IBM WebSphere Business Integration Adapters

Adapter for Ariba Buyer User Guide

Adapter Version 3.2.x

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

19December2003

This edition of this document applies to IBM WebSphere Business Integration Adapter for Ariba Buyer, version 3.2.x, and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about IBM WebSphere Business Integration documentation, e-mail doc-comments@us.ibm.com. We look forward to hearing from you.

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

Contents

About this document

The IBM® 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, business object development, and troubleshooting for the IBM WebSphere Business Integration Adapter for Ariba Buyer.

Audience

This document is for consultants, developers, and system administrators who use the adapter at customer sites.

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.

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, file name, information that you type, or information that the system prints on the screen.
italic	Indicates a new term the first time that it appears, 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, a pipe separates 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.
`< >`	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 connector for Ariba Buyer is installed on your system.
`%text%` and *`$text`*	Text within percent (%) signs indicates the value of the Windows `text` system variable or user variable. The equivalent notation in a UNIX environment is *`$text`*, indicating the value of the *`text`* UNIX environment variable.
`ProductDir`	Represents the directory where the product is installed. For IBM WebSphere InterChange Server, the default product directory is IBM\WebSphereICS. For IBM WebSphere Business Integration Adapters, the default product directory is WebSphereAdapters.

New in this release

New in release 3.2.x

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

The adapter now supports version 8.1 of Ariba Buyer.

Beginning with version 3.2, the adapter for Ariba Buyer is no longer supported on Microsoft Windows NT.

New in release 3.1.x

The adapter can now use WebSphere Application Server as an integration broker. For more information, see "Adapter environment" on page 7.

The adapter now runs on the following platforms:

- Solaris 7, 8
- AIX 5.1, 5.2
- HP-UX 11i

The adapter has a new adapter-specific configuration property, `BOLocale`. For more details, see "Application-specific configuration properties" on page 12.

New in release 3.0.x document patch

Users who are upgrading their systems can find migration information in Appendix C, "Migrating the integration channel," on page 79.

This appendix explains how to migrate the Ariba Buyer integration channel from the previous version of the adapter, WebSphere Adapter for Ariba Buyer (WAAB), to the current version, WebSphere Business Integration Adapter for Ariba Buyer, when upgrading from Ariba Buyer 7.x to Ariba Buyer 8.x.

Chapter 1. Overview of the adapter

This chapter describes the IBM WebSphere Business Integration adapter for Ariba(R) Buyer and the associated system architecture.

The adapter for Ariba Buyer consists of a connector, the adapter framework, and an integration channel. Together they enable Ariba Buyer to exchange information with an integration broker such as the WebSphere MQ Integrator Broker or the WebSphere InterChange Server (ICS).

Version 3.2 of the adapter for Ariba Buyer enables integration to version 8.1 of Ariba Buyer. It is not compatible with earlier versions of the application. The channel component runs on all platforms that support Ariba Buyer, whether or not they support the adapter framework. The connector component runs on all platforms that support the adapter framework.

For more information about the relationship of the integration broker to the adapter, see *IBM WebSphere Business Integration System Administration Guide*.

This chapter contains the following sections:

Adapter architecture

The adapter for Ariba Buyer enables Ariba Buyer to communicate with external applications such as SAP, Oracle Financials, and PeopleSoft.

Components

The adapter includes three primary components:

- **Integration channel**

  Required by Ariba Buyer to communicate with target applications. It runs as a component of Ariba Buyer and communicates with the connector remotely.
- **Connector**

  Acts as an intermediary between the integration broker and the application in the transfer of data. The connector is metadata-driven.
- **Object Discovery Agent (ODA)**

  A design-time tool that reads metadata exported by the application and converts it into business object definitions. The connector uses these definitions at run time to convert application data into business objects.

Figure 1 on page 2 shows how these adapter components work together. They are described in detail in "Adapter components" on page 3.

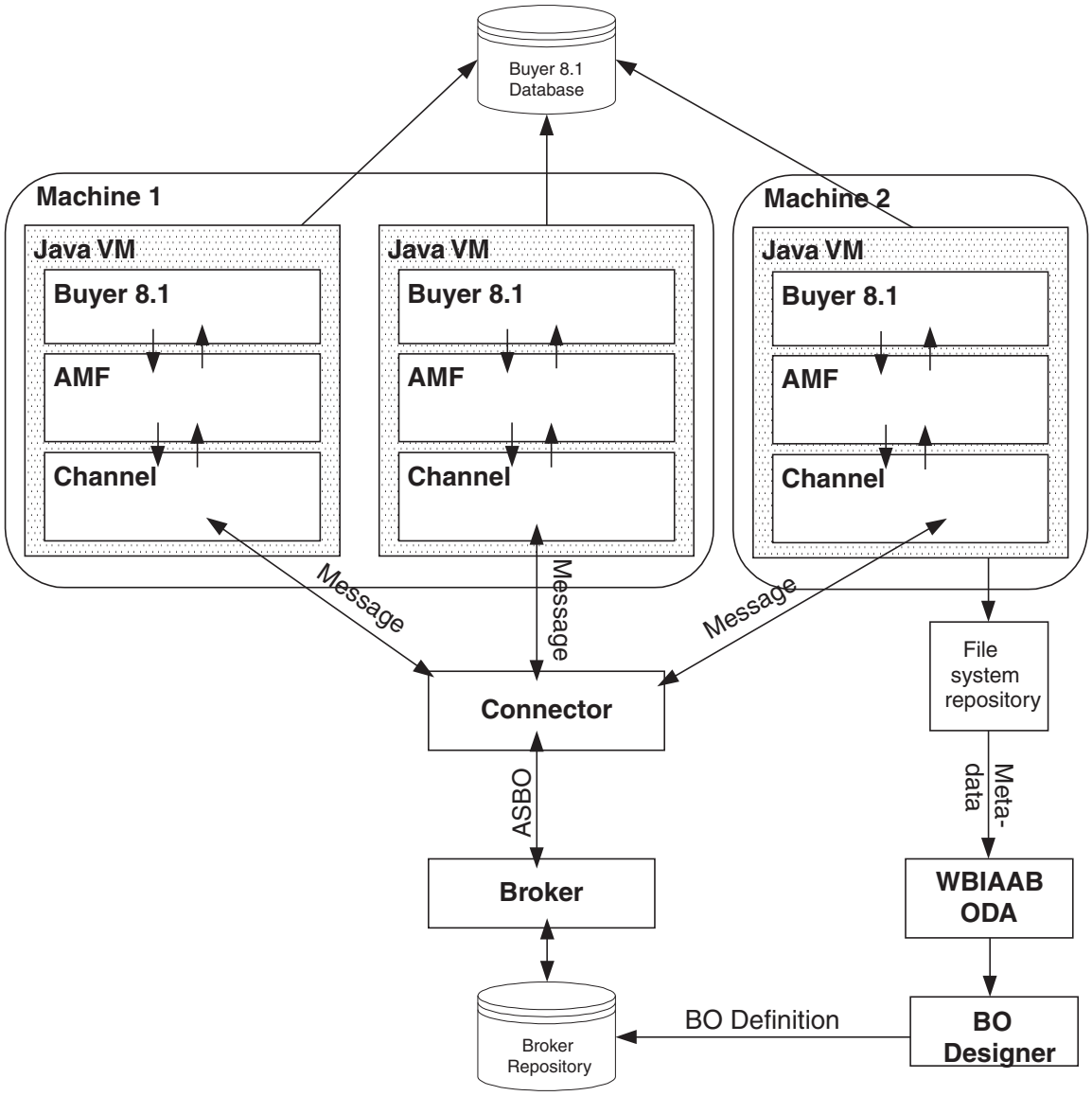

Figure 1. Adapter architecture

Connectors and nodes

Ariba Buyer may be run in a multi-node configuration with the nodes all on one machine, or distributed across many machines in a multi-server configuration—provided that all the servers run the same operating system. In a multi-server configuration, each machine has at least one node. However, there is only one instance of the connector, regardless of the number of nodes.

Each node runs in its own process and communicates with its own instance of the integration channel, but all nodes communicate with the same Ariba Buyer application database.

Run-time events flows

Run-time event flows are bi-directional, and the direction depends on where the event originates.

For events originating from Ariba Buyer:

1. A message object is sent to the integration channel of a specific Ariba Buyer node.
2. The integration channel sends the message object to the connector.
3. The connector maps the business object data in the message object to an application-specific business object (ASBO), using a business object definition created by the ODA at design time. It then sends the ASBO to the integration broker.

For events originating from the integration broker:

1. The broker sends an application-specific business object to the connector.
2. The connector determines whether the application-specific business object is a response to an earlier request by checking for the existence of a reply key in the object.
3. The application-specific business object is converted to a message object.
4. If the application-specific business object is a response to a previous request, the message object is routed to the node that originated the request. Otherwise, the message object is routed to the first available node.

Adapter components

The three adapter components process and manage the data flows to and from the brokers. They are described below.

Integration Channel

The integration channel is the link between Ariba Buyer and the connector. Ariba Buyer communicates with the channel using the Ariba Messaging Framework (AMF). The AMF is used by the application to send message objects to the channel, and by the channel to send objects to the application.

Communication between the AMF and the channel occurs whenever:

- An integration event defined on the channel is triggered within Ariba Buyer. The AMF can trigger events synchronously or asynchronously.
- A service call is received from the integration broker. The call is sent to the first available channel node.
- The metadata changes within Ariba Buyer. This happens only on the primary node and only in development mode.

Service call events are sent by the connector to the first available channel node, which relays the event to the application.

For more information, see Chapter 4, “Understanding business objects,” on page 21, and Chapter 5, “Generating business object definitions,” on page 29.

Connector

Connectors consist of two parts: the *application-specific component* and the *controller and communications framework*. The application-specific component contains code tailored to a particular application or technology (in this case, Ariba Buyer). The controller and communications framework, whose code is common to all connectors, act as an intermediary between the integration broker and the application-specific component by providing the following services:

- Receives and sends business objects

- Manages the exchange of startup and administrative messages

The connector for Ariba Buyer communicates with the application through the integration channel.

The integration channel and connector communicate with each other at run time. The message object originates in an Ariba Buyer node and is passed to the integration channel through the AMF. The integration channel sends the message object to the connector.

The connector maps the business object data contained in the message object to an application-specific business object, using a definition stored in the broker repository, and sends it to the broker. It may also receive business objects from the broker for delivery to Ariba Buyer. These business objects are sent back through the channel instance for the Ariba Buyer node that initiated the exchange.

For more information, see Chapter 4, "Understanding business objects," on page 21, and Chapter 5, "Generating business object definitions," on page 29.

Object Discovery Agent (ODA)

The Object Discovery Agent (ODA) is a design-time tool used to create business object definitions from Ariba Buyer-specific metadata. The ODA uses that metadata to build the business object definitions.

Two types of metadata are exported by the integration channel:

- **Class metadata**, which represents the business object defined in the Ariba Buyer object model.
- **Message metadata**, which represents the integration messages defined in Ariba Buyer. These messages refer to Buyer business objects; that is, Message metadata refers to Class metadata.

Metadata can be exported from the primary node of an Ariba Buyer instance running in development mode. The ODA reads these files and uses the information to create business object definitions in the integration broker repository. During run time, the connector can use these definitions to transform message objects into application-specific business objects.

Ariba Buyer provides an interface that notifies the integration channel about any changes to the metadata. The channel receives this information from the AMF through a listener and automatically saves it to the metadata repository.

Note: You can regenerate the repository by shutting down the Ariba Buyer server and running the `initdb -reshapedb` command. To regenerate the metadata, the system must be in development mode. For details, refer to the Ariba Buyer 8.x documentation.

For more information, see Chapter 4, "Understanding business objects," on page 21, and Chapter 5, "Generating business object definitions," on page 29.

How the connector works

The connector for Ariba Buyer is bi-directional. It can process events originating from Ariba Buyer applications, as well as requests sent by the broker to the application.

For event notifications, Ariba Buyer provides a mechanism using message handlers in the integration channel. To send an event, an Ariba Buyer node invokes a message handler on the associated channel instance, which in turn contacts the connector. All events are thus sent to the connector without the need for any other form of notification, such as polling.

For request processing, the connector for Ariba Buyer processes the requests coming from the integration broker in the form of business objects. It sends a message object to the Subscribe message handler in the integration channel, which in turn sends it to the application.

