IBM WebSphere Business Integration Adapters

Adapter for QAD MFG/PRO User Guide

V 2.0.0

12February2003

This edition of this document applies to the adapter for QAD MFG/PRO version 2.0.0 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.

© Copyright International Business Machines Corporation 2004. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2004

Figures

© Copyright IBM Corp. 2004

About this document

The IBM® WebSphere® Business Integration Adapter portfolio supplies integration connectivity for leading e-business technologies, enterprise applications, and 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, and business object development for the adapter for QAD MFG/PRO.

Audience

This document is for consultants, developers, and system administrators who support and manage the WebSphere business integration system at customer sites.

Prerequisites for this document

Users of this document should be familiar with the WebSphere business integration system, with business object and collaboration development, and with the QAD MFG/PRO application suite.

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, see the IBM WebSphere Business Integration Adapters InfoCenter:
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- For using adapters with WebSphere InterChange Server, see the IBM WebSphere InterChange Server InfoCenters:
  http://www.ibm.com/websphere/integration/wicserver/infocenter
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- For more information about WebSphere message brokers:
  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.

© Copyright IBM Corp. 2004

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	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 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 IBM WebSphere Business Integration Adapters product is installed.

New in this release

New in release 2.0.x

February 2004

The adapter supports MFG/PRO version eB2.

The adapter also incorporates the following changes:

- **Request processing**
  - The adapter uses a web services interface to handle broker requests
  - Requests are now synchronous
  - Requests have a new business object structure
  - The SQL Query Dispatcher has been removed because it is no longer needed for the new request processing style
- **Event processing**
  - The adapter supports triplet as well as QDoc-XML format in event processing.

December 2003

Adapter installation information has been moved from this guide. See Chapter 2 for the new location of this information.

Beginning with version 2.0.0, the adapter for QAD MFG/PRO is no longer supported on Microsoft Windows NT.

New in release 1.0.x

This is a new document.

© Copyright IBM Corp. 2004

Chapter 1. Overview

The connector for QAD(TM) MFG/PRO(R) is a runtime component of the WebSphere Business Integration Adapter for QAD MFG/PRO. The connector component allows the integration broker to exchange business objects with QAD MFG/PRO applications that send or receive data in the form of WebSphere MQ messages.

This chapter describes the connector component and the relevant business integration system architecture.It covers these topics:

Adapter environment

Before installing, configuring, and using the adapter, you must understand its environment requirements. They are listed in the following section.

Broker 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 2.0.0 of the adapter for QAD MFG/PRO is supported on the following adapter framework and integration brokers:

- **Adapter framework**:
  WebSphere Business Integration Adapter Framework versions 2.2, 2.3, 2.4.
- **Integration brokers:**
  - WebSphere InterChange Server, versions 4.2, 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
  - WebSphere Application Server Enterprise, version 5.0.2, with WebSphere Studio Application Developer Integration Edition, version 5.0.1

See *Release Notes* for any exceptions.

© Copyright IBM Corp. 2004

Note: For instructions on installing your integration broker and its prerequisites, see the following guides.
For WebSphere InterChange Server (ICS), see *IBM WebSphere InterChange Server System Installation Guide for UNIX* or *for Windows.*
For WebSphere message brokers, see *Implementing Adapters with WebSphere Message Brokers.*
For WebSphere Application Server, see *Implementing Adapters with WebSphere Application Server.*

Adapter platforms and databases

The adapter is supported on the following application platforms and databases:

Operating systems:

- AIX 4.2.1, AIX 4.3, AIX 5.1, AIX 5.2
- Solaris 7.0, Solaris 8.0
- HP UX 11.0, HP UX 11i
- Windows 2000

Databases:

- Oracle RDBMS (8.1.7)
- Progress 9.1D+ Enterprise DB
- Progress 9.1D+ 4GL

Third-party software:

- QAD MFG/PRO eB2

You must install the QXtend and the Q/LinQ MQSeries MOM products from QAD, but not necessarily on the same machine as the adapter.

For event generation, the QAD product DataSync is also needed. It is installed on top of Q/LinQ, but is sold as a seperate product.

Adapter dependencies

The adapter for QAD MFG/PRO has the client library/API dependencies shown in Table 1.

Table 1. Adapter dependencies

Library/API	Version	Operating systems
IBM JRE	1.3.1.6	All supported OSs
Java libraries for MQSeries and JMS	5.2+	All supported OSs
Xerces	1.4.3	All supported OSs
dom4j	1.4	All supported OSs
JNDI	1.2.1	All supported OSs
Apache SOAP	2.3.1	All supported OSs
WSDL4J	1.0	All supported OSs
IBM JSSE	1.0.2	All supported OSs

Note: The adapter does not support Secure Socket Layers (SSL) in WebSphere MQ 5.3. For the WebSphere MQ software version appropriate to adapter framework-integration broker communication, see the Installation Guide for your platform (Windows/Unix).

Locale-dependent data

This adapter is DBCS (double-byte character set)-enabled and is translated. It 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 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 integration 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 configuration properties, see Appendix A, "Standard configuration properties for connectors," on page 49.

Connector architecture

Connectors consist of an application-specific component and the connector framework. The application-specific component contains code tailored to a particular application. The connector framework, whose code is common to all connectors, acts as an intermediary between the integration broker and the application-specific component. 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

The connector is metadata-driven. You specify metadata— information about business object attributes and application-specific processing—in connector configuration properties and in meta-objects that serve as business object definitions. This obviates the need to hard-code instructions in the connector itself. For more on metadata for the connector, see "Connector configuration" on page 15 as well as "Adapter business object structure" on page 27.

The connector uses WebSphere MQ messaging to receive messages and a SOAP/HTTP protocol-based channel to send them. For more information, see "Application-connector communication" on page 4. The sections below describe how the connector processes information.

Connector processing

The connector allows QAD MFG/PRO eB2 applications to exchange business objects with:

- IBM WebSphere Business Integration collaborations (when configured with WebSphere InterChange Server—ICS)
- message flows (when configured with WebSphere Integration Broker message brokers)

- Enterprise Java Beans (when configured with a WebSphere Application Server).

Figure 1 shows, at a high-level, the connector processing environment.

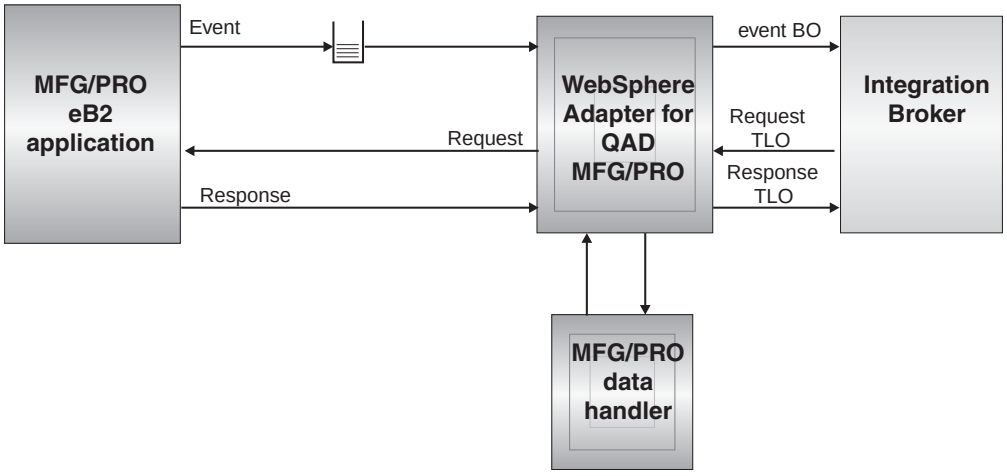

Figure 1. Connector architecture

For event processing, message routing and format conversion are initiated by an event polling technique. When an event is detected, the connector retrieves a message from a QAD MFG/PRO queue. Messages sent to the connector may be in Triplet or QDoc format. For more information on formats, see "Data formats" on page 5. The connector instantiates the QAD MFG/PRO data handler to convert the message to a corresponding business object (BO). The connector then delivers the BO to the integration broker for further processing.

Request processing is synchronous. The connector receives a top-level object (TLO) from an integration broker. A specialized business object, a TLO contains a request, response, and fault business object. Using the QAD MFG/PRO data handler, the connector adds a SOAP header and converts the request BO into a SOAP/XML format. For more information on formats, see "Data formats" on page 5. The connector then delivers the message to the MFG/PRO application via a SOAP/HTTP protocol-based interface. Upon receiving a response message, the connector converts it from SOAP/XML to a business object, which it passes back to the integration broker.

For a more detailed overview of event and request processing as well as of data formats, see "Application-connector communication" below.

Application-connector communication

For event processing, the connector makes use of IBM's WebSphere MQ messaging layer, which is an implementation of the Java Message Service (JMS). The JMS is an open-standard API for accessing enterprise-messaging systems. It is designed to allow business applications to asynchronously send and receive business data and events, and it also makes possible guaranteed event delivery. The connector polls the event (output) queues of MFG/PRO applications to retrieve events.

In the other direction, the connector makes use of QAD's QXtend SOAP/HTTP interface to place requests.

QAD MFG/PRO application interface

As part of its external integration suite (and separate from the adapter for QAD MFG/PRO), QAD provides an interface that enables data transfer between native MFG/PRO applications and the adapter for QAD. For the purposes of the connector, this interface has two channels: one for event processing and one for request processing.

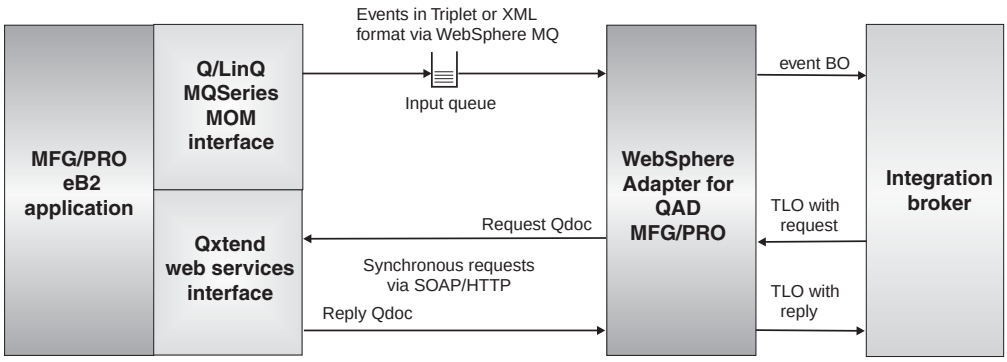

Figure 2. QAD MFG/PRO external interface architecture

Event processing interface

As shown in Figure 2, the event processing QAD interface— whose QAD component name is `QqMomAdapter`— is known as the Q/LinQ MQSeries MOM (message-oriented middleware) interface. The MQSeries component that this interface implements is based on a prior, but compatible, version of the WebSphere MQ messaging layer.

The Q/LinQ MQSeries MOM interface:

- converts native MFG/PRO application data into MQ messages and vice versa
- supports the queues (input and output) by which the connector for QAD MFG/PRO receives events from, and optionally returns confirmation data or error messages to, MFG/PRO applications

The input queue is used for processing events from an MFG/PRO application to a WebSphere-connected application. The connector does not deploy a ReplyTo queue.

Request processing interface

For request processing, the connector makes use of QAD's QXtend integration framework. The QXtend framework is a J2EE applicatio that runs on an application server. It has a web services interface that accepts requests in SOAP-XML 1.2 format.

When TLOs are delivered to the connector from the broker, the connector extracts the request BO and converts it to an XML format with a SOAP envelope. The connector then issues a synchronous request to QXtend using the SOAP-HTTP protocol. Upon receiving a response, the connector converts it back to a business object, puts this response BO into the TLO, and sends the TLO to the broker.

Data formats

The data formats handled by the connector correspond to the native format(s) supported by MFG/PRO applications. Figure 3, shows the data formats for event processing.

