMOTE>.

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, WebSphere message brokers (WMQI, WMQIB or WBIMB) 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 `ascii7` for this property.

By default, a subset of supported character encodings only is displayed 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.

ConcurrentEventTriggeredFlows

Applicable only if `RepositoryDirectory` is <REMOTE>.

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 No value.

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

- PollQuantity = 1 to 500
- SourceQueue = CONNECTORNAME/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

Applicable only if RepositoryDirectory is <REMOTE>.

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

Applicable only if RepositoryDirectory is <REMOTE>.

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

DeliveryQueue

Applicable only if DeliveryTransport is JMS.

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

The default value is CONNECTORNAME/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 the RepositoryDirectory is a local directory, 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

Applicable only if RespositoryDirectory is <REMOTE>.

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 RepositoryDirectory is <REMOTE>.

Specifies whether the connector uses the automatic and remote restart feature. This feature uses the MQ-triggered Object Activation Daemon (OAD) to restart the connector after an abnormal shutdown, or to start a remote connector from System Monitor.

This property must be set to trueto enable the automatic and remote restart feature. For information on how to configure the MQ-triggered OAD feature. see the *Installation Guide for Windows* or *for UNIX*.

The default value is false.

OADMaxNumRetry

Valid only when the RepositoryDirectory is <REMOTE>.

Specifies the maximum number of times that the MQ-triggered OAD automatically attempts to restart the connector after an abnormal shutdown. The OADAutoRestartAgent property must be set to true for this property to take effect.

The default value is 1000.

OADRetryTimeInterval

Valid only when the RepositoryDirectory is <REMOTE>.

Specifies the number of minutes in the retry-time interval for the MQ-triggered OAD. If the connector agent does not restart within this retry-time interval, the connector controller asks the OAD to restart the connector agent again. The OAD repeats this retry process as many times as specified by the OADMaxNumRetry property. The OADAutoRestartAgent property must be set to true for this property to take effect.

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 CONNECTOR/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 a WebSphere message broker or WAS, this value must be set to <*local directory*>.

ResponseQueue

Applicable only if DeliveryTransport is JMS and required only if RepositoryDirectory is <REMOTE>.

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

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. Possible values ranges from 1 to 2147483647.

The default is 1.

RHF2MessageDomain

WebSphere message brokers and WAS only.

This property allows you to configure the value of the field domain name in the JMS header. When data is sent to WMQI over JMS transport, the adapter framework writes JMS header information, with a domain name and a fixed value of `mrm`. A connfigurable domain name enables users to track how the WMQI 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

Applicable only if `DeliveryTransport` is JMS and ContainerManagedEvents is specified.

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 34.

The default value is `CONNECTOR/SOURCEQUEUE`.

SynchronousRequestQueue

Applicable only if `DeliveryTransport` is JMS.

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.

The default is `CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE`

SynchronousResponseQueue

Applicable only if `DeliveryTransport` is JMS.

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.

The default is CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE

SynchronousRequestTimeout

Applicable only if DeliveryTransport is JMS.

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 RepositoryDirectory is a local directory, the setting is CwXML.
- If the value of RepositoryDirectory is <REMOTE>, the setting isCwBO.

WsifSynchronousRequest 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.

XMLNameSpaceFormat

WebSphere message brokers and WAS integration broker only.

A strong property that allows the user to specify short and long name spaces in the XML format of business object definitions.

The default value is short.

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, WebSphere MQ Integrator Broker, and WebSphere Business Integration Message Broker, collectively referred to as the WebSphere Message Brokers (WMQI)
- WebSphere Application Server (WAS)

You use Connector Configurator to:

- Create a **connector-specific property template** for configuring your connector.
- Create a **connector configuration file**; you must create one configuration file for each connector you install.
- Set properties in a configuration file.
  You may need to modify the default values that are set for properties in the connector templates. You must also designate supported business object definitions and, with ICS, maps for use with collaborations as well as specify messaging, logging and tracing, and data handler parameters, as required.

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 44).

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 45 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
- From System Manager

Running Configurator in stand-alone mode

You can run Connector Configurator independently and work with connector configuration files, irrespective of your broker.

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, WebSphere Message Brokers or WAS, depending on your broker.

You may choose 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 49.)

Running Configurator from System Manager

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** icon 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, WebSphere Message Brokers or WAS, 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 45.
- 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:
   - **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.

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:

- **General:**
  Property Type
  Updated Method
  Description
- **Flags**
  Standard flags
- **Custom Flag**
  Flag

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. It also allows editable values. 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 top left-hand corner of the value table and click **Add**. 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 **Dependences - Connector-Specific Property Template** 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, WebSphere Message Brokers or WAS connectivity.
- Complete the remaining fields in the **New Connector** window, as described later in this chapter.