The connector for Ariba Buyer follows the metadata design principles for WebSphere Business Integration adapter connectors. This means new business objects can be defined without additional coding or customization at the connector code level. For more information, see Chapter 4, "Understanding business objects," on page 21.

Processing events

The following sections describe how the connector processes application events.

Event detection and notification

No event detection is required for Ariba Buyer.

The application notifies the integration channel of an event by sending it a message object containing the event data. When the channel receives the message object, it sends it to the connector, where the business object handler converts the message into an application-specific business object for transmission to the integration broker. The channel's message handlers can simultaneously receive and forward a number of events, each of which is sent by a different Buyer thread.

Ariba Buyer events may be asynchronous or synchronous. When the application sends an event of either type to the channel, it also stipulates the return business object that will deliver a response.

Synchronous events

When Ariba Buyer sends a synchronous event to the integration channel, it sends the message object to the connector and retains control of the Buyer thread while it waits for a response. When the connector receives the response application-specific business object, it transforms this object into a message object and sends it back to the waiting channel node.

If the connector goes down before the synchronous response is returned to the channel, the event generates an error. For more information, see Time-outs in Chapter 6, "Troubleshooting and error handling," on page 39.

Asynchronous events

When Ariba Buyer sends an asynchronous event to the integration channel, the channel will start a timeout process if timeout is specified for the message. The channel then sends the message object to the connector and returns control back to the Buyer thread that invoked it. When the connector receives the response application-specific business object, it transforms this object into a message object and sends it back to the originating channel node.

If no response is sent for an asynchronous event, or the response is sent very late, the event times out. For more information, see Time-outs in Chapter 6, "Troubleshooting and error handling," on page 39.

Service Call Request processing

Whenever a Service Call is initiated on Ariba Buyer, the connector receives the application-specific business object from the integration broker, transforms it into a message object, and passes it to the first available node. The channel instance for the receiving node receives the message object and passes the data to Ariba Buyer using the Ariba Buyer-defined APIs.

Status updates

In general, errors are logged by the failing component and the originating component is notified.

Errors that occur in the integration channel, as well as any messages, are logged by the application API and appear in the Ariba Buyer log file (AribaBuyerNodexLog.txt).

Errors that occur in the connector, along with any messages, appear in the connector log file, and the connector framework is informed. You specify the name of this file in the Connector Configurator at system setup.

For more information, see Chapter 6, "Troubleshooting and error handling," on page 39.

Chapter 2. Installing the adapter

This chapter describes how to install the IBM WebSphere Business Integration adapter for Ariba Buyer. It contains the following sections:

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 3.2 of the adapter for Ariba Buyer is supported on the following adapter framework and integration brokers:

- **Adapter framework**:
  WebSphere Business Integration Adapter Framework versions 2.3.1, and 2.4.
- **Integration brokers:**
  - WebSphere InterChange Server, versions 4.1.1, 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.

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

Before you install the adapter, your system should have the following software installed and configured:

Operating systems:

One of the following application platforms:

- AIX 4.3.3, AIX 5.1, AIX 5.2 (64-bit)
- Solaris 7.0, Solaris 8.0
- HP UX 11.0, HP UX 11i
- Windows NT (SP6a), Windows 2000 (SP3)

Databases:

- DB2 7.2 (FP9)
- Oracle 8.1.7.4, Oracle 9.2
- MS SQL Server 7 (SP4), MS SQL Server 2000

Third-party software:

- Ariba Buyer 8.1 (version 3.2 of this adapter is not compatible with Ariba Buyer 8.0)

Note: You may install the adapter before or after you install Ariba Buyer 8.1. However, the adapter must be installed **before** you configure the application. The configuration process for Ariba Buyer needs to copy files from the adapter installation directory to the Ariba Buyer installation directory.

After you install the adapter, you are ready to configure Ariba Buyer 8.1. For details, refer to the documentation for Ariba Buyer 8.1.

Adapter dependencies

The adapter for Ariba Buyer has the client library/API dependencies shown in Table 1.

Table 1. Adapter dependencies

Library/API	Version	Operating systems
ariba.util.core.zip	Ariba Buyer 8.1	All supported OSs
ariba.util.messaging.zip	Ariba Buyer 8.1	All supported OSs
ariba.amf.zip	Ariba Buyer 8.1	All supported OSs
ariba.base.zip	Ariba Buyer 8.1	All supported OSs
ariba.tools.migration.zip	Ariba Buyer 8.1	All supported OSs
ariba.tools.taskharness.zip	Ariba Buyer 8.1	All supported OSs
jakarta-oro-2.0.jar	2.0	All supported OSs

Globalization

This adapter is DBCS (double-byte character set)-enabled and is translated.

Overview of the installation process

These are the steps you follow to install the adapter for Ariba Buyer.

1. Install the integration broker.
2. Install Ariba Buyer.

Note: Select **No** when you are prompted to proceed to the Ariba Buyer configuration program at the end of the installation.

3. Install the adapter for Ariba Buyer.
4. Configure and initialize Ariba Buyer.

It is assumed that steps 1 and 2 are complete at this time. The installation instructions in this guide cover steps 3 and 4.

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

Installed file structure

Table 2 shows the file structure used by the adapter for Ariba Buyer and lists the files that are installed on the system.

Notes:

1. This document uses (\) backslashes as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes.
2. All product path names are relative to the directory where the product is installed on your system.

Table 2. Adapter files installed on system

Subdirectory of %Prod_Dir%	Description
_uninstall	Contains the files for uninstalling the adapter
connectors\Buyer	Contains: • The adapter CWBuyer. jar file, version 3.2.0, which has the connector code • The start_Buyer.bat file (WIN) or start_Buyer.sh file (UNIX) • The \Buyer directory that will be copied to the Ariba Buyer installation directory
connectors\messages	Contains the BuyerConnector.txt file, which lists error and other messages.
legal\license	Contains a text file with the terms of the adapter license agreement.
ODA\Buyer	Contains: • The Object Discovery Agent BuyerODA.jar file, version 3.2.0, which has the ODA code • The start_BuyerODA.bat file (WIN) or start_BuyerODA.sh file (UNIX)
ODA\messages	Contains the BuyerODAAgent.txt file, which lists error and other messages.
repository\Buyer	Contains the CN_Buyer.txt file, which specifies connector configuration properties.

Completing the installation

The \Buyer subdirectory resides under the \connectors\Buyer directory (see Table 2 on page 9). The files in this subdirectory contain certain configuration information for the adapter channel.

To complete the installation, copy this subdirectory on top of the Ariba Buyer 8 home directory and update the class path, as follows.

1. Copy the \Buyer subdirectory from the source directory on top of \<*Ariba Buyer Server root*>\; for example, \Ariba\app\Buyer\
2. Modify /Ariba/app/Buyer/classes/classpath.txt by adding these entries:

   ```
   #IBM ERP Channel implementation
   classes/IBMChannel.jar
   classes/ABCommon.jar
   ```

   classes/BIA_IBMMigration.jar(required only if you have migrated from WAAB 1.x or 2.x: see Appendix C, “Migrating the integration channel,” on page 79 for details.)
3. Save the file. You are now ready to configure Ariba Buyer.

Chapter 3. Configuring the adapter

This chapter describes how to configure the IBM WebSphere Business Integration adapter for Ariba Buyer and how to configure the application to work with the adapter. It contains the following sections:

Configuring Ariba Buyer for the adapter

Before you can use the connector, you must configure Ariba Buyer to recognize the adapter. You use the Ariba Buyer Configure tool to set the correct enterprise application integration (EAI) channel in Ariba Buyer, as follows.

1. In the Ariba Buyer Configuration window, check **EAI Channel** and click **Next**.
2. Select your database, for example, IBM DB2 UDB, and click **Next**.
3. Enter the database properties for your database and click **Next**.
4. If you have the Ariba Buyer 8 Hot Fix Pack 1 installed, select **IBM WBI Adapter** as your EAI Channel and set the Configuration File Location as:

   <Ariba Buyer Server root>/channels/IBM
5. If you do **not** have the Ariba Buyer 8 Hot Fix Pack 1 installed:

   Select **Other** as your EAI Channel.

   Set **IBM** as the channel name by setting the Configuration File Location as:

   <Ariba Buyer Server root>/channels/IBM
6. Continue with the configuration for Ariba Buyer 8.x.

Note: The adapter is only used if you select an ERP Type other than **Basic CSV configuration** during configuration.

Configuring the adapter

The connector component of the adapter has two types of configuration properties: standard configuration properties and application-specific configuration properties. You must set the values of these properties before running the connector.

Standard configuration properties

To configure standard connector properties, use the Connector Configurator tool. Details are given in Appendix B, "Connector Configurator," on page 63.. This tool provides a graphical user interface for configuring the connector. Click on the **Standard Config Properties** tab to add or modify configuration properties.

When you have finished specifying values for the connector's configuration properties, Connector Configurator saves the values in the adapter repository (for ICS) or generates a configuration file and places it in the adapter's local repository (for WebSphere MQ Integrator Broker or WebSphere Application Server).

A connector obtains its configuration values at startup. During a run-time session, you may want to change the values of one or more connector properties.

- Changes to some connector configuration properties, such as AgentTraceLevel, are dynamic, taking effect immediately.
- Changes to other connector properties are static, requiring component restart or system restart after a change.

To determine whether a property is dynamic or static, refer to the update method column in Connector Configurator.

Application-specific configuration properties

Application-specific connector configuration properties provide information needed by the connector at run time. They also provide a way for you to change static information or logic within the connector without having to recode and rebuild it.

To configure these properties, use Connector Configurator. Click the **Application Config Properties** tab to add or modify configuration properties. For more information, see Appendix B, “Connector Configurator,” on page 63..

Table 3 lists the application-specific configuration properties for the connector, along with their descriptions and possible values.

Table 3. Application-specific configuration properties for Ariba Buyer

Property	Description	Possible values	Default value	Required
Nodes	The Ariba Buyer node names and fully qualified machine names	<url1>:Node1; <url2>:Node2;... <urln>:NodeN	None	Yes
ConnectorPort	The connector RMI port	Any open port - must match the value specified in the Parameters.table file	2226	Yes
ChannelPort	The channel RMI port	Any open port - must match the value specified in the Parameters.table file	2225	Yes
DateFormats	Specifies valid date formats. For more details, see “DateFormat” below.	<format>;<format>;...	EEE MMM dd HH:mm:ss z yyyy;M/d/yy; M/d/yyyy; M-d-yy; M-d yyyy; yyyyMMdd	No
VariantMappings	Correlates user-defined variant names with specific ERP types. Maps variant and partition-specific business object names to a more generic business object name.	<variant name>=<variant type prefix>;<variant name>=<variant type prefix>;...	None	No

Table 3. Application-specific configuration properties for Ariba Buyer (continued)

Property	Description	Possible values	Default value	Required
BOLocale	Specifies the locale of a business object. For more details, see "BOLocale" below.	A valid locale specification; for example, en_US, ja_JP, or de_DE. If a given value is invalid, the system default locale is used.	None	No

DateFormat
For more information on valid date formats, which are determined by the Java API, see the website
http://java.sun.com/j2se/1.3/docs/api/java/text/SimpleDateFormat.html

BOLocale
This is a hierarchical property, which you can use as follows:

- Specify a single locale for all business objects by defining BOLocale at the parent level only.
- Specify a locale for a single partition by defining BOLocale at the parent level and also declaring a child property under BOLocale. The name of the child property must match the partition name given in the business object.

Configuring the integration channel

The integration channel must be configured for two purposes:

- Communicating with the connector
- Handling integration events

Configuring channel properties

The integration channel configuration properties provide the information necessary for the channel to communicate with the connector at run time. These properties are set in the Parameters.table text file, located in the Ariba Buyer installation directory:

```
\Ariba\app\Buyer\Server\config
```

To configure the integration channel properties, open the Parameters.table file and enter the property values.

Table 4 lists the required configuration properties, along with a description and possible value for each one.

Table 4. Application-specific configuration properties for the integration channel

Property	Description	Possible values	Default value	Required
ConnectorURL	The hostname of the machine on which the connector is running	Fully qualified machine name or IP address	localhost	Yes
ConnectorPort	The connector RMI port	Any open port - must match the value set for the connector	2226	Yes
ChannelPort	The channel RMI port	Any open port - must match the value set for the connector	2225	Yes

Table 4. Application-specific configuration properties for the integration channel (continued)

Property	Description	Possible values	Default value	Required
MetadataRepos	The location where generated metadata will be stored	Relative from \Ariba\app\Buyer\Server or a fully qualified directory path	"/channels/ IBM/metadata";	Yes

Configuring integration events

Each integration event must be configured before it can be sent through the integration channel. Events are configured in two configuration files, MessageDefinition.table and MessageConfiguration.table.