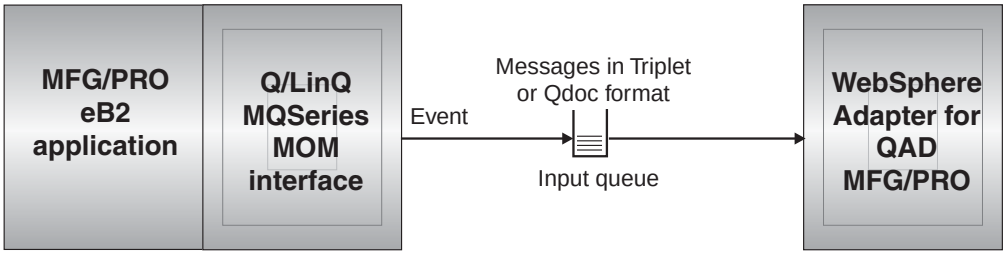

Figure 3. QAD MFG/PRO data formats—event processing

Event processing formats Event messages from the QAD MFG/PRO eB2 application are published either in triplet format or QDoc format (both QAD proprietary). The triplet format consists of a single ASCII text string made up of a serialized set of tagged field values. The QDoc format is XML-based. (For further information on triplet and QDoc formats, see QAD documentation.) The connector converts messages in either format into business objects and then delivers these to the broker.

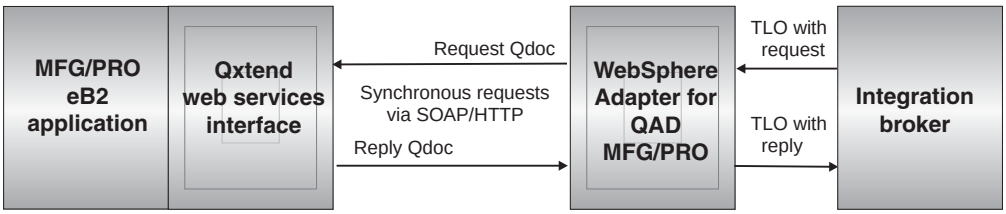

Figure 4. QAD MFG/PRO data formats—request processing

Request processing formats The connector receives a TLO from the broker. The TLO contains request, response, and fault business objects as children. The connector converts the request object into a QDoc message. QDoc is a proprietary XML format. The payload in a QDoc is wrapped into QAD-specific message envelope that conforms to the SOAP 1.2 syntax. The envelope root element contains a header and a body element. The structure of the header is common to all QDoc documents. The QDoc request message is sent to the QXtend web services module via SOAP/HTTP protocol. When the MFG/PRO eB2 application returns a response message, the connector converts it to a response business object. The TLO is then returned to the broker. For further information on QDoc format, see QAD's QDoc Specification. For further information on the TLO and business object structure, see Chapter 3, "Creating or modifying business objects," on page 27.

For information on how the connector's data handler processes messages and business objects, see Chapter 4, "QAD MFG/PRO data handler," on page 35.

Message request

Figure 5 illustrates a message request communication.

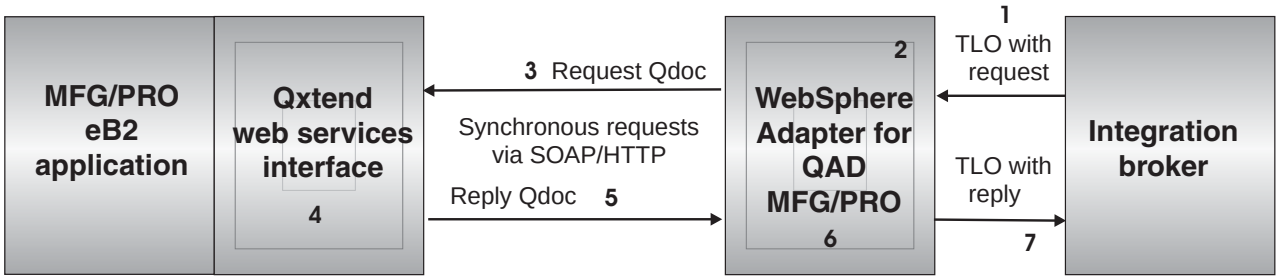

Figure 5. Application-connector communication method: Request processing

1. The integration broker sends a TLO that contains a request business object to the connector.
2. The connector's data handler converts the request BO to a SOAP-XML message in QDoc format.
3. The connector sends the QDoc request message to the QXtend web services interface via SOAP/HTTP.
4. QXtend sends the request to QAD MFG/PRO eB2 application and waits for the response. QXtend generates a response SOAP-XML message in QDoc format.
5. QXtend sends the response message to the connector.
6. The connector's data handler converts the message to a response business object and places it in the TLO originally received from the broker.
7. The connector returns the TLO to the broker.

Event delivery

Figure 6 illustrates the event delivery direction.

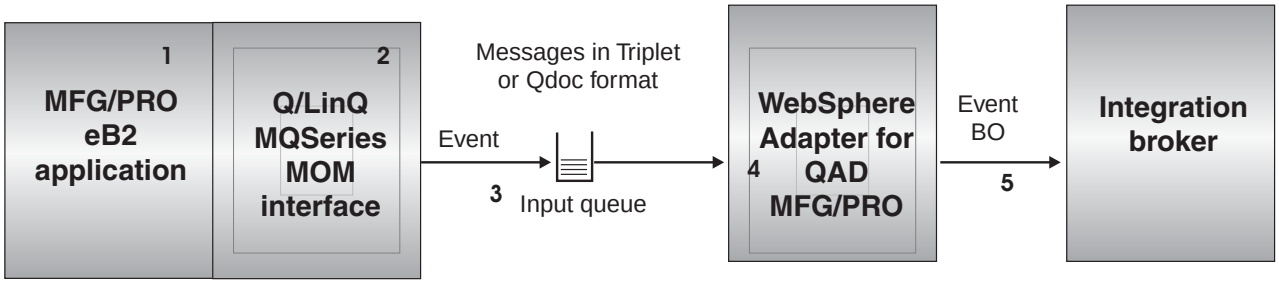

Figure 6. Application-connector communication method: Event delivery

1. Data in QAD MFG/PRO eB2 changes.
2. The Q/LinQ MQSeries MOM interface generates an event message and places it on an MQ queue.
3. The connector's`spollForEvents()` method retrieves the next message from the input queue.
4. The message is staged in the in-progress queue where it remains until processing is complete. The message is passed to the QAD MFG/PRO data handler. The data handler determines whether the message in in triplet or QDoc format. If the message is in triplet format, the data handler invokes the triplet-to-XML mapping engine, which maps the triplet message to a QDoc document. The data handler then converts the QDoc document into a WebSphere business object. Otherwise, if the message is in QDoc format, the data handler skips the triplet-to-XML mapping step and converts the QDoc into a business object.

5. The connector'sgotApplEvents() method delivers the business object to the integration broker for further processing, and the message is removed from the in-progress queue.

Event handling

The connector polls the application's output queue to detect events written to the input queue via a database trigger. You install a database trigger for each MFG/PRO application that you want to configure for connector event processing. You install triggers using the DataSync module supplied by QAD. For further information, see QAD documentation (*External Interface Guide — Data Synchronization*). An event occurs when an MFG/PRO application generates MQ messages and stores them on the input queue.

Retrieval

The connector uses the pollForEvents() method to poll the input queue at regular intervals for messages. When the connector finds a message, it retrieves it from the input queue. The connector passes the message to the configured data handler; the data handler is expected to create and populate a business object and specify a verb. See "Error handling" on page 33 for event failure scenarios.

The connector processes messages by first opening a transactional session to the input queue. This transactional approach allows for a small chance that a business object could be delivered to a collaboration twice due to the connector successfully submitting the business object but failing to commit the transaction in the queue. To avoid this problem, the connector moves all messages to an in-progress queue. There, the message is held until processing is complete. If the connector shuts down unexpectedly during processing, the message remains in the in-progress queue instead of being reinstated to the original input queue.

Note: Transactional sessions with a JMS service provider require that every requested action on a queue be performed and committed before events are removed from the queue. Accordingly, when the connector retrieves a message from the queue, it does not commit to the retrieval until three things occur: 1) The message has been converted to a business object; 2) the business object is delivered to an integration broker by the gotApplEvents() method, and 3) a return value is received.

Recovery

Upon initialization, the connector checks the in-progress queue for messages that have not been completely processed, presumably due to a connector shutdown. The connector configuration property InDoubtEvents allows you to specify one of four options for handling recovery of such messages: fail on startup, reprocess, ignore, or log error.

Fail on startup

With the fail on startup option, if the connector finds messages in the in-progress queue during initialization, it logs an error and immediately shuts down. It is the responsibility of the user or system administrator to examine the message and take appropriate action, either to delete these messages entirely or move them to a different queue.

Reprocess

With the reprocessing option, if the connector finds any messages in the in-progress queue during initialization, it processes these messages first during

subsequent polls. When all messages in the in-progress queue have been processed, the connector begins processing messages from the input queue.

Ignore

With the ignore option, if the connector finds any messages in the in-progress queue during initialization, the connector ignores them, but does not shut down.

Log Error

With the log error option, if the connector finds any messages in the in-progress queue during initialization, it logs an error but does not shut down.

Guaranteed event delivery

The guaranteed-event-delivery feature enables the connector framework to ensure that events are never lost.

Note: Without use of the guaranteed-event-delivery feature, a small window of possible failure exists between the time that the connector publishes an event (when the connector calls the gotApplEvent() method within its pollForEvents() method) and the time it updates the event store by deleting the event record (or perhaps updating it with an "event posted" status). If a failure occurs in this window, the event has been sent but its event record remains in the event store with an "in progress" status. When the connector restarts, it finds this event record still in the event store and sends it, resulting in the event being sent twice.

To configure the connector for guaranteed event delivery, see "Enabling guaranteed event delivery" on page 22..

If the connector framework cannot deliver the business object to the integration broker, then the object is placed on a FaultQueue (instead of UnsubscribedQueue and ErrorQueue) and generates a status indicator and a description of the problem. FaultQueue messages are written in MQRFH2 format.

Business object creation

You create or modify business objects using an XML object discovery agent (ODA) The XML ODA automates the process of building business objects. For further information see Chapter 3, "Creating or modifying business objects," on page 27.

Business object requests

Business object requests are processed when an integration broker sends a business object to the `doVerbFor()` method. Using the configured data handler, the connector converts the business object to a QAD MFG/PRO QDoc message in SOAP/XML format.

Verb processing

The connector processes business objects based on the verb for each business object. The connector uses business object handlers and the `doForVerb()` method to process the business objects that the connector supports. The connector supports the following business object verbs:

- Create
- Update

- Delete

Common installation and configuration tasks

This section provides an overview of some of the configuration and startup tasks that most developers will need to perform.

Install the integration broker

For further information see “Broker compatibility” on page 1 and then consult the installation guide for your integration broker.

Install the WebSphere Business Integration Adapter Framework

For further information see “Broker compatibility” on page 1.

Install the WebSphere Business Integration data handler for XML

For further information, see the *Data Handler Guide*.

Install QAD products

You must install the QXtend and the Q/LinQ MQSeries MOM products from QAD. These do not need to be on the same machine as the adapter.

Install the adapter

See Chapter 2, “Installing and configuring the connector,” on page 13 for a description of what and where you must install.

Define metadata

You must define metadata for:

- the triplet-to-XML mapping files, a sample of which is provided; for further information on the sample see “Installed file structure” on page 14; for further information on triplet-to-XML mapping, see Chapter 4, “QAD MFG/PRO data handler,” on page 35.
- the following meta-objects:
  - the outbound mapping meta-object; for further information see “Configuring the QAD MFG/PRO data handler” on page 36
  - the data handler meta-object; for further information, see “Configuring the QAD MFG/PRO data handler” on page 36

Configure the connector

This sections below list the tasks you must perform to configure an installed connector for QAD MFG/PRO:

Connectors have two types of configuration properties: standard configuration properties and connector-specific configuration properties. Some of these properties have default values that you do not need to change. You may need to set the values of some of these properties before running the connector.