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, and must be consistent with the file name for a connector that is installed on the system.

     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 WebSphere Message Brokers or WAS.
   - **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>To Project**. To save to a project, 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 resave the 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 52..
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 Handler (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. To add a child property, right-click on the parent row number and click **Add child**.
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."

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

Refer to the descriptions of update methods found in the *Standard configuration properties for connectors* appendix, under "Setting and updating property values" on page 28.

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.

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.

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

If a WebSphere Message Broker is your broker

If you are working in stand-alone mode (not connected to System Manager), you must enter the business 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 object available from the Integration Component Library project to which the connector belongs. Select the business object you want from the list.

The **Message Set ID** is an optional field for WebSphere Business Integration Message Broker 5.0, and need not be unique if supplied. However, for WebSphere MQ Integrator and Integrator Broker 2.1, you must supply a unique **ID**.

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.

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, save the connector configuration file. Connector Configurator saves the file 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:

- From System Manager, as a file with a `*.con` extension in an Integration Component Library, or
- In a directory that you specify.
- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder.

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 WebSphere Message Brokers: *Implementing Adapters with WebSphere Message Brokers*
- 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. IBM WebSphere DataHandler for RNIF over TPI

The IBM WebSphere Business Integration Data Handler for RosettaNet Implementation Framework (RNIF) over TPI, called the *TPI-RNIF data handler* facilitates the exchange of RosettaNet documents with trading partners when using the Trading Partner Interchange (TPI) server. The data handler creates and parses XML messages specific to TPI server requirements.

This appendix assumes that the reader is familiar with the basic terminology and functionality associated with the Trading Partner Interchange product (TPI), the RosettaNet Implementation Framework (RNIF), the IBM WebSphere Adapter for TPI, and the IBM WebSphere Business Integration Data Handler for XML. For further information about these products and standards, please see the following:

- The *Administrator's Guide,* included with the TPI Server
- http://www.rosettanet.org
- Previous chapters within this guide, *The Adapter for Trading Partner Interchange User Guide*
- The IBM WebSphere Business Integration Adapters *Data Handler Guide*

A number of the examples provided in this appendix use the Chemical Industry Data Exchange (CDIX) format. This format is a specialized usage of RosettaNet that has been implemented extensively with the IBM WebSphere Adapter for TPI.

Overview

When messages are exchanged between the TPI server and the IBM WebSphere adapter for TPI, the TPI server uses a Message Control Document (MCD) to describe both the trading partners involved and the RosettaNet data to be exchanged. The MCD is a TPI-specific XML document that the TPI server uses to process RosettaNet documents. The TPI server converts the information contained in the MCD to a RosettaNet document received from or sent to the trading partner.

The TPI-RNIF data handler simplifies the process of creating and parsing MCDs. It recognizes and manipulates the various RNIF headers, business signals and Partner Interface Processes (PIP) messages, if any, embedded in the MCD. It also usually minimizes the number of business object definitions you must manage when multiple PIPs are used.

The TPI-RNIF data handler determines the appropriate business object definition to which an MCD should be mapped, and it identifies the various RNIF parts, business signals and PIP documents of the RosettaNet payload embedded in the MCD. Once it has identified these parts, the TPI-RNIF data handler internally calls the IBM WebSphere Business Integration Data Handler for XML, which then converts the parts to their appropriate business object types. Therefore, business object definitions used with the TPI-RNIF data handler must conform both to the XML data handler's business object requirements and to the TPI-RNIF data handler's naming conventions described in this appendix.

Software prerequisites

The following software must be installed before you can use the TPI-RNIF data handler:

- IBM WebSphere Adapter for Trading Partner Interchange, v 3.4.0 or higher
- IBM WebSphere Data Handler for XML, v 2.5.0 or higher
- Trading Partner Interchange, v. 4.1.0 or higher with support for MCD v. 2.0

To build, the TPI-RNIF data handler requires both the WebSphere Business Integration Adapter Framework libraries and the WebSphere Business Integration Data Handler for XML class files.

At run time, the WebSphere Data Handler for XML must be present.

Installation

The TPI Server must be installed. Ensure that RosettaNet is listed as an inbound and or outbound protocol for each RosettaNet trading partner. Also, ensure that your trading partner company has RossettaNet/MCD listed as a supported XML document type. For instructions on installing and configuring the TPI Server, see the *Administrator's Guide* included with the TPI Server.

