IBM WebSphere Business Integration Adapters

Adapter for IndusConnect Framework User Guide

Adapter Version 1.0.1

IBM WebSphere Business Integration Adapters

Adapter for IndusConnect Framework User Guide

Adapter Version 1.0.1

Note

Before using this information and the product it supports, read the general information under Appendix D, "Notices," on page 67.

19December2003

This edition of this document applies to IBM WebSphere Business Integration Adapter for IndusConnect Framework, version 1.0.1, and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

Contents

About this document

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

This document describes the installation, configuration, troubleshooting, and business object development for the IBM WebSphere Business Integration Adapter for IndusConnect Framework.

In this guide the terms connector and adapter are used interchangeably.

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 the documentation or read it directly online at one of the following sites:

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

  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- For using adapters with InterChange Server:

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

  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- For more information about message brokers (WebSphere MQ Integrator Broker, WebSphere MQ Integrator, and WebSphere Business Integration Message Broker):

  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- For more information about WebSphere Application Server:

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

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

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, 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 IndusConnect Framework is installed on your system.
`%text%` and `$text`	Text within percent (%) signs indicates the value of the Windows *text* system variable or user variable. The equivalent notation in a UNIX environment is $*text*, indicating the value of the *text* UNIX environment variable.
ProductDir	Represents the directory where the product is installed.

New in this release

The version of this document for adapter version 1.0.1 of IndusConnect Framework contains the following new information:

- In Chapter 2, “Installing and configuring the adapter,” on page 7, connector configuration procedures describe using the Connector Configurator to perform configuration tasks. Also in this chapter, the software compatibility section has been revised to address changes in the platforms and software versions supported.
- New configuration parameters are provided for error handling and document encoding.
- A new section, “Using the adapter with WMQIB or WBIMB” on page 18 has been added and discusses modifications you must make when using WMQI as your integration broker.
- Appendix C. ″The IndusConnect Framework Configuration file″ in the previous version of this document has been removed.
- Appendix C, “Connector configuration hierarchy,” on page 65 has been added to reflect the different implementation of configuring connector properties this release.
- “Converting message files for non-Latin 1 and non-Unicode languages” on page 19 has been added and explains how to provide for messages in a language other than English.
- Appendix D. ″Connector feature list″ in the previous version of this document has been removed.

Chapter 1. Overview of the adapter

Introduction

The IBM WebSphere Business Integration Adapter for IndusConnect Framework facilitates transactions with the Indus integration product IndusConnect. The adapter for IndusConnect Framework enables integration of an Indus International core solution with other best-of-breed enterprise applications. The adapter communicates using the IndusConnect API to support all IndusConnect Framework integration business objects and provides data translation capabilities between the standard IndusConnect and WebSphere Business Integration business objects. With this adapter, an Indus enterprise asset management (EAM) solution is easily married to the rest of the enterprise. The IndusConnect Framework application provides support for work and asset management.

What is an adapter?

An adapter (also called a connector in this guide) consists of an application-specific component and the connector framework. The application-specific component contains code tailored to a particular application. The connector framework, whose code is common to all connectors, acts as the intermediary between the integration broker and the application-specific component.

A connector provides translation services for the WBI system, moving data between collaborations and either:

- An application
- A programmatic entity—a remote web server, for example—that understands a technology standard, such as XML, that is handled by a WBI connector A connector has two specialized ends:

The connector controller interacts directly with collaborations. A connector agent interacts directly with an application or other programmatic entity.

The connector framework provides the following services between the integration broker and the application-specific component:

- Sends business objects (pre-specified packets of information) when triggered by application events.
- Manages the exchange of startup and administrative messages.

Each connector is unique, because it communicates with its application according to the application's interfaces. If there is an application programming interface (API), the connector can use it. However, a connector for an application without an API can use whatever method the application provides, such as user exits or e-mail messages.

To detect application events in which collaborations are interested, a connector polls the application or uses the application's event callback notification mechanism, if there is one. A connector can also interact with the application at the command of a collaboration or to verify the results of its previous requests.

This guide contains information about the application-specific component and connector framework. It refers to both of these components as the connector.

Adapter architecture

The IndusConnect Framework adapter is built using the WebSphere business Integration Adapter (WBIA) framework and WBIX adapter framework extensions. WebSphere Business Integration extensions (WBIX) are a set of modular components built on top of the WBIA framework and are configured and instantiated at runtime. The components are, at a high level, two transports and one or more logic modules. Logic modules process an internal business object enroute from the originating transport to the target transport. Logic modules can be chained together if necessary. A **routing path** defines a path for a particular business object from one transport through certain logic modules and into the other transport. The WBIX XML data handler is used to convert between generic business objects and their XML representation for processing by the XSLT logic module. Figure 1 is a high-level view of the architecture.