You use Connector Configurator to configure connector properties:

- For a description of Connector Configurator and step-by-step procedures, see Appendix B, “Connector Configurator,” on page 65.
- For a description of standard connector properties, see Appendix A, “Standard configuration properties for connectors,” on page 49.
- For a description of connector-specific properties, see Chapter 2, “Installing and configuring the connector,” on page 13.

You must configure the connector to operate with one of the following integration brokers:

- InterChange Server integration broker— for an overview, see the *Technical Introduction to IBM WebSphere InterChange Server*; to configure the adapter to operate with ICS, see the *Implementation Guide for WebSphere InterChange Server*
- the WebSphere MQ Integrator broker, which is described in the *Implementation Guide for WebSphere MQ Integrator Broker*
- the WebSphere Application Server (WAS), which is described in the *Adapter Implementation Guide for WebSphere Application Server*

When you configure connector properties for the adapter for QAD MFG/PRO, make sure that:

- The value specified for connector property HostName matches that of the host of your WebSphere MQ server.
- The value specified for connector property Port matches that of the port for the listener of your queue manager.
- The value specified for connector property Channel matches the server connection channel for your queue manager.
- The queue URI’s for connector properties InputQueue, InProgressQueue, and ErrorQueue are valid and actually exist, and the value of InputQueue is set to the MQ queue where the he Q/LinQ MQSeries MOM (message-oriented middleware) interface delivers events.

To configure the URL of the Qxtend web service:

- Open the BIA_Protocol_ConfigMO in Business Object Designer and set its Destination attribute to the URL where your Qxtend web service is running. For further information, see “Generating TLOs for request processing: step-by-step” on page 31.

To configure the data handler:

- Specify the data handler meta-object that defines the configuration in the connector-specific property DataHandlerMetaObjectName. For more information, see “Connector-specific properties” on page 15..

For more information on data handler processing, see Chapter 4, “QAD MFG/PRO data handler,” on page 35.

Install a database trigger

Install a database trigger for each MFG/PRO application that you want to configure for connector event processing. You install triggers using the DataSync module supplied by QAD. For further information, see QAD documentation (*External Interface Guide — Data Synchronization*).

Download QDoc schemas

You use QDoc schemas to create business objects.For information and procedures on downloading QDoc schemas from QAD MFG/PRO, see “Generating business

object definitions for event processing: step-by-step" on page 29 or "Generating TLOs for request processing: step-by-step" on page 31. You cannot start or run the connector without first downloading and installing QDoc schemas. For further information, see "Installing the adapter and related files" on page 13.

Create business object definitions

You use an XML object discovery agent (ODA) to create business objects. The six-step automated process is described in Chapter 3, "Creating or modifying business objects," on page 27. Then you must add these definitions, along with the meta-objects, to a list of those supported by the connector. For further information, see "Specifying supported business object definitions" on page 74.

Chapter 2. Installing and configuring the connector

This chapter describes how to install and configure the connector and how to configure the message flows to work with the connector. It contains the following sections:

For information on the adapter environment, including broker compatibility, platform support, and dependencies, see "Adapter environment" on page 1.

Overview of installation tasks

To install the connector for QAD MFG/PRO, you must perform the following tasks:

- **Install the integration broker** This task, which includes installing the WebSphere business integration system and starting the integration broker, is described in the installation documentation for your broker and operating system.
- **Install the XML data handler** For further information, see the *Data Handler Guide*.
- **Install the adapter framework** For further information, see the *Installation Guide for WebSphere Business Integration Adapters*.
- **Install the adapter and related files** This task includes installing files for the adapter from the software package onto your system. See "Installing the adapter and related files."

Installing the 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

Note: To start and run the connector as well as to generate business objects, you must download the QDoc schemas you need from a QAD website or obtain QDoc schema files from the QXtend product. Then copy all .xsd files from *<QXtend installation directory>*/webapps/qxtendserver/WEB-INF/schemas/eB2 to *ProductDir*/dependencies/QDocSchemas For step-by-step downloading procedures, see "Generating business object definitions for event processing: step-by-step" on page 29 or "Generating TLOs for request processing: step-by-step" on page 31.

© Copyright IBM Corp. 2004

Installed file structure

The sections below show the installed file structure for Windows and UNIX.

Windows connector file structure

The Installer copies the standard files associated with the connector into your system.

The utility installs the connector into the *ProductDir*\connectors\QAD directory, and adds a shortcut for the connector to the Start menu.

The table below describes the Windows file structure used by the connector, and shows the files that are automatically installed when you choose to install the connector through Installer.

Subdirectory of *ProductDir*	Description
connectors\QAD\BIA_QAD.jar	Contains classes used by the QAD MFG/PRO connector
connectors\QAD\start_mfgpro.bat	The startup script for the connector (NT/2000)
connectors\messages\BIA_QADConnector.txt	Message file for the connector
repository\QAD\BIA_CN_QAD_TEMPLATE	Repository definition for the connector
connectors\QAD\dependencies\BIA_HeaderMetaData.txt	Metadata for mapping of QDoc header to CIM/triplet message header
connectors\QAD\dependencies\BIA_qdocTemplate.xml	Template used to build a QDoc XML outbound message
connectors\QAD\dependencies\BIA_soapEnvelopeTemplate.xml	Template used to build a QDoc XML inbound message
connectors\QAD\dependencies\BIA_TripletMappingMDOutbound.xml	Metadata for triplet-XML conversions
connectors\QAD\dependencies\dom4j.jar	dom4j library classes
connectors\QAD\dependencies\mail.jar	Classes for web services interaction
connectors\QAD\samples	Sample business object definitions and templates

Note: All product pathnames are relative to the directory where the product is installed on your system.

UNIX connector file structure

The Installer copies the standard files associated with the connector into your system.

The utility installs the connector into the *ProductDir*/connectors/QAD directory.

The table below describes the UNIX file structure used by the connector, and shows the files that are automatically installed when you choose to install the connector through Installer.

Subdirectory of *ProductDir*	Description
connectors/QAD/BIA_QAD.jar	Contains classes used by the QAD MFG/PRO connector
connectors/QAD/start_QAD.sh	The startup script for the connector (AIX, Solaris, HP UX)
connectors/messages/BIA_QADConnector.txt	Message file for the connector
repository/QAD/BIA_CN_QAD_TEMPLATE	Repository definition for the connector
connectors/QAD/dependencies/BIA_HeaderMetaData.txt	Metadata for mapping of QDoc header to triplet message header
connectors/QAD/dependencies/BIA_qdocTemplate.xml	Template used to build a QDoc XML outbound message

Subdirectory of *ProductDir*	Description
connectors/QAD/dependencies/BIA_soapEnvelopeTemplate.xml	Template used to build a QDoc XML inbound message
connectors/QAD/dependencies/BIA_TripletMappingMDOutbound.xml	Metadata for triplet-XML conversions
connectors/QAD/dependencies/dom4j.jar	dom4j library classes
connectors/QAD/dependencies/mail.jar	Classes for web services interaction
connectors/QAD/samples	Sample business object definitions and templates

Note: All product pathnames are relative to the directory where the product is installed on your system.

Connector configuration

Connectors have two types of configuration properties: standard configuration properties and adapter-specific configuration properties. You must set the values of these properties before running the adapter.

You use Connector Configurator to configure connector properties:

- For a description of Connector Configurator and step-by-step procedures, see Appendix B, "Connector Configurator," on page 65.
- For a description of standard connector properties, see "Standard connector properties" and then Appendix A, "Standard configuration properties for connectors," on page 49.
- For a description of connector-specific properties, see "Connector-specific properties."

An adapter obtains its configuration values at startup. During a runtime session, you may want to change the values of one or more connector properties. Changes to some connector configuration properties, such as AgentTraceLevel, take effect immediately. Changes to other connector properties require component restart or system restart after a change. To determine whether a property is dynamic (taking effect immediately) or static (requiring either connector component restart or system restart), refer to the Update Method column in the Connector Properties window of the System Manager.

Standard connector properties

Standard configuration properties provide information that all adapters use. See Appendix A, "Standard configuration properties for connectors," on page 49 for documentation of these properties.

You must provide a value for the ApplicationName configuration property before running the connector.

Connector-specific properties

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

The table below lists the connector-specific configuration properties for the adapter. See the sections that follow for explanations of the properties.

Name	Possible values	Default value	Required
ApplicationPassword	*Login password for WebSphere MQ*		No
ApplicationUserName	*Login user ID for WebSphere MQ*		No
CCSID	*Character set for queue manager connection*	`null`	No
Channel	*MQ server channel through which connector communicates with WebSphere MQ*		Yes
DataEncoding	*The encoding that is used to transform the bytes of a message from the MO queue to a java String: `US-ASCII`, `ISO-8859-1`, `UTF-8`, `UTF-16`*	UTF-8	No
DataHandlerMetaObjectName	*Data handler meta-object*	`BIA_DataHandlerConfigMO`	Yes
ErrorQueue	*Queue for unprocessed messages*		No
HostName	*WebSphere MQ server*		Yes
InDoubtEvents	`FailOnStartup, Reprocess, Ignore, LogError`	`Reprocess`	No
InputQueue	*Queue where MFG/PRO puts events*		Yes
InProgressQueue	*In-progress event queue*		No
PollQuantity	*Number of messages to retrieve from each queue specified in the `InputQueue` property*	1	No
Port	*Port established for the WebSphere MQ listener*		Yes
ProtocolHandlerFramework	*This is a hierarchical property and has no value*	None	Yes
+ProtocolHandlers	*This is a hierarchical property and has no value*		Yes
++SOAPHTTPHTTPSHandler	*This is a hierarchical property. For information on its sub-properties, see "SOAPHTTPHTTPSHandler" on page 18.*		Yes
ProxyServer	*This is a hierarchical property and has no value*		No
+HttpProxyHost	*Host name for the HTTP proxy server*		No
+HttpProxyPort	*Port number for the HTTP proxy server*	`80`	No
+HttpNonProxyHosts	*HTTP host(s) requiring direct connection*		No
+HttpsProxyHost	*Host name for the HTTPS proxy server*		No
+HttpsProxyPort	*Port number for the HTTPS proxy server*	`443`	No
+HttpsNonProxyHosts	*HTTPS host(s) requiring direct connection*		No
+SocksProxyHost	*Socks proxy server name*		No
+SocksProxyPort	*Socks proxy server port*		No
+HttpProxyUsername	*Http proxy server username*		No
+HttpProxyPassword	*Http proxy server password*		No
+HttpsProxyUsername	*Https proxy server username*		No
+HttpsProxyPassword	*Https proxy server password*		No
UnsubscribedQueue	*Queue to which unsubscribed messages are sent*		No
UseDefaults	true or false	false	

ApplicationPassword
Password used with `UserID` to log in to WebSphere MQ.

Default = None.

If the `ApplicationPassword` is left blank or removed, the connector uses the default password provided by WebSphere MQ.*

ApplicationUserName
User ID used with `Password` to log in to WebSphere MQ.

Default = None.

If the `ApplicationUserName` is left blank or removed, the connector uses the default user ID provided by WebSphere MQ.*

CCSID
The character set for the queue manager connection. The value of this property should match that of the CCSID property in the queue URI.

Default = null.

Channel
MQ server connector channel through which the connector communicates with WebSphere MQ.

Default = none.

If the `Channel` is left blank or removed, the connector uses the default server channel provided by WebSphere MQ.

DataEncoding
The encoding that is used to transform the bytes of a message from the MO queue to a java String. Possible values are: `US-ASCII`, `ISO-8859-1`, `UTF-8`, `UTF-16`.

Default = `UTF-8`

DataHandlerMetaObjectName
The name of the top-level meta-object passed to data handler to provide configuration information. For further information, see “Configuring the QAD MFG/PRO data handler” on page 36.

Default = `BIA_DataHandlerConfigMO`

ErrorQueue
Queue to which messages that could not be processed are sent.

Default = none

HostName
The name of the server hosting WebSphere MQ.