For information on installing WebSphere Business Integration adapter products, refer to the *Installation Guide for WebSphere Business Integration Adapters*, located in the WebSphere Business Integration Adapters InfoCenter at the following site:

http://www.ibm.com/websphere/integration/wbiaadapters/infocenter

The TPI-RNIF data handler installation files are included as part of the CwDataHandler.jar installed in the DataHandlers directory of the adapter installation. The installed directory structure is:

\DataHandlersw\CwDataHandler.jar

Make sure that you have completed the following tasks during installation of the adapter:

1. Configured the adapter to use a data handler configuration meta object that has configuration information for the TPI-RNIF data handler.

   You can do this by subscribing the adapter to the sample business object

   `MO_DataHandler_CIDX`

   included with the installation, and then set the connector-specific property

   `DataHandlerDefaultMO=MO_DataHandler_CIDX`

   .
2. Configured the TPI-RNIF data handler in your data handler meta object.

   For example, if you use

   `MO_DataHandler_CIDX`

   , check that `MO_DataHandler_ChemXMLConfig` has values for `TopBOPrefix`, `BOPrefix`, etc. that exactly match your business object definitions. See "Creating the Trading Partner configuration file" on page 9.

3. Configured the adapter to use the TPI-RNIF data handler for both sending and receiving trading partners.

   You can do this through the trading partner configuration file specified for the adapter. If you use

   `MO_DataHandler_CIDX`

   , specify mime-type ″text/tpi_rnif″ in this file for those trading partners with whom you want to exchange data.
4. Set the attribute

   `DocumentType=CW_RNIF`

   , in your TPI child meta object that is included in each request sent to adapter.

   If you do not set this attribute, the TPI server will not recognize the document as an MCD, and it will send it as raw XML to the target trading partner. See “Child meta-object attributes” on page 20.

Configuring the TPI RNIF data handler

You configure the TPI-RNIF data handler through data handler meta-objects. The required properties listed below are specific to the TPI-RNIF data handler. However, since the TPI-RNIF data handler internally extends the existing XML data handler functionality, see the *Data Handler Guide* for information about additional meta object properties required by the XML data handler.

Table 7. TPI-RNIF data handler required meta object properties

Property	Description	Required	Default Value
TopBOPrefix	Resolves the wrapper business object for a given service content document. Sample values include `RNETOBJ` and `CIDXOBJ` .	No, but recommended	None
BOPrefix	Like TopBOPrefix, but converts the RNIF message parts (such the Preamble, Service Header, etc.) embedded in the MCD to business objects. Sample values include `RNET` and `CIDX`	No, but recommended	None
PIPSpecificSignalList	List of semi-colon delimited doctypes for which the data handler should include the PIP identifier when resolving the wrapper business object name.	No	None
ClassName		Yes	com.crossworlds.DataHandle

Understanding business object structure

The TPI-RNIF data handler places specific requirements on the top-level strucure of any business object it processes. It requires that the top level business object acts as a wrapper, and that the wrapper object contains multiple child objects that represent the MCD and the various RNIF parts contained within the MCD.

Example of a wrapper object structure:

```
Wrapper BO
      |MessageControlDocument BO
      |Preamble BO
      |ServiceHeader BO
      |ServiceContent BO
```

Naming the wrapper business object

The data handler requires specific naming conventions in order for it to identify the appropriate business object definitions to use when converting an inbound MCD to a business object instance. The wrapper object must be named according to the following convention:

```
<TopBOPrefix_>Version_ServiceContentDoctype
```

where

`TopBOPrefix`

is optional. The TPI-RNIF data handler prepends the value of

`TopBOPrefix`

to the doctype to determine the appropriate wrapper. The doctype used to derive the object name is always that of the RNIF service content part contained within the MCD, such as OrderCreate, ReceiptAcknowledgement, etc.

The

`Version`

part of the name is determined by the value provided in the packaging protocol description of the MCD.

Example of the packaging protocol description:

```
<MessageControlDocument>
     <PackagingProtocol>
          <Standard>CIDX</Standard>
          <Version>2.0</Version>
     </PackagingProtocol>
</MessageControlDocument>
```

Note: If the version name contains periods, they are replaced with underscores when the business object name is resolved.

Example of resolving a business object name:

1. Set

   `TopBOPrefix=CIDXOBJ`

   in the TPI-RNIF data handler.
2. Pass an MCD containing a RNIF 2.0-structured OrderCreate request to the data handler.

The data handler maps the MCD to

`CIDXOBJ_2_0_OrderCreate`

.