Ariba Buyer supplies a number of out-of-the-box integration events, which are listed in the *Ariba Buyer Configuration Reference Guide.* However, events that are not configured on the integration channel will not be sent through the adapter for Ariba Buyer.

The **MessageDefinition.table** file contains general information for each integration event in a given ERP variant type. Ariba Buyer supplies a table entry for a base set of out-of-the-box integration events for each of these variants.

The **MessageConfiguration.table** file contains more specific information, including channel-specific data, for each integration event in a given partition. An entry for each event must be created in this table.

A sample MessageConfiguration.table entry is shown below.

```
PurchaseOrderPush={
    Channel={
        Name=IBM;
        Operation="Update Elements Only";
        Verb=Retrieve;
        Timeout=300000;
    };
    LoggingName=PurchaseOrderEvent;
    MessageParameters={
        Request={
             MergedSchemaName="ariba.integration.param.PurchaseOrderPush";
             Parameters={EventSource="ibmcsvp:PO:SAP3.1H";};
             SchemaName=ariba.integration.param.SimpleParams;
        };
    };
    TopicName=PurchaseOrderPush;
    ExecutionNode="Node2";
};
```

Every MessageConfiguration.table entry contains a section for channel parameters, shown as the first subentry in the example above. Table 5 lists the valid parameters and values for the integration channel.

Table 5. Event configuration parameters for the integration channel

Parameter	Description	Possible values
Name	Specifies the ERP integration channel name.	IBM
Operation	Specifies the operations that Ariba Buyer can perform. Refer to "Business object verbs" on page 23 for more details.	Create, Delete, Load, "Load And Delete", Update, "Update And Delete", "Update Elements Only"

Table 5. Event configuration parameters for the integration channel (continued)

Parameter	Description	Possible values
Verb	Specifies the operations that the destination application can perform. Refer to "Business object verbs" on page 23 for more details.	Create, Delete, Retrieve, Update ABUpdate, Load, Update_And_Delete, Update_Elements_Only
Timeout	Specifies the timeout for an Ariba Buyer event process. The value is designated in milliseconds.	Whole number >=0 Default value is 0, which means no timeout for that event.
EventSource	Uniquely identifies data associated with this event. This parameter is required only for Subscribe events.	Any string value; maximum length 50 characters. Must be unique for each partition, and may not be changed between integration events.

For complete details on configuring the other parameters for integration events, refer to the *Ariba Buyer Configuration Reference Guide.*

Creating multiple connector instances

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

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

Create a new directory

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

```
ProductDir\connectors\connectorInstance
```

where `connectorInstance` uniquely identifies the connector instance.

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

```
ProductDir\repository\connectorInstance
```

Create business object definitions

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

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

   ```
   ProductDir\repository\initialConnectorInstance
   ```

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

Create a connector definition

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

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

Create a start-up script

To create a startup script:

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

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

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

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

Starting the adapter

To start the adapter for Ariba Buyer, you must:

1. Start Ariba Buyer 8.x, which starts up the integration channel.
2. Start the connector.

Note: When you want to add or reload business object definitions, you add a third step: starting the ODA.

Starting Ariba Buyer

The procedure for starting Ariba Buyer depends on the node configuration of your system. Please refer to the documentation for Ariba Buyer 8.x for details.

Each Buyer node starts an instance of the integration channel. When the channel starts, it attempts to connect to the connector. The channel starts whether or not the connector is running.

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 6 shows.

Table 6. Startup scripts for a connector

Operating system	Startup script
UNIX-based systems	`connector_manager_connName`
Windows	`start_connName.bat`

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

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

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

    `start_connName connName brokerName [-cconfigFile ]`
  - On UNIX-based systems:

    `connector_manager_connName -start`

  where *connName* is the name of the connector and *brokerName* identifies your integration broker, as follows:
  - For WebSphere InterChange Server, specify for *brokerName* the name of the ICS instance.
  - For WebSphere message brokers (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, or WebSphere Business Integration Message Broker) or WebSphere Application Server, specify for *brokerName* a string that identifies the broker.

  Note: For a WebSphere message broker or WebSphere Application Server on a Windows system, you *must* include the `-c` option followed by the name of the connector configuration file. For ICS, the `-c` is optional.
- From Adapter Monitor (WebSphere Business Integration Adapters product only), which is launched when you start System Manager

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

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector starts when the Windows system boots (for an Auto service) or when you start the service through the Windows Services window (for a Manual service).

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

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

Starting the ODA

The ODA starts when the start_BuyerODA.bat or start_BuyerODA.sh script is run. You must specify the path to the Ariba Buyer server. You can do this by:

- Setting the BUYERPATH variable in the script, or
- Specifying the Ariba Buyer server path as a command line parameter:

  ```
  C:\Ariba\app\Buyer\Server
  ```

Note: If the Ariba Buyer application and the connector are installed on separate machines, the file system of the Ariba Buyer machine must be mapped or otherwise made accessible to the file system of the machine on which the connector is installed, so that BUYERPATH can be set.

The ODA is used by Business Object Designer. When Business Object Designer connects to the ODA, the ODA reads the Ariba Buyer metadata definitions from the adapter repository, for processing into business object definitions.

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.

Using log and trace files

The adapter components provide several levels of message logging and tracing.

The **integration channel** logs information, debugging, and error messages, which are printed to the Ariba Buyer Administrator console and to the log file for the Ariba Buyer node that is originating the messages. Table 7 on page 19 lists the types of message and the default setting.

Table 7. Logging and tracing message levels for the integration channel

Message type	Description	Default
Information	Logs key initialization and termination steps, message delivery and receipts events.	On
Debug	Logs method calls and critical points within each call.	Off
Error	Logs all error conditions within the channel or detected by it.	On

The **connector** uses the adapter framework to log error, informational and trace messages. Error and informational messages are recorded in a log file, and the trace messages and trace level (0 to 5) are recorded in a trace file.

Table 8 describes the type of messages logged at each tracing level.

Table 8. Trace file tracing levels

Trace Level	Description
0	• Logs errors and fatal errors from the Ariba Buyer ODA • Logs warnings that require a system administrator's attention • Logs important messages from the application
1	Traces all entering and exiting messages for method
2	Traces the ODA's properties and their values
3	Traces the names of all business objects
4	Traces business object properties and the values received
5	• Indicates the ODA initialization values for all of its properties • Traces the business object definition dump

You configure both the log and trace files names, as well as the trace level, in Connector Configurator. Refer to Appendix B, "Connector Configurator," on page 63 for details.

The **ODA** has no logging capability; error messages are sent directly to the user interface. Trace files and the trace level are configured in Business Object Designer. The process is described in "Configure the agent" on page 31. The ODA trace levels are the same as the connector trace levels, shown in Table 8 above.

For more details on error handling and troubleshooting, refer to Chapter 6, "Troubleshooting and error handling," on page 39.

Chapter 4. Understanding business objects

This chapter describes the structure of business objects, how the adapter processes the business objects, and the assumptions the adapter makes about them.

The chapter contains the following sections:

Defining metadata

The adapter for Ariba Buyer is metadata-driven. In the WebSphere business integration system, metadata is defined as application-specific information exported by Ariba Buyer that describes its data structures. The metadata is used to construct business object definitions which the connector uses at run time to build business objects.

A metadata-driven adapter handles each business object that it supports according to the metadata encoded in the business object definition. This enables the adapter to handle new or modified business object definitions without requiring modifications to the code. Any changes are made through the Object Discovery Agent in Business Object Designer.

Application-specific metadata includes the structure of the business object and the settings of its attribute properties. Actual data values for each business object are conveyed in message objects at run time.

The adapter makes assumptions about the structure of its supported business objects, the relationships between parent and child business objects, and the format of the data. Therefore, it is important that the structure of the business object exactly match the structure defined for the corresponding object within Ariba Buyer or the adapter will not be able to process business objects correctly.

Note: The structure of the business object should never be modified using Business Object Designer.

If you need to make changes to the business object structure, make them to the corresponding object in Ariba Buyer and then export the changes to the file system repository for input into the ODA. Refer to Chapter 5, "Generating business object definitions," on page 29 for details.

For more information on modifying business object definitions, see *WebSphere Business Integration Adapters Business Object Development Guide.*

Overview of business object structure

In the WebSphere business integration system, a business object definition consists of:

- A type name
- Supported verbs
- Attributes

An application-specific business object is a particular instance of a business object definition. It reflects a specific application's data structure and attribute properties.

Some attributes, instead of containing data, point to child business objects or arrays of child business objects that contain the data for these objects. Keys relate the data between the parent record and child records.

Business objects for adapters can be flat or hierarchical. A flat business object only contains simple attributes, that is, attributes that represent a single value (such as a string) and do not point to child business objects. A hierarchical business object contains both simple attributes and child business objects or arrays of child business objects that contain attribute values.

A cardinality 1 container object, or single-cardinality relationship, occurs when an attribute in a parent business object contains a single child business object. In this case, the child business object represents a collection that can contain only one record. The attribute type is the same as that of the child business object.

A cardinality n container object, or multiple-cardinality relationship, occurs when an attribute in the parent business object contains an array of child business objects. In this case, the child business object represents a collection that can contain multiple records. The attribute type is the same as that of the array of child business objects.

Business objects for Ariba Buyer

The structure of application-specific business objects in the WebSphere Business Integration system is set by Ariba Buyer and mirrors the structure of the Ariba Buyer message object. Ariba Buyer currently supports two types of message object:

- **Request-Response:** Used to send data to an ERP application or to request data from one
- **Subscribe:** Used to receive unsolicited data from an ERP application

These message types are converted to business objects of one type or the other. Most Ariba Buyer events are of the request-response type.

Business objects inherit their names from the matching Ariba Buyer message objects. The names combine structural divisions, using the Ariba variant and partition names, and topic name. For example, SAP_NA_PurchaseOrderPush.

Business object names may not exceed 80 characters in length. If the business object name is longer than 80 characters, the ODA will display an error message and exit during business object generation. To avoid this, keep variant and partition names short.

If you do encounter this error, shorten the length of the TopicName in the MessageConfiguration.table and MessageDefinition.table files and regenerate your business object definitions. For details, see Chapter 5, "Generating business object definitions," on page 29.

Required fields for business objects

Business objects contain two sets of fields:

- A set of required data fields common to all, and
- An additional set of fields that varies according to the message type

The common fields are described in Table 9, and the additional fields in Table 10 and Table 11.

Note: The additional fields in these two tables are mutually exclusive. You would use one set or the other, but not both.

Table 9. Required fields for all business objects

Required field	Type	Description
Variant	String	Name of the variant that sent or will receive this message.
Partition	String	Name of the partition that sent or will receive this message.
TopicName	String	Uniquely identifies an event. Default key value when creating business object definitions.
ReplyKey	String	Uniquely identifies the message. Generated by the integration channel for asynchronous events only.

Table 10 lists the additional fields required for all business objects of **Request-Response** type only.

Table 10. Required fields for Request-Response business objects

Required field	Cardinality	Type	Description
RequestSchema	1	Specified by MergedSchemaName (if present) or SchemaName in MessageConfiguration.table.	Specifies request data types.
PullSchema_<n>	N	One PullSchema exists for each Signature.Pull.Schema type defined in MessageDefinition.table.	Specifies response data types.

Table 11 lists the additional fields required for all business objects of **Subscribe** type only.

Table 11. Required fields for Subscribe business objects

Required field	Cardinality	Description
SubscribeSchema	N	An attribute specifies the inbound object type for each schema defined in the MessageDefinition.table.

Business object verbs

Every business object contains a verb, which describes how the data in the application-specific business object should be handled by the receiving application. Ariba Buyer defines a set of operations that are similar to verbs, but that describe only how incoming data should be loaded into the Ariba Buyer database.

Verbs and operations are handled as follows:

- **For synchronous Request-Response events:**
  The Request application-specific business object verb is set to the value of the Verb parameter defined in the Channels section of MessageConfiguration.table. The verb describes the intended action at the destination application. The action that is performed on the data returned in the Response application-specific business object is the value of the Operation parameter defined in the Channels section of MessageConfiguration.table. This sequence is illustrated in Figure 2.