Default = none.

InDoubtEvents
Specifies how to handle in-progress events that are not fully processed due to unexpected connector shutdown. Choose one of four actions to take if events are found in the in-progress queue during initialization:

- `FailOnStartup.` Log an error and immediately shut down.
- `Reprocess.` Process the remaining events first, then process messages in the input queue.
- `Ignore.` Disregard any messages in the in-progress queue.
- `LogError.` Log an error but do not shut down

Default = `Reprocess`.

InputQueue
The queue where MFG/PRO puts events. Message queues that will be polled by the connector for new messages. There is only1 InputQueue.

Default = none

InProgressQueue
Message queue where messages are held during processing. You can configure the connector to operate without this queue by using System Manager to remove the default InProgressQueue name from the connector-specific properties. Doing so prompts a warning at startup that event delivery may be compromised if the connector is shut down while are events pending.

Default= none

PollQuantity
The number of messages to retrieve from each queue specified in the `InputQueue` property during a `pollForEvents` scan.

Default =1

Port
Port established for the WebSphere MQ listener.

Default = None

ProtocolHandlerFramework
The Protocol Handler Framework uses this property to load and configure its protocol handlers. This is a hierarchical property and has no value.

Default = none.

ProtocolHandlers
This hierarchical property has no value. Its first-level children represent discrete protocol handlers.

Default = none.

SOAPHTTPHTTPSHandler
The name of a SOAP/HTTP-HTTPS protocol handler. Note that this is a hierarchical property. Unlike listeners, protocol handlers may not be duplicated, and there can be only one handler for each protocol. Table 2 below shows the

sub-properties for the SOAP/HTTP-HTTPS protocol handler. The + character indicates the entry's position in the property hierarchy.

Table 2. SOAP/HTTP-HTTPS protocol handler configuration properties

Name	Possible values	Default value	Required
++SOAPHTTPHTTPSHandler	This is a hierarchical property and has no value.		Yes
+++Protocol	*The kind of protocol the handler is implementing. For SOAP/HTTP and SOAP/HTTPS, the value is* `soap/http`. **Note:** If you do not specify a value for this property, the connector will not initialize this protocol handler.		Yes
+++HTTPReadTimeout	*A SOAP/HTTP-specific property that specifies the timeout interval (in milliseconds) while reading from the remote host (web service).If this property is not specified or if set to 0, the SOAP/HTTP protocol handler blocks indefinitely while reading from the remote host.*	`0`	No

ProxyServer
Configure the values under this property when the network uses a proxy server between the machine on which the adapter is running and the Tomcat QXtend server.This is a hierarchical property and has no value. The values specified under this property are used by the SOAP/HTTP/HTTPS protocol handler.

HttpProxyHost
The host name for the HTTP proxy server. Specify this property if the network uses a proxy server for HTTP protocol.

Default = none

HttpProxyPort
The port number that the connector uses to connect to the HTTP proxy server.

Default = 80

HttpNonProxyHosts
The value of this property gives one or more hosts (for HTTP) that must be connected not through the proxy server but directly. The value can be a list of hosts, each separated by a `"|"`.

Default = none

HttpsProxyHost
The host name for the HTTPS proxy server.

Default = none

HttpsProxyPort
The port number that the connector uses to connect to the HTTPS proxy server.

Default = 443

HttpsNonProxyHosts
The value of this property gives one or more hosts (for HTTPS) that must be connected not through the proxy server but directly. The value can be a list of hosts, each separated by a "|".

Default = none

SocksProxyHost
The host name for the Socks Proxy server. Specify this property when the network uses a socks proxy.

Note: The underlying JDK must support socks.

Default = none

SocksProxyPort
The port number to connect to the Socks Proxy server. Specify this property when the network uses a socks proxy.

Default = none

HttpProxyUsername
The username for the HTTP proxy server. If the destination for the web service request is an HTTP URL and you specify ProxyServer ->HttpProxyUsername, the SOAP HTTP/HTTPS protocol handler creates a Proxy-Authorization header when authenticating with the proxy. The handler uses the CONNECT method for authentication.

The proxy-authentication header is base64 encoded and has the following structure:
Proxy-Authorization: Basic
Base64EncodedString

The handler concatenates the username and the password property values, separated by a colon (:), to create the base64 encoded string.

Default = none

HttpProxyPassword
The password for the HTTP proxy server. For more on how this value is used, see "HttpProxyUsername."

Default = none

HttpsProxyUsername
The username for the HTTPS proxy server. If the destination for the web service request is an HTTPS URL and you specify ProxyServer ->HttpsProxyUsername, the SOAP HTTP/HTTPS protocol handler creates a Proxy-Authorization header for authentication with the proxy. The handler concatenates the HttpsProxyUsername and HttpsProxyPassword configuration property values, separated by colon (:), to create the base64 encoded string.

Default = none

HttpsProxyPassword

The password for the HTTPS proxy server. For more on how this value is used, see "HttpsProxyUsername" on page 20.

Default = none

UnsubscribedQueue

Queue to which messages that are not subscribed are sent.

Default = none

Note: *Always check the values WebSphere MQ provides since they may be incorrect or unknown. If so, please implicitly specify values.

UseDefaults

For request business objects, if UseDefaults is set to `true`, the connector checks whether a valid value or a default value is provided for each isRequired business object attribute. If a value is provided, the request operation succeeds. If the parameter is set to `false`, the connector checks only for a valid value and causes the request operation to fail if it is not provided. The default is `false`.

Configuring for event processing

When you configure connector properties for the adapter for QAD MFG/PRO, make sure that:

- The value specified for connector property HostName matches that of the host of your WebSphere MQ server.
- The value specified for connector property Port matches that of the port for the listener of your queue manager.
- The value specified for connector property Channel matches the server connection channel for your queue manager.
- The queue URI's for connector properties InputQueue, InProgressQueue, and ErrorQueue are valid and actually exist, and the value of InputQueue is set to the MQ queue where the he Q/LinQ MQSeries MOM (message-oriented middleware) interface delivers events.

Configuring for request processing

For request processing, you must specify the URL for the QXtend web service. You can do this by specifying that URL as the value of the Destination attribute in the ProtocolConfigMO of the request BO definition.

For further information on setting the Destination attribute in ProtocolConfigMOs, see "Overview of generating TLOs for request processing" on page 30.

Configuring the data handler

To configure the data handler:

- Specify the data handler meta-object that defines the configuration in the connector-specific property DataHandlerMetaObjectName. For more information, see "DataHandlerMetaObjectName" on page 17.

For more information on data handler processing, see Chapter 4, "QAD MFG/PRO data handler," on page 35.

Enabling guaranteed event delivery

You can configure the guaranteed-event-delivery feature for the adapter for QAD. You accomplish this by using duplicate event elimination to ensure that duplicate events do not occur. This section provides the following information about use of the guaranteed-event-delivery feature:

- "Enabling the feature for connectors"
- "Effect on event polling"

Enabling the feature for connectors

To enable the guaranteed-event-delivery feature, you must set the connector configuration properties to values shown in Table 3.

Table 3. Guaranteed-event-delivery connector properties for a connector with a non-JMS event store

Connector property	Value
`DeliveryTransport`	`JMS`
`DuplicateEventElimination`	`true`
`MonitorQueue`	Name of the JMS monitor queue, in which the connector framework stores the ObjectEventId of processed business objects

If you configure a connector to use guaranteed event delivery, you must set the connector properties as described in Table 3. To set these connector configuration properties, use the Connector Configurator tool. It displays these connector properties on its Standard Properties tab. For information on Connector Configurator, see Appendix B, "Connector Configurator," on page 65.

Effect on event polling

If a connector uses guaranteed event delivery by setting `DuplicateEventElimination` to `true`, it behaves slightly differently from a connector that does not use this feature. To provide the duplicate event elimination, the connector framework uses a JMS monitor queue to track a business object. The name of the JMS monitor queue is obtained from the `MonitorQueue` connector configuration property.

After the connector framework receives the business object from the application-specific component (through a call to `gotApplEvent()` in the `pollForEvents()` method), it must determine if the current business object (received from `gotApplEvents()`) represents a duplicate event. To make this determination, the connector framework retrieves the business object from the JMS monitor queue and compares its ObjectEventId with the ObjectEventId of the current business object:

- If these two ObjectEventIds are the same, the current business object represents a duplicate event. In this case, the connector framework ignores the event that the current business object represents; it does *not* send this event to the integration broker.
- If these ObjectEventIds are *not* the same, the business object does *not* represent a duplicate event. In this case, the connector framework copies the current business object to the JMS monitor queue. Control returns to the connector's `pollForEvents()` method, after the call to the `gotApplEvent()` method.

For the connector to support duplicate event elimination, you must make sure that the connector's `pollForEvents()` method includes the following steps:

- When you create a business object from an event record retrieved from the non-JMS event store, save the event record's unique event identifier as the business object's ObjectEventId attribute.

  The application generates this event identifier to uniquely identify the event record in the event store. If the connector goes down after the event has been sent to the integration broker but before this event record's status can be changed, this event record remains in the event store with an In-Progress status. When the connector comes back up, it should recover any In-Progress events. When the connector resumes polling, it generates a business object for the event record that still remains in the event store. However, because both the business object that was already sent and the new one have the same event record as their ObjectEventIds, the connector framework can recognize the new business object as a duplicate and not send it to the integration broker.
- During connector recovery, make sure that you process In-Progress events *before* the connector begins polling for new events.

  Unless the connector changes any In-Progress events to Ready-for-Poll status when it starts up, the polling method does not pick up the event record for reprocessing.

Creating multiple instances of the connector

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 23.
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 4 shows.

Table 4. 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* [-c*configFile*]
  - 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*.

Initializing the connector

During initialization, the connector:

- Retrieves its configuration properties
- Checks the accessibility of the configured input and output queues
- Loads the data handler mapping metadata

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. Creating or modifying business objects

The connector comes with sample business objects only. The systems integrator, consultant, or customer must build business objects. For further information on building business objects, see "Overview of generating business object definitions for event processing" on page 28 below.

The connector is a metadata-driven connector. In WebSphere business integration system business objects, metadata is data about the application, which is stored in a business object definition and which helps the connector interact with an application. A metadata-driven connector handles each business object that it supports based on metadata encoded in the business object definition rather than on instructions hard-coded in the connector.

Business object metadata includes the structure of a business object, the settings of its attribute properties, and the content of its application-specific information. Because the connector is metadata-driven, it can handle new or modified business objects without requiring modifications to the connector code. However, the connector's configured data handler makes assumptions about the structure of its business objects, object cardinality, the format of the application-specific information, and the database representation of the business object. Therefore, when you create or modify a business object for QAD MFG/PRO, your modifications must conform to the rules the connector is designed to follow, or the connector cannot process new or modified business objects correctly.

This chapter describes how the connector processes business objects and describes the assumptions the connector makes. You can use this information as a guide to implementing new business objects.

Adapter business object structure

After installing the adapter, you must create business objects. Business object definitions are based on QAD-defined QDoc XML schemas. QDoc is a proprietary XML format. There are two business object structures: one that corresponds to event processing and another that corresponds to request processing.

Event processing business object structure

The QAD MFG/PRO business object mirrors the structure of a QDoc. For event processing, the definition contains two first-level child business objects, one for the QDoc header and one for the QDoc body. When you installed the adapter, you also installed a header business object definition (that is common to all QDocs) and a template for creating a business object definition that contains a placeholder for the body. The body business object is message-type specific. For guidelines on creating business object definitions for event processing, see "Overview of generating business object definitions for event processing" on page 28.

© Copyright IBM Corp. 2004

Business objects take the name of the corresponding QDoc schema with the addition of the prefix`BIA_` and the suffix `BO`. For example: `BIA_maintainSupplierBO`.

Note: The adapter retrieves messages from a queue and attempts to populate a business object with the message contents. Strictly speaking, the connector neither controls nor influences business object structure. Those are functions of meta-object definitions as well as the connector's data handler requirements. In fact, there is no business-object level application information. Rather, the connector's main role when retrieving and passing business objects is to monitor the message-to-business-object (and vice versa) process for errors.