Naming child business objects

Child business objects must also conform to naming conventions so that the data handler can process them correctly. The child object that represents the MCD envelope must match the following structure:

`BOPrefix_Doctype`

An example of a child object that represents the MCD envelope:

`CIDX_MessageControlDocument`

Child objects that represent RNIF message parts, including preamble, service header, and service content must also be properly named and match the following structure:

`BOPrefix_Version_Doctype`

where

`DocType`

represents the RNIF message part. An example of a child object that represents part of the RNIF message:

`CIDX_2_0_Preamble`

Defining child object attributes

The data handler requires that you define at least the following four child object attributes in the wrapper:

- `MessageControlDocument`
- `Preamble`
- `ServiceHeader`
- `ServiceContent`

The following example shows the required attributes matched to a RosettaNet business object definition that adheres to the RNIF 2.0 specification.

`RNETOBJ_2_0_OrderCreate`

is the wrapper object.

`TopBOPrefix=RNETOBJ`

`BOPrefix=RNET`

Table 8. Required attributes matched to type

Attribute Name	Attribute Type
MessageControlDocument	RNET_MessageControlDocument
Preamble	RNET_2_0_Preamble
ServiceHeader	RNET_2_0_ServiceHeader
ServiceContent	RRNET_2_0_OrderCreate

Note: When defining attributes, do not change the default values for fields that are not shown in the example. These fields include cardinality, application-specific information, etc.

Using PIP-specific signals

You can configure the data handler to include the PIP identifier in the top-level object wrapper. This is optional, but it can be useful when RosettaNet signal messages are being exchanged. If you want to include PIP-specific signals, you must select the

`PIPSpecificSignals`

option within the data handler meta object, and then specify all doctypes. Doctypes must be delimited with semicolons and without spaces. Whenthe data handler receives an MCD document, it checks to see if the service content doctype is of a type specified in the option PIPSpecificSignals. If it is, the data handler extracts the PIP identifier from the MCD and seeks the wrapper object name based on the following syntax:

`<TopBOPrefix_>Version_PIPType_ServiceContentDoctype`

When RosettaNet signal messages are being exchanged, and the data handler is configured to use the included PIP identifier, the data handler can convert the generic signal message to a distinct business object depending on the PIP for which the signal was generated. This means that processes can subscribe to only those signals of interest.

Without PIP-specific signals, the data handler might convert a receipt acknowledgement signal to a generic wrapper object, such as the following:

`CIDXOBJ_1_1_ReceiptAcknowledgement`

WithPIP-specific signals, the data handler can map the receipt acknowledgement signal to a specific wrapper object, such as the following:

`CIDXOBJ_1_1_2A1_ReceiptAcknowledgement`

The following example shows the required attributes matched to a RosettaNet business object definition that adheres to the RNIF 1.1 specification, and uses PIP-specific identifiers. The example assumes that

`PIPSpecificSignals`

contains the doctype

`OrderCreate`

.

`RNETOBJ_1_1_E41_OrderCreate`

is the wrapper object.

`TopBOPrefix=RNETOBJ`

`BOPrefix=RNET`

Table 9. Required attributes matched to type, using PIP-specific identifiers

Attribute Name	Attribute Type
MessageControlDocument	RNET_MessageControlDocument
Preamble	RNET_1_1_Preamble
ServiceHeader	RNET_1_1_ServiceHeader

Table 9. Required attributes matched to type, using PIP-specific identifiers (continued)

Attribute Name	Attribute Type
ServiceContent	RNET_1_1_E41_OrderCreate

Note: RosettaNet standards specify that OrderCreate documents fall under PIP E41.

Using delivery header

You can define delivery header, a fifth attribute, in the wrapper object. It is optional.

During request processing, if the data handler finds and populates an existing child attribute with this name, the data handler serializes the child object and adds it to the outbound MCD.

During event notification, if the MCD contains a delivery header structure, the data handler checks to see if the wrapper object contains a delivery header attribute. If it does, the data handler maps the delivery header structure to this attribute.

If the data handler cannot find a delivery header attribute within the wrapper object, it generates a warning. However, if the MCD does not contain a delivery header structure, the data handler does not generate a warning even if the wrapper object contains a delivery header attribute.

The following example shows the required attributes matched to a RosettaNet business object definition that adheres to the RNIF 2.0 specification. This wrapper object uses PIP-specific signals and the DeliveryHeader attribute.

```
RNETOBJ_2_0_E41_OrderCreate
```

is the wrapper object.

```
TopBOPrefix=RNETOBJ
BOPrefix=RNET
```

Table 10. Required attributes matched to type, using delivery header

Attribute Name	Attribute Type
MessageControlDocument	RNET_MessageControlDocument
Preamble	RNET_2_0_Preamble
ServiceHeader	RNET_2_0_ServiceHeader
ServiceContent	RNET_2_0_E41_OrderCreate
DeliveryHeader	RNET_2_0_ServiceHeader

Using key link attributes

It can be useful to include meaningful keys in a top-level business object. You can do this with the TPI-RNIF data handler by using key links. Key links are defined and supported only in the wrapper, and they instruct the data handler to copy key information from within one of the child business objects to the wrapper object.