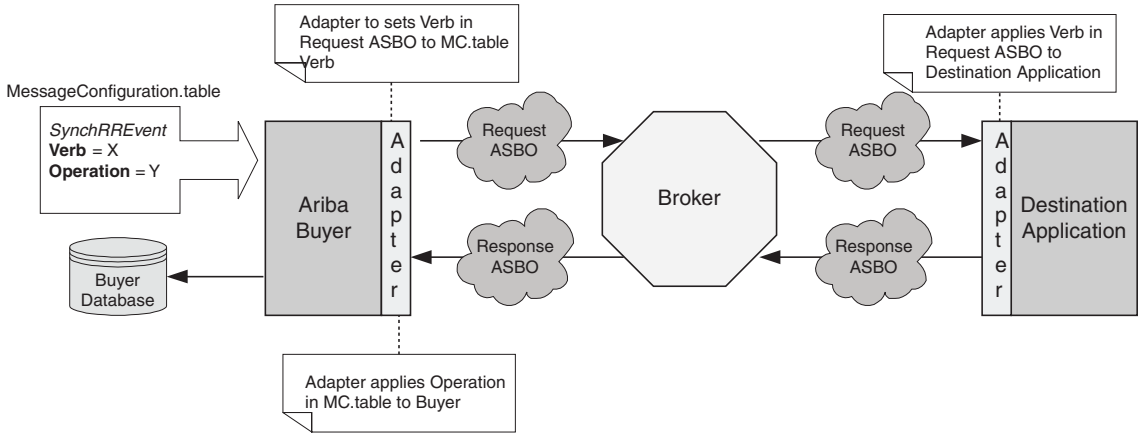

Figure 2. Synchronous Request-Response events

- **For asynchronous Request-Response events:**
  The Request application-specific business object verb is set to the value of the Verb parameter defined in the Channels section of MessageConfiguration.table. The verb describes the intended action at the destination application. The destination application sets the verb of the Response application-specific business object. The action that is performed on the data returned in the Response application-specific business object is determined by converting the verb in this application-specific business object to the operation value given in Table 12 on page 25. This sequence is illustrated in Figure 3.

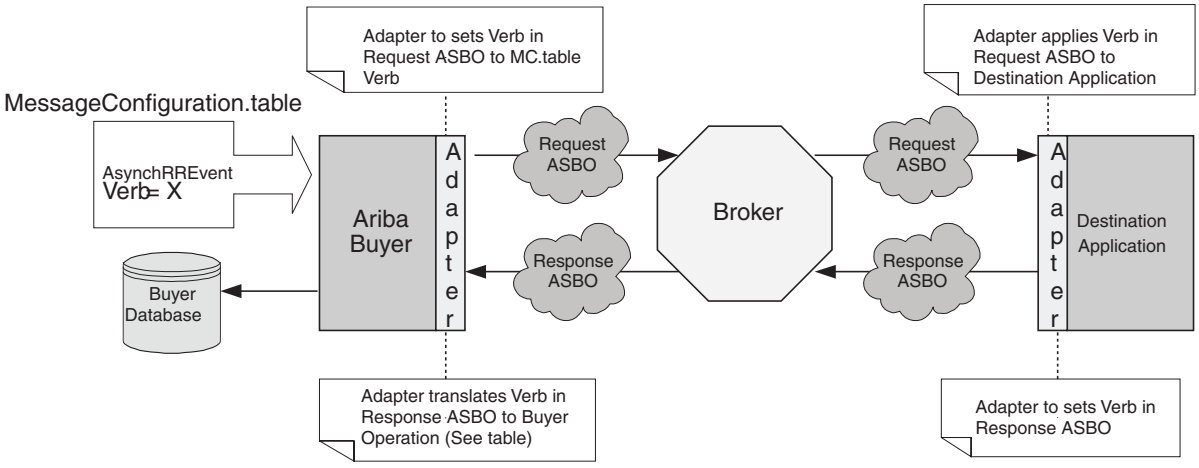

Figure 3. Asynchronous Request-Response events

- **For Subscribe events**:
  The source application sets the business object verb. The action that is performed

on the data received is determined by converting the business object verb to the operation value given in Table 12. This sequence is illustrated in Figure 4.

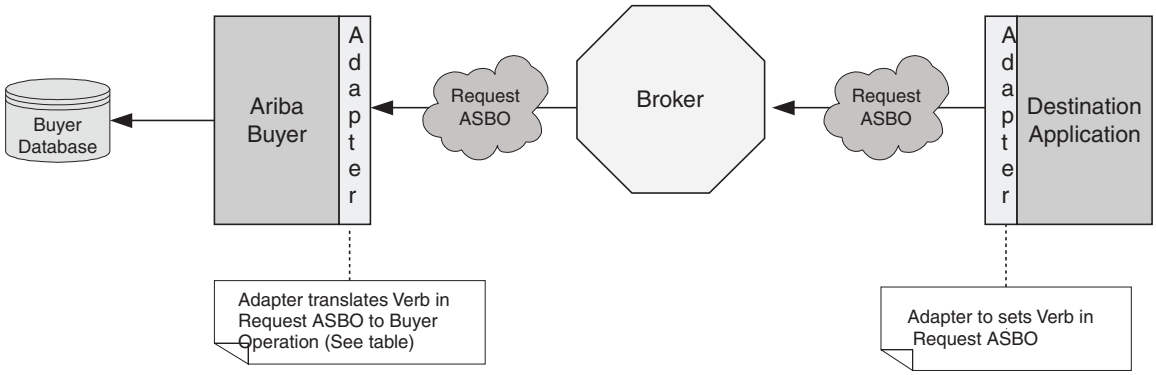

Figure 4. Subscribe events

Table 12 shows the correlation between incoming application-specific business object verbs and Ariba Buyer operations.

Table 12. Correlation between ASBO verbs and Ariba Buyer operations

Verb (WBI)	Operation (Ariba Buyer)	Action
Create	Create	Creates any new objects in Ariba Buyer.
Delete	Delete	Removes from Ariba Buyer any objects that are in the message.
Retrieve	N/A	Ariba Buyer cannot perform retrieves.
Update	"Load And Delete"	Updates new data values for existing objects, creates new objects, and deletes objects in Ariba Buyer that do not exist in the incoming message.
ABUpdate	Update	Updates new data values for existing objects. Does not create new objects or delete objects in Ariba Buyer that do not exist in the incoming message.
Load	Load	Updates new data values for existing objects and creates new objects.
Update_And_Delete	"Update And Delete"	Updates new data values for existing objects. If the object does not exist in the incoming message, it is deleted from Ariba Buyer.
Update_Elements_Only	"Update Elements Only"	Updates vector objects.

Attribute properties

The connector has various properties that you can set on its business object attributes. The tables below list the attributes and their values.

Table 13 lists simple attributes.

Table 13. Simple attributes

Attribute	Property
Name	Specifies the business object field name.
Type	Specifies the business object field type.

Table 13. Simple attributes (continued)

Attribute	Property
MaxLength	255 by default.
IsKey	Each business object must have at least one key attribute, which you specify by setting the key property to true for an attribute.
IsForeighKey	Not used.
Is Required	Not used.
AppSpecInfo	Holds the original Java type.
DefaultValue	Specifies a default value that the connector uses for a simple attribute in the inbound business object if the attribute is not set and is a required attribute.

Table 14 shows the properties for child object attributes.

Table 14. Child object attributes

Attribute	Description
Name	Specifies the business object field name.
Type	Specifies the business object field type.
Contained ObjectVersion	For all attributes that represent child business objects, this property specifies the child's business object version number.
Relationship	If the child is a container attribute, this is set to `Containment`.
IsKey	Set to true if the attribute name equals UniqueName, otherwise it is set to false.
IsForeighKey	Not used.
Is Required	Not used.
AppSpecInfo	Holds the original Ariba Buyer field name.
Cardinality	Set to N if the type represents an array or vector, otherwise set to 1.

Application-specific information

Application-specific information provides the connector with application-dependent instructions on how to process business objects. If you extend or modify a business object definition, you must make sure that the application-specific information in the definition matches the syntax that the connector expects.

Application-specific information can be specified on the business object and also on each business object attribute.

The application-specific information for the business object handles two key properties for the adapter for Ariba Buyer. They are:

- The original object type name
- The name of the collaboration used by the business object

Note: The system integrator must fill in the collaboration name. This value is not used when WebSphere MQ Integrator is the broker. However, IBM recommends that this value be set to match the name of the broker flow that will handle this transaction within WebSphere MQ Integrator Broker.

The application-specific information for the business object attributes holds the original Ariba Buyer-specific java type.

Generating business objects

The integration channel component of the adapter for Ariba Buyer runs in Ariba Buyer itself. Once events have been configured for the integration channel, Ariba Buyer exports application-specific metadata through the channel to a file system repository. This metadata is used by the Object Discovery Agent to create business object definitions, which are exported to the integration broker repository.

Each time an event occurs during run time, Ariba Buyer sends a message object containing object-level data and information about the type of transaction. The connector maps this data to the corresponding business object definition, to create an application-specific business object. The connector sends these business objects on to the integration broker for processing. It also receives business objects back from the integration broker, which it passes back to Ariba Buyer.

Note: If the object model in Ariba Buyer is changed, use the ODA to modify the original business object definition to reflect the update, or create a new definition. If the business object definitions in the integration broker repository do not match exactly the data that Ariba Buyer exports, the connector is not able to create a business object and the transaction will fail.

Business Object Designer provides a graphical interface that enables you to generate business object definitions for use at run time. For details, see Chapter 5, “Generating business object definitions,” on page 29

Chapter 5. Generating business object definitions

This chapter describes the Object Discovery Agent (ODA) for Ariba Buyer, and how to use it to generate business object definitions for the IBM WebSphere Business Integration Adapter for Ariba Buyer.

This chapter contains the following sections:

Overview of the ODA for Ariba Buyer

An ODA (Object Discovery Agent) enables you to generate application-specific business object definitions. A business object definition is a template for a business object. The ODA examines specified application objects, "discovers" the elements of those objects that correspond to business object attributes, and generates business object definitions to represent the information. Business Object Designer provides a graphical interface to access the Object Discovery Agent and to work with it interactively.

The Object Discovery Agent (ODA) for Ariba Buyer generates business object definitions from metadata exported by Ariba Buyer at initialization. The Business Object Designer wizard automates the process of creating these definitions. You can view or make modifications to a business object definition before you save it to the server.

You use the ODA to generate business object definitions at two different stages:

1. **When the system is first set up:** You must create a set of business object definitions for all integration events defined on the integration channel. Business object definitions and broker collaborations and workflows must be defined for all integration channel events that will be run as part of the database initialization.
2. **Whenever an object used in an integration channel event is modified in Ariba Buyer:** You must run the ODA in Business Object Designer to pick up the changed metadata for the object and export a new business object definition to the broker repository.

Generating business object definitions

This section describes how to use Ariba Buyer ODA in Business Object Designer to generate business object definitions. For information on launching and using Business Object Designer, see *IBM WebSphere Business Integration Adapters Business Object Development Guide.*

Starting the ODA

The ODA can be run from any machine that can mount the file system on which the metadata repository resides, using the start_BuyerODA.bat (WIN) or start_BuyerODA.sh (UNIX) start file. This file contains start parameters, including the paths to certain required Ariba Buyer and connector .jar files. These .jar files must also be accessible from the machine on which you are running the ODA.

The ODA for Ariba Buyer has a default name of BuyerODA. The name can be changed by changing the value of the AGENTNAME variable in the start script.

To start the ODA, run this command:

```
start_BuyerODA <Ariba server root path>
```

For example,

```
start_BuyerODA C:\Ariba\app\Buyer\Server
```

Running Business Object Designer

Business Object Designer provides a wizard that guides you through the steps to generate a business object definition using the ODA. The steps are as follows:

Select the agent

1. Start Business Object Designer.
2. Select **File > New Using ODA**. The *Business Object Wizard - Step 1 of 6 - Select Agent* screen appears.
3. Select the ODA/AGENTNAME (from the start_BuyerODA script) in the Located agents list and click **Next**.