Request processing business object structure

For request processing, which is synchronous, you create top level objects (TLOs). TLOs are specialized objects that contain child request, response, and fault business objects. The request child BO corresponds to, and is generated from, a QDoc XML schema. It has a header and a body part. The data handler converts the request child business object into a QDoc SOAP/XML message. The connector's protocol handler then sends the request message to the QXtend web service. When it receives a the reply QDoc, it populates the response child BO of the TLO with the reply data. The response child BO corresponds to the QDoc XML response schema. If a fault message is returned, the connector populates the fault child BO. The BIA_FaultBO has been created based on the SOAP 1.2 schema (http://www.w3.org/2002/12/soap-envelope), using the XML ODA.

Business object definitions use the name of the corresponding QDoc and have a `QDoc_TLO` suffix.

For guidelines on creating business object definitions for request processing, see "Overview of generating TLOs for request processing" on page 30.

Overview of generating business object definitions for event processing

You generate business objects for event processing from QDoc schemas and WebSphere business object templates. A QDoc schema is an XML schema that describes message content. As shown in Figure 1, the process is an automated one in which you use the XML Object Discovery Agent (ODA) and Business Object Designer to create business objects (BO) that conform to QDoc schemas.

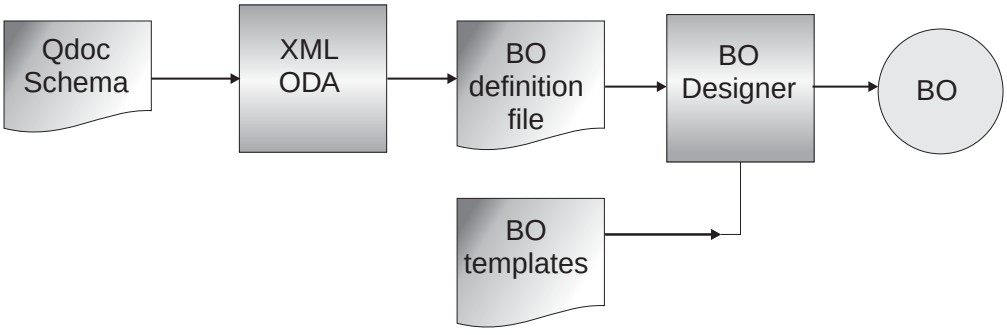

Figure 7. Generating an event processing business object from a QDoc schema

To run the XML ODA, see the *Data Handler Guide*. You use QDoc schemas (rather than DTDs) as input to the XML ODA.

The XML ODA can generate individual definitions for the header and body sections of the message only, but not for the entire QDoc. The XML ODA generates the header business object definition that is common to all messages. This definition, BIA_QDocHeaderBO, is provided with the adapter installation. In addition, a QDoc template business object definition, BIA_TemplateEnvelopeBO, is provided. You use this template to help construct a QDoc business object definition. The template includes two attributes, one for the header and one for the body of a QDoc. Figure 8 shows BIA_TemplateEnvelopeBO in Business Object Designer.

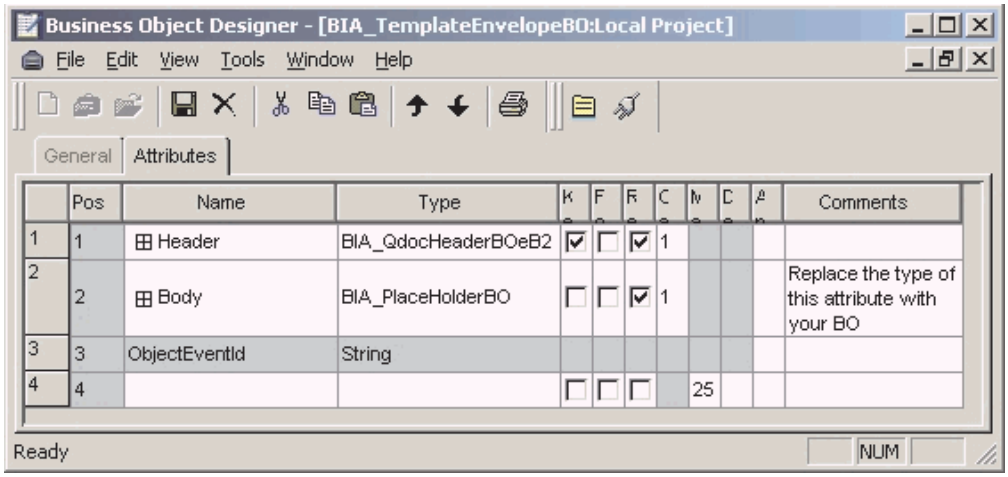

Figure 8. BIA_TemplateEnvelopeBO

Generating business object definitions for event processing: step-by-step

To generate a business object definition for a QDoc for event processing:

1. Download the QDoc schemas you need from a QAD website or obtain QDoc schema files from the QXtend product by copying all .xsd files from *<QXtend installation directory>*/webapps/qxtendserver/WEB-INF/schemas/eB2 to *ProductDir*/dependencies/QDocSchemas
2. Use the XML ODA to generate the definition of the QDoc body from the QDoc schema; for example for the maintainSupplier QDoc use the maintainSupplier.xsd as input for the ODA; then save the definition under the same name (maintainSupplier).
3. Open BIA_TemplateEnvelopeBO in Business Object Designer.

   Note: You can find this and other BO templates in the installed directory *ProductDir*/connectors/QAD/samples.
4. Change the body type to the actual business object definition for the body that was generated previously, (for example, maintainCustomer, which is shown in Figure 9.

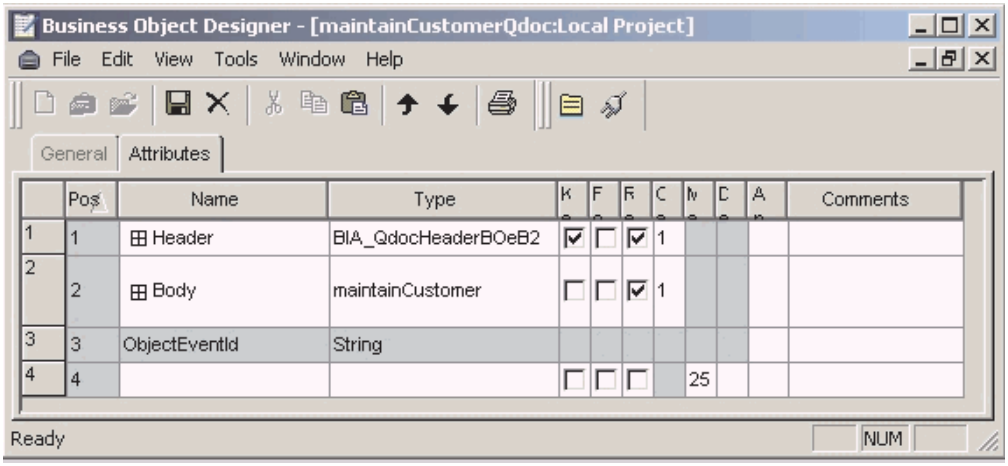

Figure 9. BO envelope with maintainCustomer body

5. Save the new business object definition as *QDoc_name*`BO` where *QDoc_name* is the name of the QDoc with the first letter capitalized; in the example above, the name of the new BO would be `MaintainCustomerBO`.

Overview of generating TLOs for request processing

You use QDoc schemas, business object templates, the XML ODA, and Business Object Designer to create TLOs for request processing. TLOs contain request, response and fault business objects.

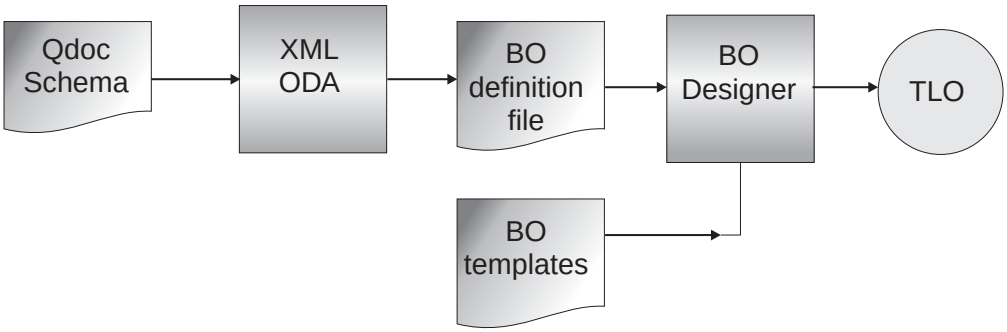

Figure 10. Generating a request processing TLO from a QDoc schema

Figure 11 shows the structure of a TLO used for request processing. The request, response, and fault attributes correspond to business objects.

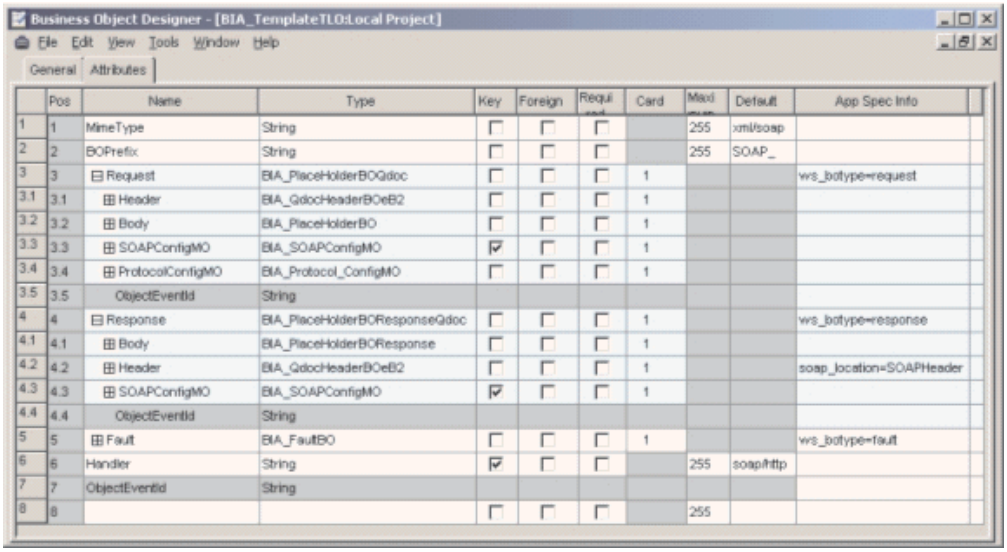

Figure 11. BIA_TemplateTLO

The ProtocolConfigMO in the request BO definition contains a Destination attribute that specifies the URL of the QXtend web service. You set this attribute's value to the QXtend web service URL that you are targeting. Or you can specify a default URL for a QXtend web service in the connector-specific property QXtendDefaultURL. If no URL is defined in the Destination attribute of the ProtocolConfigMO of the request business object, the value of the QXtendDefaultURL property is used.

A template TLO, `BIA_TemplateTLO`, is installed with the adapter. To run the XML ODA, see the *Data Handler Guide*. You use QDoc schemas (rather than DTDs) as input to the XML ODA.

To generate TLOs for request processing, see "Generating TLOs for request processing: step-by-step."

Generating TLOs for request processing: step-by-step

To build the actual TLO, perform the following steps:

1. Download the QDoc schemas you need from a QAD website or obtain QDoc schema files from the QXtend product by copying all .xsd files from *<QXtend installation directory>*`/webapps/qxtendserver/WEB-INF/schemas/eB2` to *ProductDir*`/dependencies/QDocSchemas`
2. Use the XML ODA to generate the definition of the QDoc body from the QDoc schema; for example for the `maintainSupplier` QDoc use the `maintainSupplier.xsd` as input for the ODA; then save the definition under the same name (`maintainSupplier`).
3. Open `BIA_PlaceHolderBOQDoc` in Business Object Designer.