When the data handler receives and MCD and constructs the corresponding wrapper object, it populates any key link attributes it finds in the wrapper with nested attributes in the child objects.

To define a key link, do the following:

1. Create an attribute in the wrapper business object.

   You can use any name except those reserved for the four required attributes described earlier.
2. Specify application-specific text according to the syntax

   `type=key;ref=someInnerAttribute`

   where

   `someInnerAttribute`

   defines the path to an attribute, relative to the wrapper object.

The following example shows how the TPI-RNIF data handler processes key links.

Key link attribute name:

`InstanceId`

Application-specific information:

`type=key;ref=MessageControlDocument.TLO.ProcessInfo.PIPInstanceID`

The data handler resolves this by copying the value of the child attribute

`PIPInstanceID`

to the wrapper attribute

`InstanceId`

.

Creating business object definitions

In order to create a business object structure that is compatible with the TPI-RNIF data handler, you must create three types of business object definitions.

- Create a business object definition for the MCD envelope.

  Only one MCD business object definition is needed. Create it using the XML Object Disvovery Agent (ODA), and use the MCD XSD from your TPI server installation as the input. Be sure that the business object prefix you specify for the XML ODA is the same as what you specify in the TPI-RNIF data handler meta object. See the section, "Using an XML ODA to create business object definitions" in the *Data Handler Guide*.

  With most MCD business object definitions, you will need to complete one additional step using the Business Object Designer tool. Use this tool to modify the application-specific information of the attribute

  `MessageControlDocument.ROOT.ManifestInfo.MessageContentInfo.Body.Body`

  so that it is of the type "cdata." For example,

  `mcd:Body;type=pcdata;notag`

  should be modified to

  `mcd:Body;type=cdata;notag`

- Create business object definitions for the RNIF headers, business signals and messages.

  One business object definition is needed for each type of service content message you plan to exchange, and one preamble and service header business object is needed for each RNIF version you plan to support. Include the RNIF version number in the BOPrefix, such as

  ```
  RNET_2_0
  ```

  . Use the XML ODA to create these business object definitions as well.
- Create a wrapper business object definition using the Business Object Designer tool.

  Name the wrapper object according the naming conventions described earlier, and then add the MCD envelope business object, the appropriate RNIF header business objects, and a signal or PIP-specific business object, if needed.

  You must also define a child meta object to provide routing information required by the adapter for TPI. See"Child meta-object attributes" on page 20.

Appendix D. Adapter for Trading Partner Interchange sample scenarios

This appendix provides sample scenarios to help you understand using the Adapter for Trading Partner InterChange (TPI) to exchange XML documents between trading partners. The scenarios cover two methods of implementing an integration solution using the adapter for TPI. The scenarios do not reflect an actual installation or a real trading situation, and there may be other possible solutions. The scenarios cover these implementations:

1. Using the Adapter for Trading Partner Interchange with IBM WebSphere InterChange Server.
2. Using the Adapter for Trading Partner Interchange with IBM WebSphere MQ Integrator Broker.

Pre-installation notes and assumptions

1. You are experienced with the IBM connectivity and integration software you will be using, and with the TPI server.
2. You have two computers (physical machines) available.
3. On the first computer, you have installed the the IBM integration and connectivity software you will be using.

   This computer will be referred to as ″IBM″ throughout the scenarios in this appendix.

   For installations using Websphere InterChange Server:
   - WebSphere InterChange Server 4.x
   - WebSphere Adapter for TPI
   - TPI Server
   - WebSphere MQ

   For installations using IBM WebSphere MQ Integrator Broker:
   - WebSphere Business Integration Adapters 4.x
   - WebSphere Adapter for TPI
   - TPI Server
   - WebSphere MQ Integrator Broker
4. On the second computer, you have installed the TPI server.

   This computer will be referred to as ″TPI″ throughout the scenarios in this appendix.

Note: Whenever %CROSSWORLDS% is mentioned in this document, it refers to one of the following:
- **If you are using WebSphere MQ Integrator Broker :** It refers to the folder containing your WebSphere Business Integratation Adapters installation.
- **If you are using Websphere InterChange Server:** It refers to the folder containing your IBM Websphere Integration Server installation.