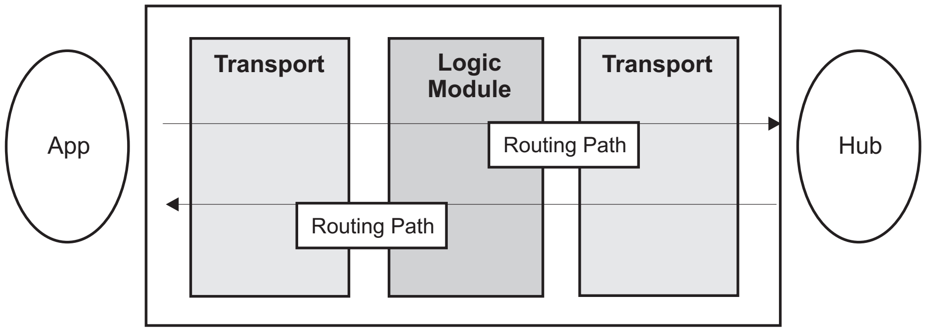

Figure 1. A high-level view of the adapter architecture

Figure 2 is a slightly more detailed view showing more of the components. Descriptions of each of these components follow.

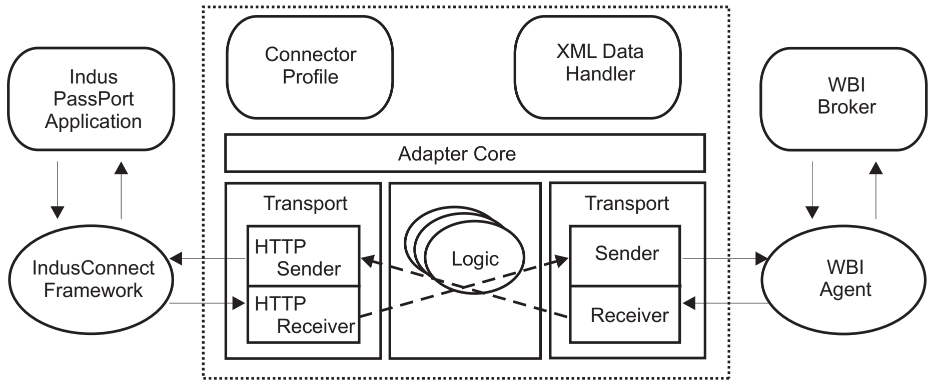

Figure 2. Slightly more detailed view.

The adapter components

The following are the adapter components:

Adapter Core
The adapter core controls the data and logic flows in the adapter.

Connector Profile
The connector profile describes the connector configuration and its properties. The Connector Configurator, which is a tool with a graphical user interface, enables you to define the configuration and properties.

WBIX XML Data Handler
The WBIX XML data handler provides conversion between generic business objects and their XML representation for processing by the XSLT logic module.

Event store
The event store provides persistence throughout adapter processing. Events are persisted within the queue, by default, until written to the event store. See “Event life cycle” on page 5 and “Event store file naming” on page 5 for additional details on the event store component.

The event store for this adapter is com.ibm.wbix.adapter.transports.FileSystemEventStore.

Transport
A transport encapsulates inbound, outbound, and persistent communications logic for the adapter and integration broker. The outbound component of the transport is the sender. The inbound component of the transport is the receiver. The persistence component is the event store.

Because an application might have a mix of different protocols as its integration interface (for example, an SQL database interface for some data and an MQ messaging interface for other data), multiple transports can be loaded with a single adapter to enable the adapter to communicate across disparate protocols. In addition, each transport can house multiple incoming (receivers) and outgoing (senders) endpoints, such as a single receiver or sender for each of the application messaging queues. Transports are provided for some common interfaces. See “Transport modules” for additional details on the transport component.

Logic Modules
Logic modules process data from an inbound transport receiver enroute to the outbound transport sender. These modules provide translation functionality between the application and integration broker data formats. For example, an XSLT logic module is provided for the situation where both inbound and outbound formats are in XML format.

You can create custom logic modules to provide processing specific to your requirements. For example, you could create a specific XSL file to define a specific grammar transformation. Logic modules can also be chained if you require that capability.

The logic module for this adapter is com.ibm.wbix.adapter.logicmodules.XsltLogicModule.

Transport modules

The standard WebSphere Business Integration sender and receiver transport modules are used for communication with the integration broker. They expect the body of the message to contain an XML format recognized by the adapter or the WBI transport, which then gets converted into a WebSphere Business Integration Java object and sent to the integration broker for processing. The transport component contains modules that communicate with the integration broker and modules that communicate with the application.

For this adapter, the WebSphere Business Integration transport consists of the following components:

com.ibm.wbix.datahandlers.XMLDataHandler
The data handler converts WBI business objects to and from the adapter's WBI XML.

com.ibm.wbix.adapter.transports.wbi.WBISender
This module does not require any configuration parameters.

com.ibm.wbix.adapter.transports.wbi.WBIReceiver
This module does not require any configuration parameters.