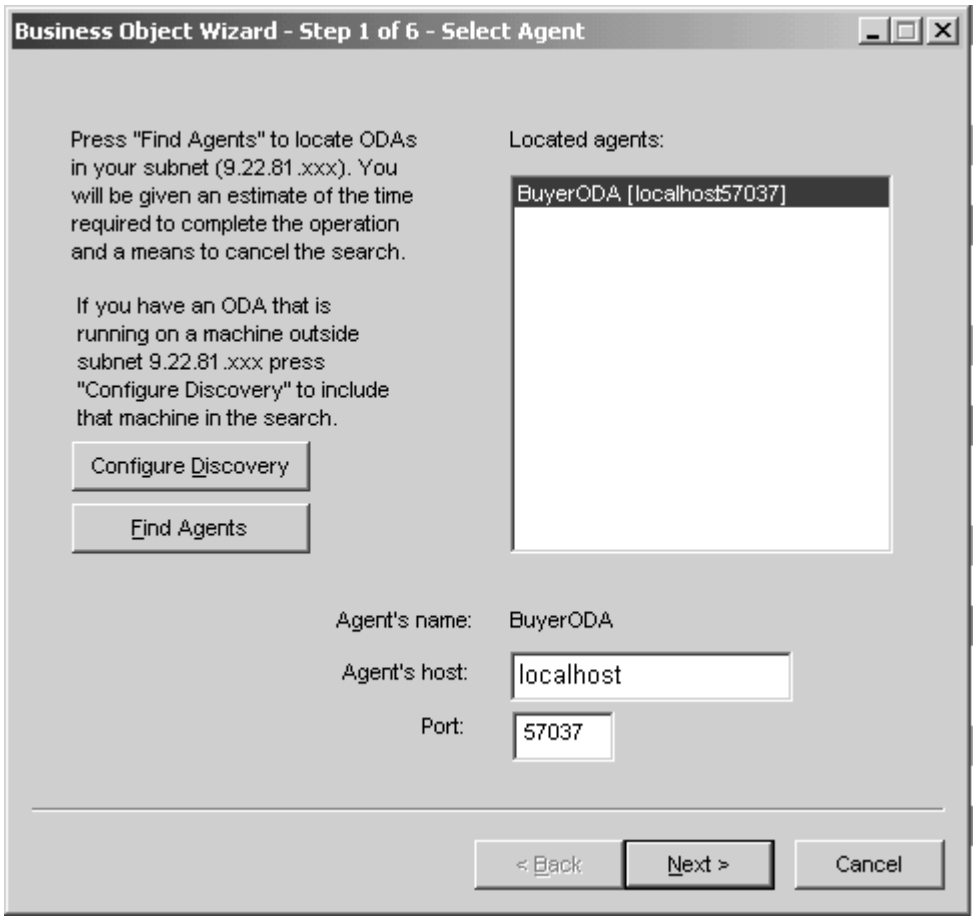

Figure 5. Select Agent screen

Configure the agent

The *Business Object Wizard - Step 2 of 6 - Configure Agent* screen appears. The first time you open this screen, it is blank.

In this screen you:

- Enter the path information that the ODA needs to locate the metadata repository and the message file.
- Specify a trace message file; for example, `BuyerODAtrace.txt`.
- Specify a tracing level (from 0 to 5).

Note: If the name of the message file is not correctly specified, the ODA will run without messages.

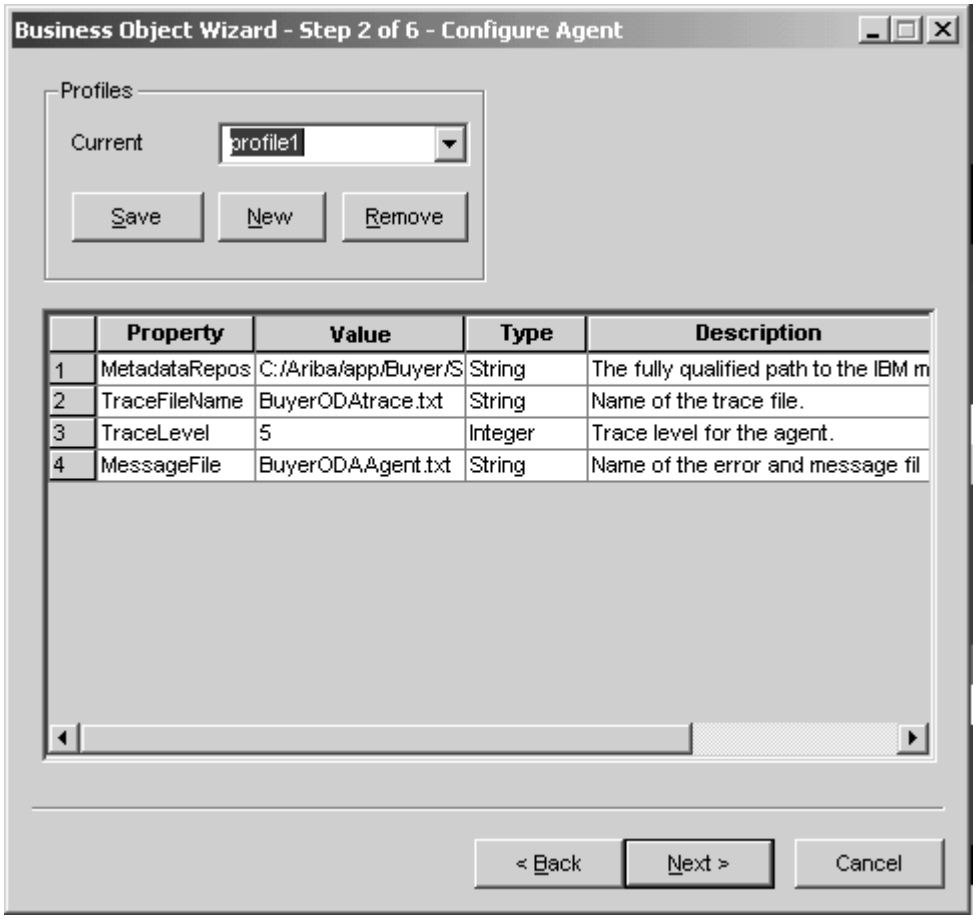

Figure 6. Configure Agent screen

All messages displayed by the Buyer ODA appear in the `BuyerODAAgent.txt` file, which has a standard message file format. For more information on the range of trace levels and their meaning, refer to "Using log and trace files" on page 18.

You can also can save all the values you enter on this screen to a profile. Instead of re-typing all the data the next time you run the ODA, you simply select a profile from the drop-down menu and use the saved values.

You can save multiple profiles, each with a different set of specified values.

1. Use the Profiles menu the first time you run the ODA to create a new profile. In successive uses, you may select an existing profile.
2. Type the name of each property, its value, type and description.

   Note: If you use a profile, the property values are filled in for you. Table 15 lists the properties and their possible values.

Table 15. Property values for configuring the ODA

No.	Property name	Property type	Description	Required
1	MetadataRepos	String	Directory where the metadata file repository resides. For example, C:\temp\metadata	Yes
2	TraceFileName	String	The name of the trace file.	No
3	TraceLevel	Integer	Trace level for the ODA.	Yes
4	MessageFile	String	The path to the message file.	Yes

Note: The metadata repository is always in the location specified for the MetadataRepos property, found in: `<Ariba_server_root>/config/Parameters.table`. You can set MetadataRepos to this value, or copy the `\metadata` directory to another machine and reset the MetadataRepos value accordingly. This is possible because there is no interaction between the ODA and the application during the process of generating business object definitions.

When you have finished, click **Next**.

Select a business object

The *Business Object Wizard - Step 3 of 6 - Select Source* screen appears. Use this screen to select any number of MessageConfigurations for which the ODA will generate business object definitions.

The screen lists the variants that have been defined in Ariba Buyer (derived from the structure of the metadata repository). If Ariba Buyer has not been initialized, the list will be empty.

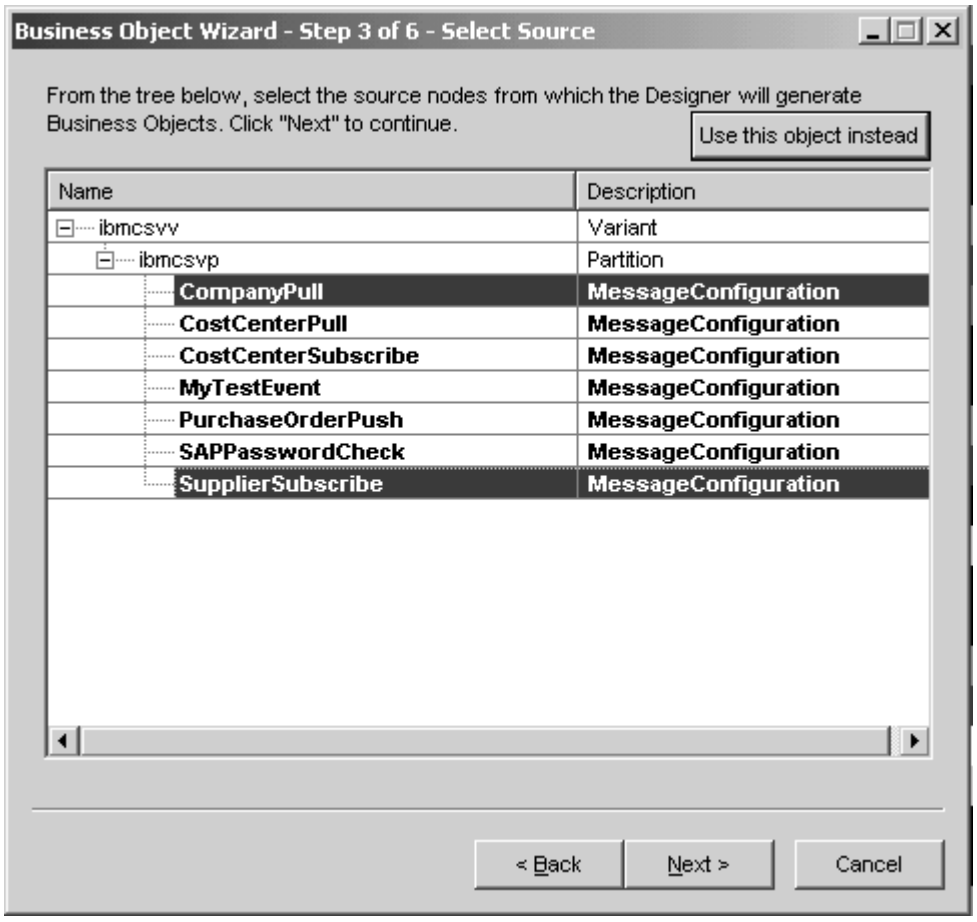

Figure 7. Select Source screen

- Expand a variant to see a list of its partitions.
- Expand a partition to see a list of MessageConfigurations.

1. Select the MessageConfigurations you want to use.
2. Click **Next**.

Confirm selection

The *Business Object Wizard - Step 4 of 6 - Confirm source nodes for business object definitions* screen appears. It shows the MessageConfigurations you selected.

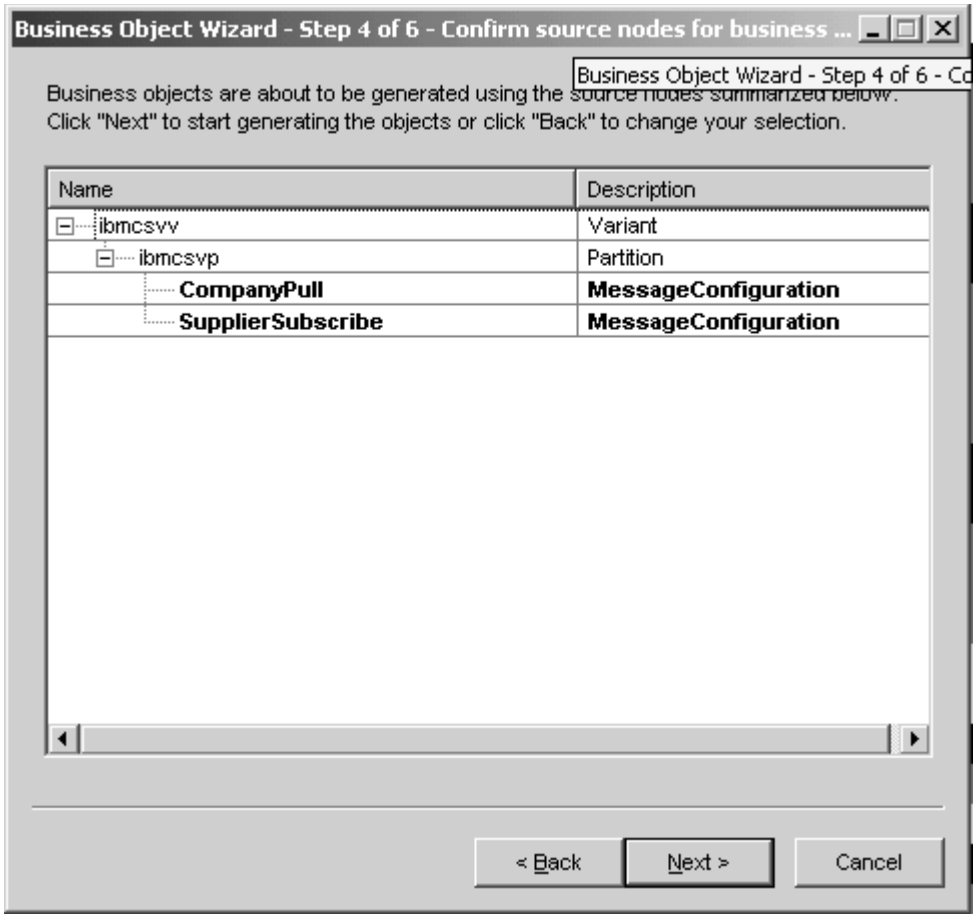

Figure 8. Confirm source nodes screen

Click **Back** to make changes or **Next** to confirm the list is correct.