All environment variables and file separators are specified in the Windows format. Please make the appropriate changes if running on AIX or Solaris. (ex. %CROSSWORLDS%\connectors would be ${CROSSWORLDS}/connectors)

Using Websphere InterChange Server

This fictitious scenario involves two trading partner companies, **IBM Corporation (IBM)** and **IBM Trading Partner (IBMTP)** that will exchange order information. **IBM** is using the IBM WebSphere Adapter for TPI and IBM Websphere InterChange Server for integration to and from a backend fictitious application called ″Port.″ This application will use the PortConnector.

Data flow summary

The repos_copy file contains two collaboration objects, each one designed to handle one direction of integration.

IBM to IBMTP

The first collaboration object is a simple pass through object called `Port_To_TPI`. It is handled according to the following data flow:

- It accepts a `TPI_Order` object from the PortConnector.
- It sends the `TPI_Order` object to the adapter for TPI.
- The adapter for TPI receives `TPI_Order` and does the following:
  1. Uses the XML DataHandler to convert it to an XML document.
  2. Bundles the XML document into HTTP.
  3. Sends the bundled HTTP to the TPI sever running at **IBMTP**, the trading partner site.

IBMTP to IBM

The second collaboration object is called `TPI_To_Port` and is used in the following data flow:

- **IBMTP** sends bundled HTTP containing an order XML document to the adapter for TPI residing at **IBM**.
- The adapter for TPI receives the bundled HTTP and does the following:
  1. Uses the XML data handler to convert the XML document to a business object.
  2. Sends the business object to `TPI_To_Port`, the second collaboration object.
- `TPI_To_Port` uses the PortConnector to send the business object to the Port application.

Installation of the sample scenario for interChange server

1. Set up **IBM Corporation** TPI server:
   a. Start the TPI Administrator Tool on the **IBM** machine.
   b. From the Company Profiles tab, import the company profile from the file: `%CROSSWORLDS%\connectors\TPI\samples\WebSphereICS\IBM\IBMCorporation_company.xml.`
   c. Leave the import password blank.
   d. Also from the Administrator Tool, select the partners tab, and import the partner profile from the file: `%CROSSWORLDS%\connectors\TPI\samples\WebSphereICS\IBM\IBMTradingPartner_partner.xml.`

      The company identification for this TPI Server will be **IBM**

e. Open the properties for the partner profile, and navigate to the bundled HTTP under the Outbound Transport tab. Change the URL from "http://IBMHost:4080/exchange/IBM" to fit your machine name.

f. Update the TPHost to the name of the IBM trading partner machine.

2. Set up **IBM Trading Partner** TPI server:

   a. Start the TPI Administrator Tool on the **IBMTP** machine.

   b. From the Company Profiles tab, import the company profile from the file: `%CROSSWORLDS%\connectors\TPI\samples\WebSphereICS\IBMTP\IBMTradingPartner_company.xml`.

   c. Also from within the Administrator Tool, select the partners tab, and import the partner profile from the file: `%CROSSWORLDS%\connectors\TPI\samples\WebSphereICS\IBMTP\IBMCorporation_partner.xml`.

      The company identification for this TPI Server will be **IBMTP**.

   d. Open the properties for the partner profile, and navigate to the bundled HTTP under the Outbound Transport tab. Change the URL from "http://IBMHost:4080/exchange/IBM" to fit your machine name.

   e. Update the IBMHost to the name of the IBM machine.

3. Load business objects into the repository:

   a. Start WebSphere InterChange Server from the IBM machine.

   b. Using the Business Object Designer, load the repository file:

      `Sample_TPI_Order_Objects.in`

      from the

      `%CROSSWORLDS%\connectors\TPI\samples\WebSphereICS\IBM`

      folder.

   c. Confirm that ten business objects have been loaded.

4. Load connectors into the repository:

   a. Using Connector Configurator, load the repository file:

      `Sample_TPI_Order_Connectors.in`

      from the

      `%CROSSWORLDS%\connectors\TPI\samples\WebSphereICS\IBM`

      folder.

   b. Confirm that the TPI Connector and and PortConnector definitions have been loaded.

5. Configure TPI Connector:

   a. Use your System Manager to select the TPI Adapter definition and launch the Connector Designer. Modify the following adapter configuaration properties to fit your file structure. If they don't exist, create these paths and files:

      - `TradingPartnerConfigurationFile`
      - `MetaEventDir`
      - `DocumentOutDir`
      - `ArchiveProcessedDocDir`

6. Load collaboration objects and templates into the repository:

   a. Use your System Manager to load the repository file labeled

      `Sample_TPI_Order_Collaborations.in`

located in the
`%CROSSWORLDS%\connectors\TPI\samples\WebSphereICS\IBM`

folder.

b. Confirm that the
`Order_PassThrough`

template definition, and the
`Port_To_TPI`

and
`TPI_To_Port`

collaboration objects have been loaded.