   Note: You can find this and other BO templates in the installed directory *ProductDir*`/connectors/QAD/samples`.
4. Change the Body type to the actual BO definition for the body that was generated previously; `maintainSupplier` in the example.

5. Save the new definition as <*qdoc_name*>RequestQDoc where <*qdoc_name*> is the name of the QDoc; in the example above, the name of the new BO would be maintainSupplierRequestQDoc.
6. Open the ProtocolConfigMO of the request object and specify the URL for the QXtend web service as the value of the Destination attribute. Figure 12 shows the Destination attribute of the ProtocolConfigMO for the Request BO.

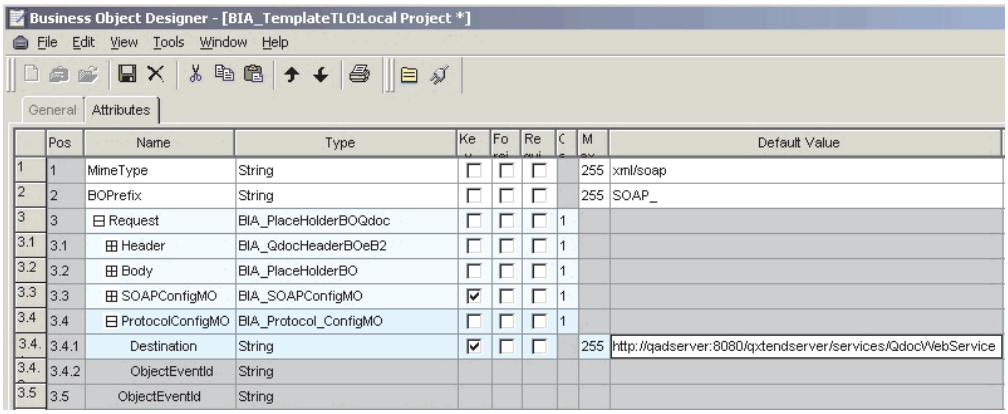

Figure 12. ProtocolConfigMO Destination attribute

7. Use the XML ODA to generate the definition of the response QDoc body from the QDoc schema; for example for the maintainSupplierResponse QDoc use the maintainSupplierResponse.xsd as input for the ODA. Then save the definition under the same name (maintainSupplierResponse).
8. Load the BIA_PlaceHolderBOResponseQDoc in Business Object Designer.

   Note: You can find this and other BO templates in the installed directory *ProductDir*/connectors/QAD/samples.
9. Change the body type to the actual BO definition for the body that was generated previously, maintainSupplierResponse in the example.
10. Save the new definition as <*qdoc_name*>ResponseQDoc where <*qdoc_name*> is the name of the QDoc; in the example above, the name of the new BO would be maintainSupplierResponseQDoc.
11. Open BIA_TemplateTLO in Business Object Designer.

    Note: You can find this and other BO templates in the installed directory *ProductDir*/connectors/QAD/samples.
12. Change the request type from BIA_PlaceHolderBOQDoc to the type you saved previously for the request BO; in the example, the name would be maintainSupplierRequestQDoc.
13. Change the response type from BIA_PlaceHolderBOResponseQDoc to the type you saved previously for the response BO, in the example, the name would be maintainSupplierResponseQDoc.
14. Save the new definition as <*qdoc_name*>QDoc_TLO where <*qdoc_name*> is the name of the QDoc; in the example, the name of the new TLO would be maintainSupplierQDoc_TLO.

Error handling

All error messages generated by the connector are stored in a message file named BIA_QADConnector.txt. (The name of the file is determined by the LogFileName standard connector configuration property.) Each error has an error number followed by the error message:

```
Message number
Message text
```

The connector handles specific errors as described in the following sections.

Event processing

- When unable to access the event queue, the connector logs a fatal error and terminates.
- When the data handler cannot generate a business object, the connector logs an error. For further information, see Chapter 4, "QAD MFG/PRO data handler," on page 35.
- If the business object is not subscribed, the connector logs an error. For further information, see "Configuring the QAD MFG/PRO data handler" on page 36.
- If there is a business object delivery failure, the connector logs an error and leaves the message in the queue.

Application timeout

The error message ABON_APPRESPONSETIMEOUT is returned when:

- The connector cannot establish a connection to the JMS service provider during message retrieval.
- The connector successfully converts a business object to a message but cannot deliver it the outgoing queue due to connection loss.
- The connector issues a message but times out waiting for a response for a business object with conversion property TimeoutFatal equal to True.
- The connector receives a response message with a return code equal to APP_RESPONSE_TIMEOUT or UNABLE_TO_LOGIN.

Connector not active

When the gotApplEvent() method returns a CONNECTOR_NOT_ACTIVE code, the pollForEvents() method returns an APP_RESPONSE_TIMEOUT code and the event remains in the InProgress queue.

Data handler conversion

If the data handler fails to convert a message to a business object, or if a processing error occurs that is specific to the business object (as opposed to the JMS provider), the message is delivered to the queue specified by ErrorQueue. If the ErrorQueue is not defined, messages that cannot be processed due to errors will be discarded.

If the data handler fails to convert a business object to a message, BON_FAIL is returned.

Tracing

Tracing is an optional debugging feature you can turn on to closely follow connector behavior. Trace messages, by default, are written to STDOUT. See the connector configuration properties in Chapter 2, "Installing and configuring the connector," on page 13 for more on configuring trace messages. For more information on tracing, including how to enable and set it, see the *Connector Development Guide*.

What follows is recommended content for connector trace messages.

Level 0 — This level is used for trace messages that identify the connector version.

Level 1 — Use this level for trace messages that provide key information on each business object processed or record each time a polling thread detects a new message in an input queue.

Level 2 — Use this level for trace messages that log each time a business object is posted to an integration broker, each time a request business object is received, or that identify the business object handlers used for each object the connector processes.

Level 3 — Use this level for trace messages that provide information regarding message-to-business-object and business-object-to-message conversions or provide information about the delivery of the message to the output queue.

Level 4 — Use this level for trace messages that identify when the connector enters or exits a function, to identify application-specific information (for example, the values returned by the methods that process the ASI fields in business objects), or to record thread-specific processing (for example, if the connector spawns multiple threads, a trace message can log the creation of each new thread.

Level 5 — Use this level for trace messages that indicate connector initialization, represent statements executed in the application, indicate whenever a message is taken off of or put onto a queue, or record business object dumps.

Chapter 4. QAD MFG/PRO data handler

The QAD MFG/PRO data handler is a data-conversion module whose primary roles are to convert business objects into MFG/PRO formats and vice versa. For an overview of these formats, see "Data formats" on page 5. This chapter describes how the data handler processes messages from one format to another as well as metadata requirements.

Event Processing As shown in Figure 13,, the QAD MFG/PRO data handler has a mapping engine that converts a triplet message to an XML QDoc, which is QAD's proprietary XML format. The connector's XML data handler is then invoked to convert the QDoc to the corresponding business object (BO). If the event message arrives in XML format, then the connector skips the triplet-to-QDoc process and passes the message to the XML data handler for conversion to an event BO. (The business object is generated from QDoc schema during design time; for further information, see "Overview of generating business object definitions for event processing" on page 28.) Applying the values of operation tags in the QDoc, the data handler sets the verb at the business-object level as well as for any child BOs.

Metadata helps guide the format conversions:

- Metadata specified in the QADDataHandlerMO (meta-object) is used to perform the triplet-to-QDoc mapping (for further information on this meta-object, see "Configuring the QAD MFG/PRO data handler" on page 36).
- Metadata stored in a child (OutboundMappingMO) object of the QADDataHandlerMO helps maps the QDoc to a BO.

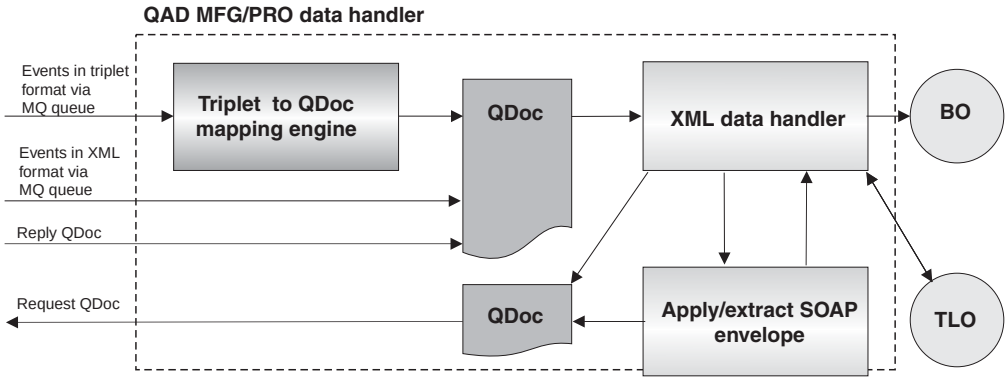

Figure 13. QAD MFG/PRO data handler processing

Request Processing In the other direction, the XML data handler component converts the request business object of a TLO into a QDoc XML message which is then wrapped in a SOAP 1.2 envelope. The request BO must have a header attribute of type `BIA_QdocHeaderBOeB2` and a Body attribute that is request specific. The protocol handler then passes the QDoc to the QXtend web service.

© Copyright IBM Corp. 2004

When it receives a reply QDoc, the connector calls the XML data handler to create a response business object and to extract the SOAP envelope. Like the request BO, the response BO has a header of type `BIA_QdocHeaderBOeB2` and a `Body` attribute. If a fault QDoc message is returned, the connector calls the data handler to convert it to a `BIA_FaultBO`. For Fault messages, the QDoc header is discarded.

Metadata guides the request processing conversions, as discussed in "Configuring the QAD MFG/PRO data handler."

Configuring the QAD MFG/PRO data handler

The QAD MFG/PRO data handler is a pivotal component in the connector for QAD MFG/PRO. The connector calls the data handler to transform business objects into QDoc messages and to transform triplet and QDoc messages into business objects.

The information in data handler meta-objects plays a crucial role in these transformations. You configure this information after you install the product files, but before startup. Unless you are customizing or extending the QAD MFG/PRO data handler, configure the following item only:

- Using Connector Configurator, set the value of the connector-specific property DataHandlerMetaObjectName to `BIA_QADDataHandlerMO`. For information on launching and using Connector Configurator, including a step-by-step procedure, see Appendix B, "Connector Configurator," on page 65.

Figure 14 on page 37 illustrates the data handler meta-object hierarchy, which includes the BIA_QADDataHandlerMO and the OutboundMappingMO objects.