Generate the business object

The *Business Object Wizard - Step 5 of 6 - Generating business objects...* screen appears.

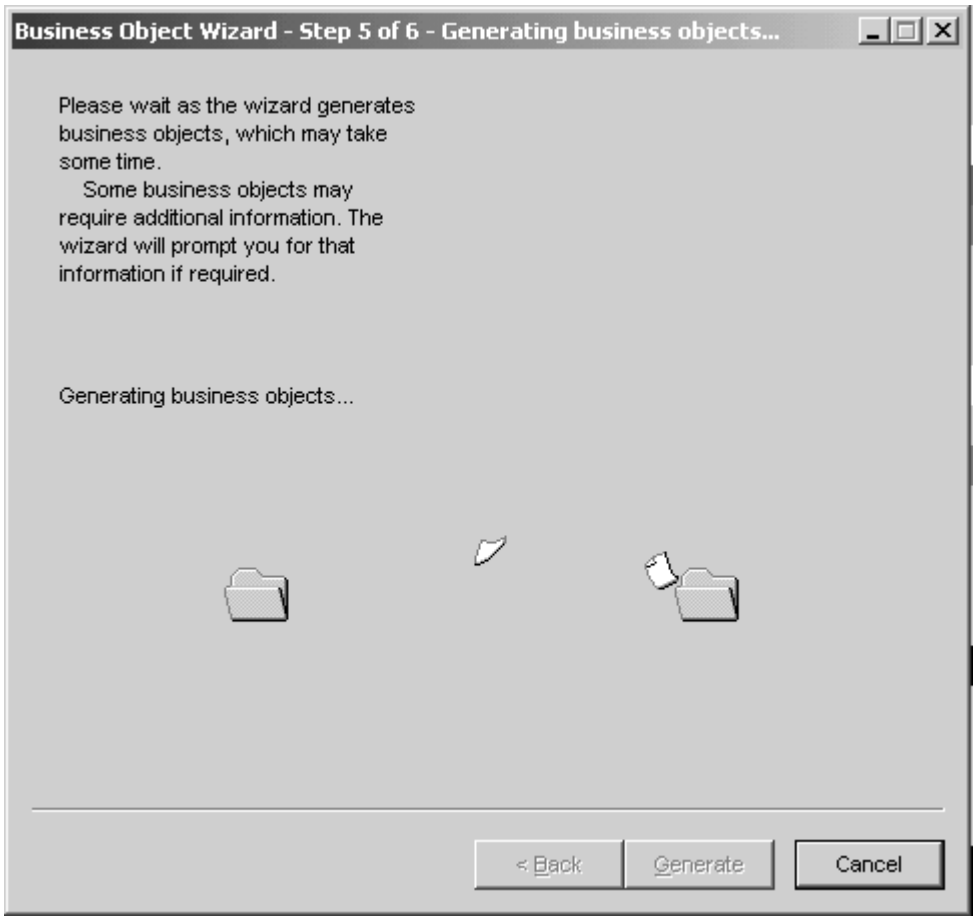

Figure 9. Generating business objects screen

The message states that the wizard is generating the business objects.

Save the a copy of the business object to a separate file

The *Business Object Wizard - Step 6 of 6 - Save business objects* screen appears.

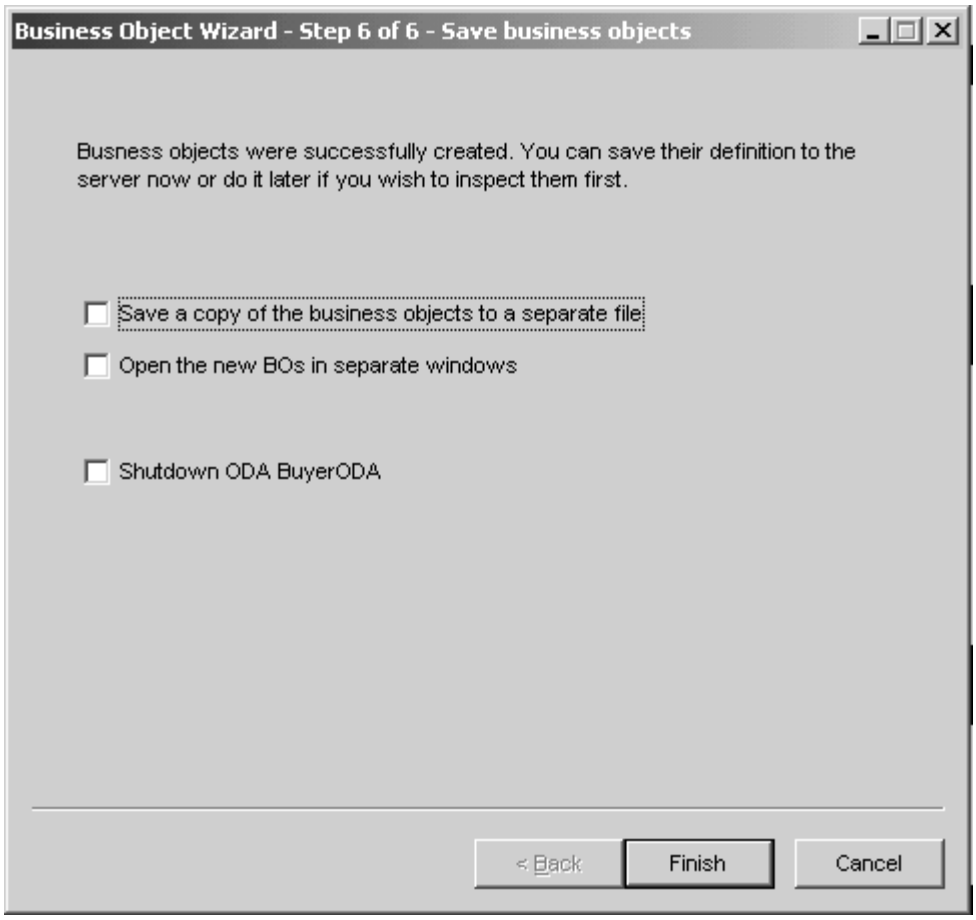

Figure 10. Save business objects screen

You can optionally save the generated business object definitions to a file. To do so:

1. Check **Save business objects to a file**. A dialog box appears.
2. Type the location in which you want the copy of the new business object definitions to be saved.

Business Object Designer saves the files to the specified location.

Note: If you create new custom objects in Ariba Buyer, you must add new business object definitions for them, using the ODA as described in this chapter.

If you have finished working with the ODA, you can shut it down by checking "Shutdown ODA Buyer ODA" before clicking **Finish**.

Uploading your files

The newly created business object definition files must be uploaded to the integration broker once they have been created. The process depends on whether you are running ICS, WebSphere MQ Integrator Broker, or WAS as your integration broker.

For ICS: If you have saved your business object definition files to a local machine and need to upload them to the repository on the server, refer to the *Implementation Guide for WebSphere InterChange Server*.

For WebSphere MQ Integrator Broker: You must export the business object definitions out of Business Object Designer and into the integration broker. For details, refer to *Implementing Adapters with WebSphere MQ Integrator Broker.*

For WAS: You must export the business object definitions out of Business Object Designer and into the integration broker. For details, refer to *Implementing Adapters with WebSphere Application Server.*

Chapter 6. Troubleshooting and error handling

This chapter describes how the adapter for Ariba Buyer handles errors. The three components of the adapter generate logging and tracing messages. The chapter describes these messages by component The chapter contains the following sections:

Channel error handling

The integration channel logs error messages using the Ariba Buyer API. It logs any abnormal condition that it encounters during processing, regardless of the trace level. It writes the text to an Ariba Buyer log file located in `<Ariba server root>\logs` and named AribaBuyer<nodename>Log.txt; for example, AribaBuyerNode1Log.txt.

The message contains a detailed description of the condition and the outcome and may also include extra information that may aid in debugging, such as business object dumps or stack traces (for exceptions).

Channel error messages are of two types:

- Initialization errors
- Run-time errors

Channel initialization errors are usually fatal, and will prevent the Ariba Buyer node from starting. Table 16 lists possible errors and the action you can take to remedy each one.

Table 16. Channel initialization errors

Error message	Description	Corrective action
Error accessing RMI registry on port xxx	Fatal error. If the ChannelRemote cannot be started, the connector is not able to communicate with the channel.	Verify the ChannelPort parameter specified in Parameters.table and change it if necessary.
Error binding ChannelRemote object at registry: <url> Exception: <message text>	Fatal error. If the ChannelRemote cannot be started, the connector is not able to communicate with the channel.	Verify the ChannelPort parameter specified in Parameters.table. See the Exception message for more information.
Error initializing ChannelRemote object. Exception: <xxx>	An error occurred when starting the integration channel's remote object.	Verify the ChannelPort parameter specified in Parameters.table. See the Exception message for more information.
Error creating file: xxx	The channel receives an error when attempting to create the specified directory or file.	Verify that: • There is enough disk space to store files. • The user has the authority to create files and directories.

Table 17 lists possible run-time errors and the action you can take to remedy each one.

Table 17. Channel run-time errors

Error message	Description	Corrective action
Exception processing Message: <topic name> Exception <error text>	The message handler receives an error when trying to send a message to the connector.	See the error text for more information.
The specified reply key was not found in the table. Key = xxx	The message handler receives a message with a key not found in the reply table.	Check log files to see if specified reply key was found and processed on another node. If not, rerun the event. If a node has been stopped or shut down, the reply key will be invalid.
Received Exception connecting to Connector remote object. Exception <error text>	Message handler's reference to the connector is missing or invalid.	Verify that: • The ConnectorPort and ConnectorURL parameters are correct in Parameters.table. • The Ariba Buyer connector has been started.
The xxx parameter is not specified.	The specified parameter is not found in the message configuration or message definition.	Edit the message configuration or message definition accordingly.
xxx has a null value	A parameter in the reply received from the connector has a null value.	Verify that this parameter is being set correctly by the map, adapter, or collaboration.
Message Configuration not found for <topic name> in partition: <partition name>	A message configuration entry cannot be found for the specified topic name in the specified partition.	Verify that the specified partition exists in MessageConfiguration.table in the integration channel directory.
Channel parameter <xxx> not set	The specified parameter has not been set in the integration channel section of the Parameters.table file.	Add the parameter to Parameters.table with the correct value.
Processing for the Message with TopicName <xxx> has exceeded the timeout interval specified	A reply for the message with the specified topic name has not been received within the allotted time.	Verify that the Timeout parameter for the specified topic name has been set correctly in MessageConfiguration.txt. Check the integration channel and connector log files for related error messages.
ABRequestMessage Handler not found in table for: <partition - topic name>	A message handler cannot be found for a message with the specified topic name and partition.	Add an entry to MessageConfiguration.table for the specified topic name and partition.
Reply for Message with TopicName <xxx> has no data for defined PullSchema objects	The integration channel has received a response with missing data for the specified message.	Verify that: • The PullSchema(s) are defined in MessageDefinition.table. • The PullSchema parameter is being set correctly by the map, collaboration or adapter.

Connector error handling

The connector logs any abnormal condition that it encounters during processing, regardless of the trace level. It writes the text to the connector log file.

The message contains a detailed description of the condition and the outcome and may also include extra information that may aid in debugging, such as business object dumps or stack traces (for exceptions).

Connector error messages are of two types:

- Initialization errors
- Run-time errors

Table 18 lists possible initialization errors and the action you can take to remedy each one.