7. Compile the collaboration template. Using your system manager, select the Collaboration Templates folder, and then select Compile All.
8. Restart the WebSphere InterChange Server.

Running the service call request scenario

Complete these steps to run the service call scenario.

1. **On the IBM machine:**
   a. Start the WebSphere InterChange Server, and then start the adapter for TPI. (This automatically starts the TPI server.)
   b. Start one instance of the Visual Test Connector.
2. **On the IBMTP machine:** Start the TPI server.
3. Simulate the Port Connector:
   a. Use the Test Connector to define a profile for PortConnector.
   b. Select Connect Agent from the Test Connector's file menu to begin simulating the agent.
4. Load the test data:
   a. Use the Visual Test Connector simulating the PortConnector, and select Load BO from the Edit menu.
   b. Select and load the file:
   `%CROSSWORLDS%\connectors\TPI\samples\WebSphereICS\IBM\sampleOrderData.bo`
5. Send the test data:
   a. Use the Visual Test Connector simulating the PortConnector, and select the loaded test Business object.
   b. Select Send from the Request menu.
6. Confirm successful processing by checking the XLMin folder on the **IBMTP machine** for the XML document.

Running the polling scenario

Complete these steps to run the polling scenario.

1. **On the IBM machine:**
   a. Start the WebSphere InterChange Server, and then start the adapter for TPI. (This automatically starts the TPI server.)
   b. Start one instance of the Visual Test Connector.
2. **On the IBMTP machine:** Start the TPI server.
3. Simulate the Port Connector **On the IBM machine:**

a. Use the Test Connector to define a profile for PortConnector.
b. Select Connect Agent from the Test Connector's file menu to begin simulating the agent.

4. Send the sample data:
   a. **On the IBMTP machine:**Move the following file into the XMLout directory:
      `%CROSSWORLDS%\connectors\TPI\samples\WebSphereICS\IBMTP\sampleOrderData.xml`
   b. The TPI server running on the **IBMTP machine**should process this event and hand it off to the adapter for TPI running on the **IBM machine.**
5. Accept the request using the Visual Test Connector simulating the PortConnector. Reply with a successful response.
6. Confirm successful processing by checking the data in the accepted request.

Using WebSphere MQ Integrator Broker

This fictitious scenario involves two trading partner companies (″IBM Corporation″ and ″IBM Trading Partner″) that will be exchanging order information. **IBM Corporation (IBM)** is using the TPI Adapter with WebSphere MQ Integrator Broker to exchange XML documents between itself and its trading partner named **IBM Trading Partner (IBMTP)**.

There are two parts included in this sample scenario. Each is designed to handle one direction of message exchange. For the first, the WebSphere MQ Integrator Broker will place an event onto an MQ Series queue for the Adapter for TPI to read. The adapter will convert the message into an XML document using the XML DataHandler and send the document to the **IBM Trading Partner**.

The second part requires the **IBM Trading Partner** company to send an XML document to the **IBM Corporation** company. The TPI Adapter located at **IBM Corporation** will convert the XML document into an MQ Series message and place the message on an MQ Series queue for the WebSphere MQ Integrator Broker to read.

Note: In this sample, a real WebSphere MQ Integrator Broker will not be used. The TPI Adapter is simply going to read and write from MQ Series queues. The Visual Test Connector will be used to simulate the WebSphere MQ Integrator Broker.

Installation of the sample scenario

1. Set up **IBM Corporation** TPI server:
   a. Start the TPI Administrator Tool on the **IBM** machine.
   b. From the Company Profiles tab, import the company profile from the file: `%CROSSWORLDS%\connectors\TPI\samples\WebSphereMQIntegratorBroker \IBM\IBMCorporation_company.xml.`
   c. Leave the import password blank.
   d. Also from the Administrator Tool, click on the partners tab, and import the partner profile from the file: `%CROSSWORLDS%\connectors\TPI\samples\ WebSphereMQIntegratorBroker\IBM\IBMTradingPartner_partner.xml.`
   e. The company id for this TPI Server will be **IBM**
2. Set up **IBM Trading Partner** TPI server:
   a. Start the TPI Administrator Tool on the **IBMTP** machine.

b. From the Company Profiles tab, import the company profile from the file: %CROSSWORLDS%\connectors\TPI\samples\WebSphereMQIntegratorBroker\IBMTP\IBMTradingPartner_company.xml.

c. Also from the Administrator Tool, click on the partners tab, and import the partner profile from the file: %CROSSWORLDS%\connectors\TPI\samples\WebSphereMQIntegratorBroker\IBMTP\IBMCorporation_partner.xml.

d. The company id for this TPI Server will be **IBMTP**.