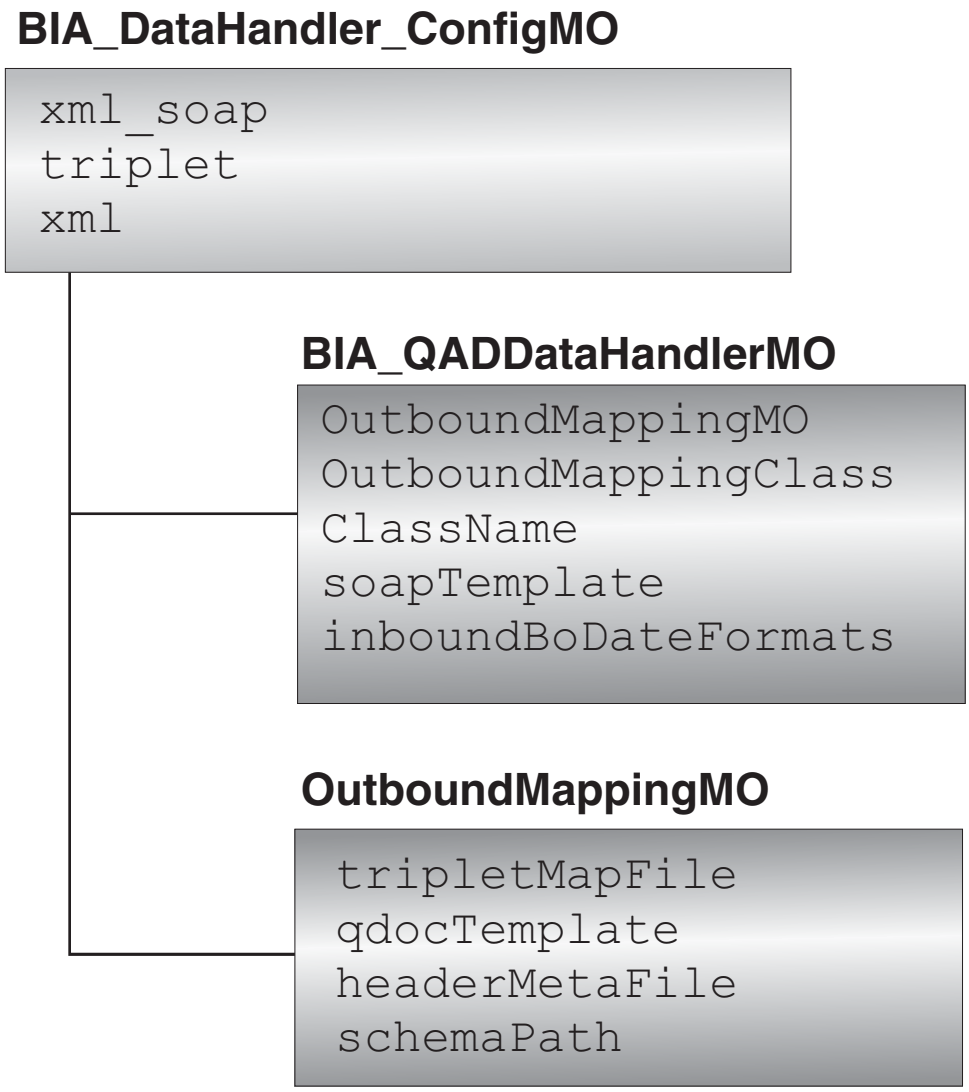

Figure 14. Data handler meta-objects

Table 6 describes the attributes required for the `BIA_DataHandlerConfigMO`. The name of this top-level meta-object is read from the DataHandlerMetaObjectName connector configuration property.

Table 5. Meta-object attributes for BIA_DataHandlerConfigMO

Attribute name	Type	Description
`xml_soap`	`BIA_QADDataHandlerMO`	Meta-object for the QAD MFG/PRO data handler, used with `xml_soap` mime type
`triplet`	`BIA_QADDataHandlerMO`	Meta-object for the QAD MFG/PRO data handler, used with `triplet` mime type
`xml`	BIA_XMLDataHandlerMO	Meta-object for the XML data handler, which is configured as a component of the QAD MFG/PRO data handler.

Table 6 describes the attributes for the QADDataHandlerMO.

Table 6. Meta-object attributes for QADDataHandlerMO

Attribute name	Type	Description
OutboundMappingMO	String	Name of the meta-object for outbound mapping (event processing). If this attribute is set to CxIgnore, no outbound mapping is performed.
OutboundMappingClass	String	Class name for event processing mapping handler. This class is used to perform the mapping from triplet to QDoc XML. If this attribute is set to CxIgnore, no triplet-to-QDoc mapping is performed.
ClassName	String	Data handler class with value of com.ibm.adapters.qad.datahandlers.QADDataHandler. Do not change this value.
soapTemplate	String	Fully qualified pathname of the SOAP envelope template file BIA_SoapEnvelopeTemplate.xml.
inboundBoDateFormats	String	List of patterns used to parse incoming date fields, separated by ";". Default: mm/dd/yyyy;mm-dd-yyyy;yyyy-mm-dd

Table 5 describes the attributes for the OutboundMappingMO (event processing).

Table 7. Meta-object attributes for OutboundMappingMO (event processing)

Attribute name	Type	Description
tripletMapFile	String	Fully qualified pathname for the triplet XML mapping file
qdocTemplate	String	Fully qualified pathname for the QDoc template file
headerMetaFile	String	Fully qualified pathname of the file containing the triplet-to-QDoc header mappings
schemaPath	String	Fully qualified pathname for the root directory containing the QDoc XML schema files.

For a description of the BIA_XMLDataHandlerMO, see the *Data Handler Guide*. Mapping metadata and its role in format conversions is discussed in the appropriate sections below.

QAD MFG/PRO data handler processing

The QAD MFG/PRO data handler performs transformations between QAD MFG/PRO messages and business objects in the following ways:

- **MFG/PRO-message-to-business-object processing**
  - QAD MFG/PRO message-to-business-object data handling occurs during event processing when a QAD MFG/PRO application make calls to a WebSphere business process. The data handler calls a mapping engine that converts the triplet message to a QDoc and then invokes the XML data handler to convert the QDoc to a business object. If the message is a QDoc instead of a triplet, the mapping engine is bypassed and the XML data handler is called directly.

- **Business-object-to-MFG/PRO-message processing** occurs during request processing. The data handler uses the XML data handler to convert the request business object of a TLO to a request QDoc, wrapping the message in a SOAP 1.2 envelope.

QAD MFG/PRO-message-to-business-object processing

The sections below describe:

- Triplet-to-QDoc processing
  - Rules
  - metadata
  - Step-by-step processing
  - Header mapping
- QDoc-to-business-object processing
- The mapping metadata required for such transactions

Triplet-to-QDoc processing: rules

The rules below apply when the QAD MFG/PRO data handler's mapping engine converts triplet messages to QDoc XML:

- **Field-element mapping** Triplet field names are mapped to XML elements.
- **Naming convention** All element names corresponding to fields in the triplet message may be derived by converting the underscore-delimited MFG/PRO names to mixed-case or 'camel-case' notation (for example `pt_pl_line` becomes `ptPlLine`).
- **Name-field association** The entity-level element names corresponding to MFG/PRO business objects or database records that contain the triplet fields are not derived automatically from any existing MFG/PRO metadata. Rather, the triplet fields are established by QAD and reflect common business and MFG/PRO usage. For example, the operation element that corresponds to the action field in the triplet must be present on all XML elements that have been updated in MFG/PRO as part of the published event. In the case of a parent element that was not affected, but is present in the QDoc only to reference the affected child, the operation field may be omitted (but is still required on the child). This element is relevant for Create, Update and Delete requests only, not for business transactions such as order shipment, inventory issue, receipt, and so on.
- **Array representation** Arrays representations in QDocs must comply with SOAP 1.2 encoding standards that prohibit the transmission of partial arrays inside a SOAP message. To distinguish between unpopulated array entries and array entries whose value is being set to the empty string, two QDoc-specific attributes, 'index' and 'skip' have been defined. Here is an example:

```
<orderQuantity
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:enc="http://www.w3.org/2002/12/soap-encoding"
   xmlns:qcom="http://www.qad.com/qdoc/common"
   enc:itemType="xsi:decimal"
   enc:arraySize="6"
   >
   <!-- Entries 1, 3 are set to empty and 4, 6 are skipped -->
   <qcom:entry index="1"/>
   <qcom:entry index="2">10.583</qcom:entry>
   <qcom:entry index="3"/>
   <qcom:entry index="4" skip=ītrueî/>
   <qcom:entry index="5">22</qcom:entry>
   <qcom:entry index="6" skip="true">ignore!</qcom:entry>
</orderQuantity>
```

- **De-Normalization of parent-child tables for QDocs** Triplet documents may contain data from tables that are closely related to other tables, through parent-child associations. In most cases, a single QDoc representing the parent object is used to publish events affecting all the child tables as well. The use of a single type of QDoc for multiple types of triplet documents does not imply that multiple instances of triplet documents should be merged at run-time to generate a single QDoc instance. In order to preserve the batch-of-one-immediate, event-driven nature of DataSync, each triplet document is mapped to a single QDoc. However, the same QDoc type will be shared by multiple triplet document types. For example, the triplet document types for the Analysis Code Master (`an_mstr`), Analysis Code Link Detail (`anl_det`), and Analysis Code Selection Detail (`ans_det`) tables are mapped to a single `maintainAnalysisCode` QDoc type.
  In order to accomplish the mapping, triplet data from the parent table is mapped to the highest-level XML element inside the QDoc. Data from the child table triplets are mapped to lower-level child elements. However, the QDoc XML data is represented in a structured, de-normalized fashion, with elements for the common keys in the parent and removed from the child elements. Such redundant foreign keys are unnecessary in XML, in which the parent-child relationship is implicit in the structured nesting of elements. The mapping algorithm recognizes the common key fields for each MFG/PRO table at run-time. If the Triplet document updates child, rather than parent, table records, the common key values are moved from the child XML element to the parent XML element in the resulting QDoc. Mapping of fields representing common keys that appear in the triplet at the child table level is performed by replacing the child table prefix with the parent table prefix. By MFG/PRO convention, all field names have a prefix that is an abbreviation for the database table name, followed by an underscore character, followed by one or more underscore-delimited tokens that together identify the field. Common keys across parent and child tables always have the same field name, except for the table-specific prefix. For example, the costSimulation (`sct_det`) fields equivalent to the `spt_det` foreign key fields are `sct_sim` for `spt_sim`, `sct_part` for `spt_part`, and `sct_site` for `spt_site`.
- **Document-to-record mapping** All triplet documents contain data from one and only one MFG/PRO database record.
- **Document-to-record mapping exceptions: customers and suppliers with address data** Customer and supplier address data are stored inside a common, multi-purpose MFG/PRO table (`ad_mstr`) and cross-referenced elsewhere (`ls_mstr`) to allow for the possibility of reusing the same address record as a customer and supplier address. In addition to the primary customer or supplier address, there are a variable number of ship-to addresses associated with a given customer and remit-to addresses associated with a given supplier. Triplet documents for the customer (`cm_mstr`), supplier (`vd_mstr`), and address (`ad_mstr`) tables contain address data mixed with customer or supplier master data. Accordingly, the triplet format for customer and supplier data varies slightly from the other triplet document types as follows:
  - The data consists of a variable number of newline-delimited lines, with the data retrieved from each physical MFG/PRO database record stored on its own line.
  - Instead of beginning each line with the table name in brackets, the token `customer` or `supplier` is used instead of `cm_mstr` or `vd_mstr`. The lines containing address data begin with the token `customeraddress`, `supplieraddress`, `ship-toaddress`, or `remit-toaddress` in brackets, depending on the type of address data present, instead of the table name `ad_mstr`.

– When the primary customer or supplier address data is updated, a fixed set of customer or supplier master (cm_mstr or vd_mstr) elements are published as well as the new address data, even if the former was not affected by the maintenance event.
– When a 'ship-to' or 'remit-to' address is updated, a fixed set of customer or supplier master (cm_mstr or vd_mstr) and sometimes the primary customer or supplier address (ad_mstr) elements are published as well as the new address data, even if the first two were not affected by the maintenance event.

Triplet-to-QDoc processing: metadata

The rules described for triplet-to-QDoc processing are applied during runtime and guided by mapping metadata. The mapping logic resides in the OutboundMappingClass, which is also a child attribute in the QADDataHandlerMO, and the metadata resides in the mapFile, which is a child of the OutboundMappingMO. The mapFile is an XML file containing metadata that:

- maps the triplet table name to the corresponding QDoc schema and business object name
- specifies the parent business object name and the foreign keys used in the child representation
- and, for exceptions:
  – maps the triplet token used in lieu of the table name to the name of the corresponding MFG/PRO table maps primary customer and supplier address fields (that appear at the same level as the customer and supplier elements within the QDoc) to the customer or supplier master table, not to the address table

Table 8 below describes this information, which is stored in the mapFile.