Table 18. Connector initialization errors

Error message	Description	Corrective action
An Exception occurred attempting to bind the ABConnectorRemote object to the RMI Registry using name <name> Exception: <error text>	Fatal error. If the ConnectorRemote cannot be started, the channel is not able to communicate with the connector.	Set the ConnectorPort property correctly in the connector application configuration properties. See the error text for more information.
Could not connect to the remote object of the ABChannel	The connector cannot connect to any channel nodes at initialization.	Verify that: • The Ariba Buyer server is started. • The connector application configuration property Nodes is set correctly.
Error connecting to remote object <node name>, using URL: <url>	The connector cannot connect to the channel's remote object on the specified Ariba Buyer node.	Set the Nodes property correctly in the connector application configuration properties.
Connector property not specified: <name>	The specified property is not set.	Set the named property correctly in the connector application configuration properties.

Table 19 lists possible run-time errors and the action you can take to remedy each one.

Table 19. Connector run-time errors

Error message	Description	Corrective action
Collaboration name not found in Business Object: <name>	The connector attempts to send a business object that does not have the collaboration application-specific property set.	Set the Collaboration property correctly in the business object-level application-specific information in the specified business object definition.
Connector failed to deliver business object: <BO name>	The gotApplEvent returns an error.	Check the log files for more information.

Table 19. Connector run-time errors (continued)

Error message	Description	Corrective action
Failed to deliver business object: <BO name> because the connector is not active.	The gotApplEvent returns an error. Attempted to deliver a business object to an inactive connector.	Verify that the connector is active.
Connector failed to deliver business object: <BO name> because no subscription was found.	The gotApplEvent returns an error.	Verify that the business object is subscribed to by a collaboration.
Invalid verb specified: <verb> for business object: <BO name>	The verb in a message sent to the connector is not supported by the business object.	Verify that: • The Verb parameter is set correctly in MessageConfiguration.table. • All supported verbs are correctly set in the business object definition.
The attribute <xxx> was not found in the Business Object <BO name>	The connector is processing a message containing a field that does not have a corresponding attribute in the business object definition.	Verify that the business object definition is correctly set up.
The Buyer Connector does not poll. Set the Connector property PollFrequency to "No" to avoid this message.	The Ariba Buyer connector does not poll.	Set the connector standard property PollFrequency to No.
The business object <name> is not defined	A business object definition with the specified name cannot be found.	Create the specified business object using the ODA.
The value specified for attribute <name> is not a valid type	The specified attribute in the message object does not have the same type as the attributes in the business object definition.	Verify the business object definition and recreate it using the ODA if necessary.
The attribute: <name> was not found in the business object definition: <BO name>	The attribute specified in the message object is not in the specified business object definition.	Verify the business object definition and recreate it using the ODA if necessary.
An error occurred during the processing of the business object: <BO name>. Exception: <error text>	An error occurred after the connector sent the specified business object.	Check the error text for more information.
An exception occurred unbinding the remote object from the RMI Registry.Name <name> Exception: <error text>	There was a problem unbinding the connector's remote object from the registry. This may prevent the connector from binding its remote object to the RMI registry the next time the connector is started.	Verify the ConnectorPort property in the connector application configuration properties. Check the error text for more information. You may have to restart the system before attempting to start the connector.

Table 19. Connector run-time errors (continued)

Error message	Description	Corrective action
A SpecNameNotFound exception occurred when adding the child business object: <child BO name> to the business object: <parent BO name>	The connector is attempting to add the specified child business object to an n-cardinality attribute of the specified parent business object.	Verify that the cardinality is correctly set for the parent business object.
Exception occurred instantiating class for type: <name>. Exception: <error text>	The class specified in the attributes App Spec Info could not be instantiated.	Verify that the specified class is in the classpath. See the error text for more information.
No String constructor found for type: <class name>	The class specified in the attributes App Spec Info does not have a constructor that accepts one string parameter.	The class specified in the metadata cannot be used if it does not have a string constructor.
An error occurred on sending object to Channel on node: <name>. Exception: <error text>	The specified error occurred on sending an object to the channel.	See the error text for more information. Check the Ariba Buyer logs for more information.
Reply key: <key> for object not found in Channel on Node: <node name>	The connector has sent an object to the channel with a reply key that the channel does not have in its table.	The reply key may be invalid, or may exist on another node. Check the logs to see if another node has processed the object. The request may have timed out and the reply key has been removed from the table. Check the logs to verify.
An error occurred on sending the business object: <BO name> to Channel. Check the logs for more information.	An error occurred while sending the specified business object to the channel.	Check the logs for more information.
An error occurred on creating a java.util.Date object from the String "xxx". Locale: <locale>	The Date string does not match the default format or the formats specified in the DateFormats parameter in the connector application configuration properties.	Specify the correct format string in the connector application configuration properties. See the API for java.text.SimpleDateFormat for information on specifying date format strings.
The business object for TopicName: <topic name>, variant: <variant name>, partition: <partition name> is not defined	A matching business object definition could not be found for the specified topic name, variant and partition.	Create a matching business object definition using the ODA. Specify a business object prefix in the VariantMappings connector application configuration property.
Variant prefix not specified for variant: <variant name>	A prefix for the specified variant could not be found in the VariantMappings connector application configuration property.	Specify a business object prefix in the VariantMappings connector application configuration property.

ODA error handling

The ODA logs error messages to the ODA log. It logs any abnormal condition that it encounters during processing, regardless of the trace level.

The message contains a detailed description of the condition and the outcome and may also include extra information that may aid in debugging, such as business object dumps or stack traces (for exceptions).

Table 20 lists possible errors and the action you can take to remedy each one.

Table 20. ODA errors

Error message	Description	Corrective action
The specified path does not exist: <path name>	The connector ODA property BuyerPath is not set or is invalid.	Verify that the BuyerPath property is set correctly.
File not found: <file name>	The specified metadata file could not be found.	Verify that the BuyerPath property is set correctly. Regenerate the metadata file from Ariba Buyer.
Class not found for file: <file name	The ABClassMeta class cannot be found, or the metadata file was created with a backlevel ABClassMeta class	Verify that the ABClassMeta class is in the class path. Verify that the metadata was generated with the latest ABClassMeta class.
IO Error reading file: <file name>	The ODA receives an error when reading a metadata file.	Verify that the user has the authority to open the file. Verify that the file is not in use.
The ABClassMeta is null for class name: <class name> variant name: <variant name>	A metadata file could not be found for the specified class and variant.	Regenerate the metadata file from Ariba Buyer.
Error creating BusObjAttr for <field name> in class <class name>	An error occurred when adding the specified attribute to the business object definition for the specified class.	Verify that: • The field type is supported by the business object. • There is no duplicate attribute name in the business object.
Error creating Business Object definition: <BO name>. Error Text: <error text>	The ODA receives an error when creating a business object definition.	See the error text for more information.

Tracing messages

Tracing is an optional debugging feature you can turn on to closely follow a connector's behavior. Tracing messages are configurable and can be changed dynamically. You set various levels depending on the desired detail. Trace messages, by default, are written to STDOUT (screen). You can also configure tracing to write to a file.

Recommendation: Tracing should be turned off on a production system or set to the lowest possible level to improve performance and decrease file size.

Table 21 describes the types of tracing messages that the Ariba Buyer connector outputs at each trace level. All the trace messages appear in the file specified by the connector property TraceFileName. These messages are in addition to any tracing messages output by the IBM WebSphere Business Integration Adapter architecture.

Table 21. Tracing messages

Tracing Level	Tracing Messages
Level 0	Message that identifies the connector version. No other tracing is done at this level. This message is always displayed; for example, `'2003/03/21 15:01:46.812: This is version 3.0 of the Adapter for Ariba Buyer'.`
Level 1	Messages delivered each time the `pollForEvents` method is executed.
Level 2	• Messages logged each time a business object is posted to the integration broker from gotApplEvent. • Messages that indicate each time a business object request is received.
Level 3	Not applicable.
Level 4	• Application-specific information messages; for example, messages showing the values returned by the functions that parse the business object's application-specific information fields. • Messages that identify when the connector enters or exits a function, which helps trace the process flow of the connector.
Level 5	• Messages that indicate connector initialization; for example, messages showing the value of each configuration property retrieved from the integration broker. • Messages that comprise a business object dump. At this trace level, the connector outputs a textual representation of the business object before it begins processing the object (showing the object the connector receives from the collaboration), and after it has finished processing the object (showing the object that the connector returns to the collaboration).

Tips for troubleshooting

Use the following tips for troubleshooting problems:

- If the channel is running remotely, try to ping the remote machine and also ping this machine from the remote machine.
- Check that the connector is running.
- Check that the server is running.
- Be sure the business object structure is consistent with the operation.

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
- RHF2MessageDomain

Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties
- Connector-specific configuration properties

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

Using Connector Configurator

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

Note: Connector Configurator and System Manager run only on the Windows system. If you are running the connector on a UNIX system, you must have

a Windows machine with these tools installed. To set connector properties for a connector that runs on UNIX, you must start up System Manager on the Windows machine, connect to the UNIX integration broker, and bring up Connector Configurator for the connector.

Setting and updating property values

The default length of a property field is 255 characters.

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

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

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

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

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

Summary of standard properties

Table 22 on page 49 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker, 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 22. 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 22. 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 22. 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	
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

Table 22. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
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 `true`to 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.

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 53.

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 64).

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

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

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

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

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

Starting Connector Configurator

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

- Independently, in stand-alone mode
- From System Manager

Running Configurator in stand-alone mode

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

To do so:

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

You may choose to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see "Completing a configuration file" on page 69.)

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 65.
- To use an existing file, simply modify an existing template and save it under the new name.

Creating a new template

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

To create a template:

1. Click **File>New>Connector-Specific Property Template**.
2. The **Connector-Specific Property Template** dialog box appears, with the following fields:
   - **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 72..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

Setting the configuration file properties

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

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

- Standard Properties
- Connector-specific Properties
- Supported Business Objects
- Trace/Log File values
- Data 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 48.

Specifying supported business object definitions

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

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

If ICS is your broker

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

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

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

To delete a business object from the supported list:

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

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

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

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

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

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

For most connectors, `Best Effort` is the only possible choice.

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

If a WebSphere Message Broker is your broker

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

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

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

If WAS is your broker

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

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

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

Associated maps (ICS only)

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

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

will be used to transform the application-specific business object to the generic business object or the generic business object to the application-specific business object.

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

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

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

- **Business Object Name**

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

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

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

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

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

  To explicitly bind a map:

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

Resources (ICS)

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

Messaging (ICS)

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

Setting trace/log file values

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

To change the logging and tracing values:

1. Click the **Trace/Log Files** tab.
2. For either logging or tracing, you can choose to write messages to one or both of the following:
   - To console (STDOUT):
     Writes logging or tracing messages to the STDOUT display.

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

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

Data handlers

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

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

Saving your configuration file

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

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

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

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

- For ICS: *Implementation Guide for WebSphere InterChange Server*
- For WebSphere Message Brokers: *Implementing Adapters with WebSphere Message Brokers*
- For WAS: *Implementing Adapters with WebSphere Application Server*

Changing a configuration file

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

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

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

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

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

Appendix C. Migrating the integration channel

This appendix describes how to migrate the integration channel from Ariba Buyer 7.1a to Ariba Buyer 8.1.

If you are upgrading from Ariba Buyer 8.0 to Ariba Buyer 8.1, simply migrate the Buyer application according to the Ariba Buyer documentation. You do not have to migrate the integration channel.

Overview

Migrating from Ariba Buyer 7.1a to Ariba Buyer 8.1 is a two-fold process. You need to:

1. Migrate the Ariba Buyer application from version 7.1a to version 8.1. For instructions on performing this migration, refer to the *Ariba Buyer Migration Guide Version 8.1*.
2. Migrate the integration channel from the earlier version of the adapter to the current version.

Migration process

This appendix details the steps you take to migrate the integration channel to the WebSphere Business Integration Adapter for Ariba Buyer. The steps are divided into the following phases:

- Installing the adapter
- Migrating Ariba Buyer
- Migrating the integration channel

Installing the WebSphere Business Integration adapter for Ariba Buyer

Before you install this version of the adapter, IBM recommends that you back up and uninstall the previous version. This section describes how to:

- Uninstall the previous version (WAAB).
- Install WebSphere Business Integration Adapter for Ariba Buyer.
- Install the WebSphere Business Integration Adapter migration utility.