3. Set up MQ Series:
   a. Create and start an MQ Series queue manager with a running channel initiator and listener.
   b. Create the following named queues:
      - ADMININQUEUE
      - ADMINOUTQUEUE
      - DELIVERYQUEUE
      - FAULTQUEUE
      - REQUESTQUEUE
4. Open the start_TPIfile located in the %CROSSWORLDS%\connectors\TPI directory. Change the CYCLONEHOMEDIR value to the TPI home directory path. Save and close the file.
5. Configure the TPI Adapter and the Port Connector CFG file:
   a. Open the %CROSSWORLDS%\connectors\TPI\samples\WebSphereMQIntegratorBroker\TPIConnector.cfg file.
   b. The following properties in the file may need to be modified for your particular setup. See the User Guide for the IBM WebSphere Business Integration Adapter for TPI for detailed information. If the property represents a path or filename and it does not already exist, it needs to be created.
      - QueueManager
      - RepositoryDirectory
      - TradingPartnerConfigurationFile
      - MetaEventDir
      - DocumentOutDir
      - ArchiveProcessedDocDir
   c. Open the %CROSSWORLDS%\connectors\TPI\samples\WebSphereMQIntegratorBroker\PortConnector.cfg file. The following properties in the file may need to be modified for your particular setup. If the property represents a path or filename and it does not already exist, it needs to be created.
      - QueueManager
      - RepositoryDirectory
6. Specify the TPI Adapter CFG file:
   - For Windows: Open the properties of the shortcut for the TPI Adapter. As the last argument in the target, add "-c" + <full path and filename for the TPIConnector.cfg file>
     – For example: -c%CROSSWORLDS%\connectors\TPI\Samples\WebSphereMQIntegratorBroker\IBM\TPIConnector.cfg
   - For UNIX: Open the ${CROSSWORLDS}/bin/connector_manager_TPI file. Set the value of the AGENTCONFIG_FILE property to "-c" + <full path and filename for the TPIConnector.cfg file>
     – AGENTCONFIG_FILE=-c${CROSSWORLDS}/connectors/TPI/samples/WebSphereMQIntegratorBroker/IBM/TPIConnector.cfg

Running the request processing scenario

Complete these steps to run the request processing scenario.

1. **On the IBM machine:**
   a. Start the adapter for TPI. (This automatically starts the TPI server.)
   b. Start one instance of the Visual Test Connector.
2. **On the IBMTP machine:** Start the TPI server.
3. Simulate the Port Connector:
   a. Use the Visual Test Connector to define a profile for PortConnector.
   b. Select Open Profile from the Test Connectors File menu, and specify as the connector configuration file:
      `%CROSSWORLDS%\connectors\TPI\samples\WebSphereMQIntegratorBroker\IBM\PortConnector.cfg`
   c. Connect the agent.
4. Load the test data:
   a. Use the Visual Test Connector simulating the PortConnector, and select Load BO from the Edit menu.
   b. Select and load the file:
      `%CROSSWORLDS%\connectors\TPI\samples\WebSphereMQIntegratorBroker\IBM\sampleOrderData.bo`
5. Send the test data:
   a. Use the Visual Test Connector simulating the PortConnector, and select the loaded test Business object.
   b. Select Send from the Request menu.
6. Confirm successful processing by checking the XLMin folder on the **IBMTP machine** for the XML document.

Running the polling scenario

Complete these steps to run the polling scenario.

1. **On the IBM machine:**
   a. Start the adapter for TPI. (This automatically starts the TPI server.)
   b. Start one instance of the Visual Test Connector.
2. **On the IBMTP machine:** Start the TPI server.
3. Simulate the Port Connector:
   a. Use the Visual Test Connector to define a profile for PortConnector.
   b. Select Open Profile from the Test Connectors File menu, and specify as the connector configuration file:
      `%CROSSWORLDS%\connectors\TPI\samples\WebSphereMQIntegratorBroker\IBM\PortConnector.cfg`
   c. Connect the agent.
4. Send the sample data:
   a. **On the IBMTP machine:**Move the following file into the XMLout directory:
      `%CROSSWORLDS%\connectors\TPI\samples\WebSphereMQIntegratorBroker\IBMTP\sampleOrderData.xml`
   b. The TPI server running on the **IBMTP machine**should process this event and hand it off to the adapter for TPI running on the **IBM machine.**
5. Accept the request using the Visual Test Connector simulating the PortConnector. Reply with a successful response.
6. Confirm successful processing by checking the data in the accepted request.

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.

WebSphere Business Integration Adapter Framework V2.4.0

Printed in USA