Table 8. mapFile metadata

Table name	QDoc name	BO name	Parent object	Foreign keys
ac_mstr	maintainAccount	account		
an_mstr	maintainAnalysisCode	analysisCode		
anl_det	maintainAnalysisCode	analysisCodeLink	analysisCode	anl_type, anl_code
ans_det	maintainAnalysisCode	analysisCodeSelection	analysisCode	ans_type, ans_code
bom_mstr	maintainBillOfMaterial	billOfMaterial		
cc_mstr	maintainCostCenter	costCenter		
ccd1_det	maintainCostCenter	costCenterAcct Validation	costCenter	ccd1_cc
ccd2_det	maintainCostCenter	costCenterSubacct Validation	costCenter	ccd2_cc
cd_det	maintainMasterComments	masterComment		
cm_mstr (published with related ad_mstr data)	maintainCustomer	customer		
ad_mstr (also accessed to synchronize customers, suppliers)	maintainAddress or maintainCustomer or maintainSupplier			

Table 8. mapFile metadata (continued)

Table name	QDoc name	BO name	Parent object	Foreign keys
ls_mstr (also accessed to synchronize customers, suppliers; synchronized independently only for non-MFG/PRO address types)	maintainAddressList	addressList		
code_mstr	maintainCode	code		
cp_mstr	maintainCustomerItem	item		
cs_mstr	maintainCostSet	costSet		
cu_mstr	maintainCurrency	currency		
dpt_mstr	maintainDepartment	department		
en_mstr	maintainEntity	entity		
exr_rate	maintainExchangeRate	exchangeRate		
fcs_sum	maintainForecastSummary	forecastSummary		
glc_cal	maintainGeneralLedger Calendar	generalLedger Calendar		
is_mstr	maintainInventoryStatus	inventoryStatus		
isd_det	maintainInventoryStatus	inventoryStatus		inventorySta tus
pc_mstr	maintainPurchasingPrice List	purchasingPriceList		
pi_mstr	maintainPriceList	priceList		
pid_det	maintainPriceList	priceListDetail		priceList
pl_mstr	maintainProductLine	productLine		
ps_mstr	maintainProductStructure	productStructure		
pt_mstr	maintainItem	item		
ro_det	maintainRoutingOperation	routingOperation		
sb_mstr	maintainSubAccount	subAccount		
sbd_det	maintainSubAccount	subAccountDetail		subAccount
si_mstr	maintainSite	site		
sct_det	maintainCostSimulation	costSimulation		
spt_det	maintainCostSimulation	costSimulationItem		costSimula tion
um_mstr	maintainAlternateUnitOf Measure	alternateUnitOf Measure		
vd_mstr (published with related ad_mstr data)	maintainSupplier	supplier		
vp_mstr	maintainSupplierItem	supplierItem		
wc_mstr	maintainWorkCenter	workCenter		

The following sample fragment shows how the metadata in mapFile maps items:

```
<!-- Master table record - no parent -->
<record  name="ac_mstr">
        <qdoc name ="maintainAccount" version="1.0"/>
        <bo> account</bo>
</record>

<!-- child table record ñ parent and foreign keys required -->
<record name= "anl_det" >
        <qdoc name = "maintainAnalysisCode" />
        <bo> analysisCodeLink </bo>
         <parent>an_mstr</parent>
        <fkey>anl_type</fkey>
        <fkey>anl_code</fkey>
</record>

<!-- Customer Master table record - no parent -->
<record name= "cm_mstr">
        <qdoc name="maintainCustomer" />
        <bo> customer </bo>
</record>
<!-- Address master table record -->
<record name= "ad_mstr" >
        <qdoc name="maintainAddress" version="1.1"/>
        <bo>address</bo>
</record>

<!-- exception cases mapping -->
< record name= "customer" >
        <qdoc name="maintainCustomer" />
        <bo> customer </bo>
</ record >
< record name= "customeraddress"alias="ad_mstr" >
        <qdoc name="maintainCustomer" />
         <parent>customer</parent>
</ record >
< record name= "ship-toaddress" >
        <qdoc name="maintainCustomer" />
         <parent>customer</parent>
        <bo>shipToAddress</bo>
</record>

<exit_on_error>
        <bo> customer</bo>
        <bo> analysisCodeLink </bo>
</exit_on_error>
```

The exception cases in the fragment are mapped using the same syntax as that for the regular records. The difference is that the triplet token is specified instead of the table name. The `alias` attribute will contain the actual table name and will be used to retrieve the prefix for building the corresponding QDoc element names. For primary customer and supplier addresses, no business object name is specified. This forces the address fields to appear within the customer and supplier fields at the same level.

The `exit_on_error` tag indicates whether the connector should shutdown when it encounters an error during the processing of the specified business object. By default, the connector ignores the such errors and continues event retrieval and processing.

Triplet-to-QDoc header mapping

Table 9 illustrates the mapping of the QDoc header elements to QAD MFG/PRO control tags in the triplet message header.

Table 9. QDoc-triplet header mapping

QDoc element	Description	Triplet content
`senderId`	Sender URI	`@SYSID` tag
`receiverId`	Receiver URI	`@TRADPTRID` tag
`senderDocumentId`	QAD MFG/PRO's internally assigned document ID	The `senderDocumentId` on QDocs must be referenced on QDoc Confirmation messages so that the QAD MFG/PRO application can cross -reference and confirm the original QDoc
`descriptor`	Multiple-occurrence text string describing the type of document contained in the message. First occurrence: Document Standard Second occurrence: Document Type Third occurrence: Document Revision	Not used.
`confirmationLevel`	Designates under what circumstances a QDoc Confirmation document is requested. Possible values are `none`, `error`, or `all`.	`@ACKLVLREQD` tag
`dateTimeCreated`	Contains the date- and time-stamp for the creation of the document, expressed using the `DateTime` type prescribed by XML schema	Combination of the following triplet header tags: `@DATECREATE`, `@TIMECREATE`, and `@TIMEZONE`
`senderDocumentRef`	Multiple occurrence (up to two occurrences) string providing a reference to the sending application	`@MFGPROSITE` (if specified)`@MFGPROKEY`
`receiverDocument Ref`	Multi-occurrence string (up to two occurrences) providing a reference to the receiving application.	`@MFGPROSITE` (if specified)`@MFGPROKEY`

This mapping data is stored in a text file that is referenced in the outbound mapping meta-object. The file's content is as follows:

```
senderId @SYSID
receiverId @TRADPTRID
senderDocumentId @DOCID
confirmationLevel @ACKLVLREQD
dateTimeCreated @DATECREATE:@TIMECREATE:@TIMEZONE
senderDocumentRef @MFGPROSITE:@MFGPROKEY
receiverDocumentRef @MFGPROSITE:@MFGPROKEY
```

QDoc-to-business-object processing

Once the QDoc is retrieved, the QAD MFG/PRO data handler invokes the XML data handler to create the corresponding business object. The business object verb is based on the value of the corresponding operation tag in the QDoc:

- If the root entity in the QDoc has the operation tag set, the business object verb is set as follows:
  - `Create` for `A`
  - `Update` for `C`
  - `Delete` for `R`
- If the root entity in the QDoc does not have the operation tag set, the operation value is retrieved from the child entity and:

1. The root business object verb is set to <action>Child where <action> is based on the operation value: `Create` for `A`, `Update` for `C`, `Delete` for `R`
2. The child business object verb is set to the appropriate value for the operation.

For a full description of XML data handler processing, see the *Data Handler Guide*.

Business-object-to-QAD MFG/PRO-message processing

For request processing, the connector passes the request portion of a TLO to the QAD MFG/PRO data handler. The data handler calls the XML data handler to transform the header and body portions of the request into XML. It assembles these pieces together to build a SOAP-XML 1.2 message and returns it to the connector. The connector then makes an HTTP request to the QXtend web service and wait for a reply. The reply message is converted to a BO by the QAD MFG/PRO data handler, which again calls the XML data handler to do part of the work. This BO is put into the response attribute of the TLO, and the TLO is returned to the broker. If the QXtend web service returns a fault QDoc message instead of a reply QDoc message, the QAD MFG/PRO data handler converts it into a `BIA_FaultBO` business object. The protocol handler sets this BO as the value of the fault attribute of the TLO, and the header is removed.

Chapter 5. Troubleshooting

The chapter describes problems that you may encounter when starting up or running the connector.

Start-up problems

In case of unexpected failure of the adapter, specify a trace level of 5 in the AgentTraceLevel connector configuration property and restart. For further information on tracing, see “Tracing” on page 34. For further information on the AgentTraceLevel property, see Appendix A, “Standard configuration properties for connectors,” on page 49.The table below describes additional start-up troubleshooting.

Problem	Potential solution / explanation
The connector shuts down unexpectedly during initialization and the following message is reported: `Exception in thread "main" java.lang.NoClassDefFoundError: javax/jms/JMSException...`	Connector cannot find file `jms.jar` from the IBM WebSphere MQ Java client libraries. Windows: Ensure that the environment variable `%MQ_LIB%` on your system points to the IBM WebSphere MQ Java client library folder Unix: Ensure the variable `$MQ_LIB` in the file `<wbia>/bin/CWSharedEnv.sh` is set to the IBM WebSphere MQ Java client library folder.
The connector shuts down unexpectedly during initialization and the following message is reported: `Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/mq/jms/MQConnectionFactory...`	Connector cannot find file `com.ibm.mqjms.jar` from the IBM WebSphere MQ Java client libraries. Windows: Ensure that the environment variable `%MQ_LIB%` on your system points to the IBM WebSphere MQ Java client library folder. Unix: Ensure the variable $MQ_LIB in the file `<wbia>/bin/CWSharedEnv.sh` is set to the IBM WebSphere MQ Java client library folder.
The connector shuts down unexpectedly during initialization and the following message is reported: `Exception in thread "main" java.lang.NoClassDefFoundError: javax/naming/Referenceable...`	Connector cannot find file `jndi.jar` from the IBM WebSphere MQ Java client libraries. Windows: Ensure that the environment variable `%MQ_LIB%` on your system points to the IBM WebSphere MQ Java client library folder. Unix: Ensure the variable `$MQ_LIB` in the file `<wbia>/bin/CWSharedEnv.sh` is set to the IBM WebSphere MQ Java client library folder.
The connector shuts down unexpectedly during initialization and the following exception is reported: `java.lang.UnsatisfiedLinkError: no mqjbnd01 in shared library path`	Connector cannot find a required run-time library (`mqjbnd01.dll` [NT] or `libmqjbnd01.so` [Solaris]) from the IBM WebSphere MQ Java client libraries. Ensure that your path includes the IBM WebSphere MQ Java client library folder.
The connector reports `MQJMS2005: failed to create MQQueueManager for ':'`	Explicitly set values for the following properties: `HostName`, `Channel`, and `Port`.

Event processing

In case of unexpected connector termination, check the InProgressQueue queue for events that may have not yet been processed. For further information on the InProgressQueue, see “InProgressQueue” on page 18.

© Copyright IBM Corp. 2004

Loss of connection to QAD MFG/PRO application

For event handling, the adapter communicates with the application through WebSphere MQ queues. Make sure that the queue manager is up and running. For request processing, SOAP-HTTP is used for communication. Make sure the Tomcat application server where Qxtend is installed is up and running, and that you can reach that machine from the adapter machine. You can test this by opening a browser on the adapter machine and open the URL `http://`*`qxtendmachine`*`:8080/qxtendserver` where *qxtendmachine* is the host name of the machine where Tomcat is running; `8080` is valid only if you haven't changed the default in Tomcat. If this URL does not open, or if you get an error page back, that means Qxtend is not running or cannot be reached. See Qxtend documentation for details.

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

© Copyright IBM Corp. 2004

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 10 on page 51 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 10. 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 10. 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 10. 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 10. 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 56.

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 66).

© Copyright IBM Corp. 2004

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 67 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 when the broker is IBM WebSphere InterChange Server:

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

To do so when you have WebSphere Business Integration Adapters and another broker installed:

- From **Start>Programs**, click **IBM WebSphere Business Integration Adapters>Tools>Connector Configurator**.
- Select **File>New>Connector Configuration**.
- When you click the pull-down menu next to **System Connectivity Integration Broker:**, you can select WMQI or WAS connectivity, 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 71.)

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 67.
- 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 74..

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 50.

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>
```

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

© Copyright IBM Corp. 2004

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

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

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

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

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

IBM WebSphere Business Integration Adapter Framework V2.4.0.