Uninstalling WAAB

IBM recommends that you back up your WAAB instance by copying the entire adapter installation directory to an archive location before you uninstall the adapter.

After you have backed up the adapter installation directory, follow the instructions for uninstalling the adapter that are given in the *WebSphere Adapter for Ariba Buyer User Guide*.

Note: It is important that you remember the location of your MQCSVMapping.xml file(s), as you will need to manually migrate any customized CSV mappings from this file to the new Ariba IntegrationMappings.aml file later in the migration process.

Installing WebSphere Business Integration Adapter for Ariba Buyer

To install the new adapter, refer to Chapter 2, “Installing the adapter,” on page 7of this guide.

Note: Stop when you reach the section“Broker compatibility” on page 7. You do not need to run the Ariba Buyer Configure tool for the migration.

Configuring the adapter migration utility

IBM provides a utility to assist customers migrating from WAAB to WebSphere Business Integration Adapter for Ariba Buyer. The migration utility is included as part of the adapter installation.

To configure the migration utility:

1. Modify the *<Ariba Buyer 8 Server dir>*/classes/classpath.txt file by appending this line to it:

   ```
   classes/BIA_IBMMigration.jar
   ```

2. Create the following directory in your Buyer 8.1 installation directory:

   ```
   \Server\channels\MQSeries
   ```

   You may delete this directory after running the migration task harness successfully.

The utility is now installed. You will run it as one of the integration channel migration tasks described below.

Table 23 lists the files included in the migration utility package and provides a brief description of each one.

Table 23. Contents of migration utility package

Subdirectory	Description
\channels\IBM\migration\ WBIAABMigrationTaskList.xml	File that describes the list of tasks to be run as part of the adapter migration. This file is used as input to the migration harness.
\channels\IBM\migration\ UpdateIBMParameters.txt	File that describes the changes to be performed on the Parameters.table file.
\channels\IBM\migration\ loaddb_removelist.txt	List of integration events that must be removed from the config\LoadDB.txt file. These events are not required by the new adapter.
\classes\BIA_IBMMigration.jar	File that contains all the code required to migrate the adapter.

Migrating Ariba Buyer

Perform the migration to Ariba Buyer 8.1 according to the instructions given in the *Ariba Buyer Migration Guide Version 8.1.*

After launching the migrationHarness utility, run tasks 0 through 26 only. Do this by issuing the command:

```
rr 0-26
```

The remaining Ariba Buyer integration tasks will be done after you have run the integration channel migration utility (see “Completing the Buyer migration tasks” on page 83).

Migrating the integration channel

After you have installed the adapter and Ariba Buyer has been successfully migrated, you must perform a series of integration channel migration tasks. These tasks include:

1. Running the adapter migration utility
2. Migrating integration events
3. Configuring the adapter and integration channel
4. Generating metadata
5. Creating object mappings

Running the migration utility

The adapter migration utility carries out several integration channel migration modifications in the following files:

- the <*Ariba Buyer 8 Server*>\config\Parameters.table file
- the <*Ariba Buyer 8 Server*>\config\LoadDB.txt file
- the <*Ariba Buyer 8 Server*>\...\<*variantName*>\... \<*partitionName*>\IBM\MessageConfiguration.table file

Run the utility

To run the utility, execute this command from the <*Ariba Buyer8 Server*>\bin directory:

```
migrationHarness -tasklist <Ariba Buyer 8 Server dir>
\channels\IBM\migration\WBIAABMigrationTaskList.xml
```

For example:

```
migrationHarness -tasklist
C:\Ariba\app\Buyer8\Server\channels\IBM\migration\WBIAABMigrationTaskList.xml
```

The actions performed by the migration utility are described below.

Create backups

The utility creates a backup of all the files that it deletes or modifies on the <*Ariba Buyer 8 Server*>\config directory. The backup is located at <*Ariba Buyer 8 Server*>\channels\IBM\migration\backups\backup<timestamp>.

Update the Parameters.table file

The utility performs these tasks on the Parameters.table file:

- Replaces the System.Messaging.Channels.MQSeries section with a System.Messaging.Channels.IBM section.
- Adds entries for each non-CSV partition to Partitions.<*Partition Name*>.Application.Messaging.MessageConfigurationFiles.IBM.
- Modifies the logging category name specified by System.Logging.Console.Categories and System.Logging.MainLogFile.Categories from “integrationVendor_IBM” to “integrationChannel_IBM”.

Update the LoadDB.txt file

The utility performs these tasks on the LoadDB.txt file:

- Removes all references to None.IntegrationEvent.MQMetaDataExport.
- Removes all references to None.IntegrationEvent.MQSchemaDataExport.
- Removes all references to None.IntegrationEvent.MQXMLDataExport.

Update the MessageConfiguration.table files

The utility performs these tasks on the MessageConfiguration.table files:

- Creates a file called IBMMessageConfiguration.table in \config\variants\<*variant name*>\partitions\<*partition name*>\file. This file contains CSV events that need to be migrated to the file channel. The process is described in "Transferring CSV events to the File Channel" on page 83.
- Changes the <*event name*>.Channel.Name parameter from "MQSeries" to "IBM".
- Changes the <*event name*>.Channel.Action parameter to <*event name*>.Channel.Operation.

  Note: LoadAndDelete will become "Load And Delete" and UpdateAndDelete becomes "Update And Delete".
- If the event is a data push or a data pull, the utility creates a new parameter for <*event name*>.Channel.Verb and assigns it as follows:
  - For data push, assign <*event name*>.Channel.Verb = Create
  - For data pull, assign <*event name*>.Channel.Verb = Retrieve

Table 24 summarizes the changes that the migration utility makes to the MessageConfiguration files.

Table 24. Values for Channel section of ERP integration events

Parameter tag	Old value	New value
Name	MQSeries	IBM
Operation	Parameter tag did not exist	Assigned the value of the Action parameter.
Verb	Parameter tag did not exist	If event is a data push, new value is Create. If event is a data pull, new value is Retrieve. If Subscribe, do not create parameter.
Timeout	Parameter tag did not exist	If event is an asynchronous data push, set the value to the number of milliseconds to wait before returning a timeout.

Changes to the MessageConfiguration.table files for Subscribe events

All Subscribe events should be defined in MessageConfiguration.table events at the Variant level. For example:

<*Ariba Buyer 8 Server*>\config\
variant\<*Variantname*>\IBM\MessageConfiguration.table.

For each Subscribe event, you will need to manually create a new channel parameter for EventSource and provide a value for the parameter (see the example below).

```
CostCenterSubscribe = {
    Channel = {
                EventSource = "myPart1_CostCenterSubscribe"
```

```
    }
    LoggingName = CostCenter;
        TopicName = CostCenterSubscribe;
};
```

For a detailed description of the EventSource parameter and its usage, refer to the *Ariba Configuration Reference Guide*.

Completing the Buyer migration tasks

Once the integration channel migration utility has successfully finished its tasks, run the migration harness using the migrationHarness command. Then complete the remaining Ariba Buyer migration tasks by issuing the command:

```
rr 27-29
```

Migrating integration events

The files that define integration events in Ariba Buyer 8.x differ from their counterparts in Ariba Buyer 7.x (see Table 25).

Table 25. Integration event definition files

Ariba Buyer version	Integration events defined in...
Ariba Buyer 7.x	• IntegrationEvents.table • IntegrationSchema.table
Ariba Buyer 8.x	• MessageConfiguration.table There can be multiple tables per partition (one for each defined integration channel). • MessageDefinition.table There can be one table per variant.

The MessageDefinition.table files and MessageConfiguration.table files are created when you migrate Ariba Buyer from version 7.x to version 8.x (see "Migrating Ariba Buyer" on page 80).

The MessageDefinition.table files are complete. However, the MessageConfiguration.table files still require some modification. These modifications include:

- Transferring CSV event definitions to the File Channel
- Converting CSV mappings to File Channel format

Each of these tasks is described below.

Transferring CSV events to the File Channel

In Ariba Buyer 7.x, the WAAB version of the adapter used a component called the CSVAdapter to load CSV file data. In Ariba Buyer 8.x, CSV file data is loaded exclusively by the Ariba Buyer File Channel adapter.

Hence, customizations made to the old IntegrationEvents.table file (see Table 25) are not automatically migrated to the new MessageConfiguration.table file. You need to:

1. Re-apply the customizations to the MessageConfiguration.table files located in:
   <*Ariba Buyer 8 dir*>\config\variants\<*variant name*>\partitions\<*partition name*>**file**\MessageConfiguration.table
2. Remove all CSV event definitions located in:

<Ariba Buyer 8 dir>\config\variants*<variant name>*\partitions*<partition name>***IBM**\MessageConfiguration.table

To ensure that all customizations are correctly applied, follow these steps.

1. For each CSV event definition, compare the entries in:
   <Ariba Buyer 8 dir>\...*<partition name>***file**\MessageConfiguration.table and
   <Ariba Buyer 8 dir>\...*<partition name>***file**\IBMMessageConfiguration.table
2. If there are **no** differences apart from those contained in the Channels sections, remove the entry from *<partition name>***file**\IBMMessageConfiguration.table.
3. If there are any differences apart from those in the Channels section (such as filename and action), make the corresponding change in *<partition name>***file**\MessageConfiguration.table, and then remove the entry from *<partition name>***file**\IBMMessageConfiguration.table
4. Once you have confirmed that all the modifications made in *<partition name>***file**\IBMMessageConfiguration.table have been transferred to *<partition name>***file**\MessageConfiguration.table, you may move *<partition name>***file**\IBMMessageConfiguration.table to a backup location, as this file is no longer required.

Note: The WAAB action called LoadAndDelete is equivalent to the WebSphere Business Integration Adapter for Ariba Buyer operation “Load And Delete”, and the action called UpdateAndDelete is equivalent to the operation “Update And Delete”.

Converting CSV mappings to File Channel format

Ariba Buyer 8.x provides mappings for all out-of-the-box CSV integration events.

If the mapping file used by the WAAB CSVAdapter was not customized, you can proceed to the next section.

If mappings were changed or added to the mapping file used by the WAAB CSVAdapter, you must make the same changes and additions to the mapping file used by the Ariba Buyer File Channel.

For details on the format of the mapping file, IntegrationMappings.aml, refer to chapter 4 in the *Ariba Buyer 8.x Customization Guide*.

Configuring the adapter and integration channel

The WebSphere Business Integration Adapter for Ariba Buyer consists of two main components; the integration channel and the connector. Each component runs in its own process. The two processes communicate with each other via RMI (remote method invocation).

You need to set parameters in each component to configure this communication. For information on setting the parameters correctly, refer to “Configuring the adapter” and “Configuring the integration channel” in Chapter 2 of this guide.

Generating metadata

The WebSphere Business Integration Adapter for Ariba Buyer is metadata-driven. This means that the adapter can handle changes to the structure of the business objects that flow through it without the need for recoding and recompilation. The structure of a business object is stored as metadata.

The metadata must always match the business object structure exactly. Whenever a business object's structure is changed, new metadata must be generated for that business object.

To generate all the required metadata, follow these steps.

1. From the <*Ariba Buyer 8 Server*>\bin directory, execute this command:
   ```
   initdb -reshapedb
   ```
2. Start Ariba Buyer 8.x.
3. Stop Ariba Buyer. 8.x

This procedure triggers Ariba Buyer to output metadata to the file system repository. The location of this repository is specified by the MetadataRepos parameter in Parameters.table.

You can verify that the metadata was generated by checking the MetadataRepos directory. The directory should contain a folder for each variant that in turn contains a file for each Ariba Buyer object defined in the variant, and it should also include a folder for each partition in the variant. This folder contains all the MessageConfiguration.table entries for the partition.

The next step is to create business object definitions from the metadata stored in the file system repository. For details, refer to Chapter 2, "Generating business object definitions" in this guide.

Creating object mappings

When you have created the business object definitions, you can use them to construct the object mappings in the broker. These object mappings replace the mappings currently being done by customer adapters or existing broker flows.

For details on creating collaborations for WebSphere InterChange Server, see *WebSphere Business Integration Server: Collaboration Development Guide*.

For details on creating MQ flows for WebSphere message brokers, see *Implementing Adapters with WebSphere Message Brokers*.

For details on creating MQ flows for WebSphere Application Server, see *Implementing Adapters with WebSphere Application Server*.

Appendix D. Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

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

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

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

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

IBM WebSphere Business Integration Adapter Framework V2.4.0.