Adapter servlets

A set of servlets are provided for use in conjunction with the adapter and the IndusConnect Framework. These servlets are used as an abstraction for the IndusConnect external event table. The abstraction enables the adapter to interface with IndusConnect in a uniform manner over HTTP. The servlets enable the adapter to do the following:

- Retrieve events from the event table
- Update the status of events in the event table
- Delete events from the event table

The servlets are packaged as a J2EE Web application (a .warfile). This application must be installed on the application server hosting the IndusConnect application. The servlets' database access is configured using the IndusConnect apifw.properties file. To resolve this file location, the servlets must be passed a Java™ system property named "*passPort_properties*". The value of this property is the directory in which the apifw.properties file resides. This configuration ensures that IndusConnect and the servlets access the same database with the same parameters. See the application server documentation for instructions on how to install a Web application and supply a system property.

A description of the adapter servlets and their parameters follows:

printEventTable
Displays all events in the external event table (TIDAPIEV).

getEventXML
Gets full outbound event XML detail of event. Multiple documents can be returned.

eventID
Unique identifier (timestamp from above) to identify the event.

Note: Not setting eventID returns XML for all documents in the table.

status Filter by event status (all events have status equal to this argument).

Note: The status and notStatus parameters are mutually exclusive.

notStatus
Reverse filter by event status (all events status not equal to this argument).

removeEvent
Removes event from the external event table (TIDAPIEV).

eventID
Unique identifier (timestamp from above) to identify an event. Specifying eventID = all removes all events in the table.

setStatus
Sets the status of an event in the event table.

eventID
Unique identifier (timestamp from above) to identify an event.

status String value to set as status for this event.

Event life cycle

Events generated in the Indus PassPort component of the application are propagated to the integration broker as follows:

Note: In the following description, remote event store refers to the Indus PassPort event table.

1. Indus PassPort publishes the application events to the remote event store with status PRE_POLL_STATUS.
2. The adapter servlet returns the published event to the adapter in the next poll.
3. The adapter adds the event to the local event store with status EVENT_RECEIVED. See Table 1 on page 6 for a list of the status constants.
4. The adapter updates the status of the event in the remote event store to POST_POLL_STATUS
5. Processing occurs on the event in the local event store which changes status:
   - Translation changes status
   - WBIA polling for events changes status
6. When the processing and polling completes, the local event status is changed to the appropriate status value, for example, WBIA polling results in ERROR_POSTING_EVENT or SUCCESS.
7. If the final local status value is an error, the event in the remote event store is set to ERROR_COMPLETION_STATUS. If successful, the event in the remote event store is set to SUCCESS_COMPLETION_STATUS.
8. The event is archived:
   - If archiving is enabled, the local event is copied to the archive store.
   - If the DeleteOnArchive option is set, the event is deleted from the local and remote event stores.

In the current configuration:
PRE_POLL_STATUS=R
POST_POLL_STATUS = 9
ERROR_COMPLETION_STATUS = 7
SUCCESS_COMPLETION_STATUS = 8

Event store file naming

The FileSystemEventStore component of the IndusEventStore uses the local file system as the provided default persistent event store implementation. When the adapter is started, it creates separate directories for the event store proper and the event store archive.

For each event added to store, the adapter writes a new log file with the event contents. Each event record uses a naming convention consisting of a unique ID, appended with an event status separated by an underscore (_), for example, *uniqueID_status value*. The unique ID is obtained by computing a hash function on the event ID of the message.

If you use this event store, ensure that the event IDs generated by the receivers contain no special characters that are disallowed by your file system. For the adapter for IndusConnect Framework, the event ID is the timestamp of an event outbound from the Indus PassPort event table.

For event status changes, the event store generally renames the file to reflect the updated status. However, for status changes that indicate a change in the event body, for example, IN_PROCESSING to READY_FOR_POLL, the event store rewrites the file contents.

The status constants are listed in Table 1. Error constants are listed in Table 2.

Table 1. Status constants

Status constant	Value
EVENT_RECEIVED	2147483647
IN_PROCESSING	2147483646
EVENT_PROCESSED	2147483645
READY_FOR_POLL	0
IN_PROGRESS	1
SUCCESS	3

The error constants are:

Table 2. Error constants

Error constant	Value
ERROR_IN_LOGIC_PROCESSING	-2147483647
UNSUBSCRIBED	2
ERROR_PROCESSING_EVENT	-1
ERROR_POSTING_EVENT	-2
ERROR_OBJECT_NOT_FOUND	-3

Chapter 2. Installing and configuring the adapter

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

Compatibility

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

- **Adapter framework:** WebSphere Business Integration Adapter Framework, version 2.4
- **Integration brokers:**
  - WebSphere InterChange Server, versions 4.2.1 and 4.2.2
  - WebSphere MQ Integrator Broker, version 2.1 with CSD05
  - WebSphere Application Server Enterprise, version 5.0.2, with WebSphere Studio Application Developer Integration Edition, version 5.0.1
  - WebSphere Business Integration Message Broker, version 5.0 with CSD02.

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 MQ Integrator Broker, see *Implementing Adapters with WebSphere MQ Integrator Broker*.
- For WebSphere Application Server, see *Implementing Adapters with WebSphere Application Server*.

Installing related software

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

One of the following application platforms:
- Windows 2000 with Service Pack 4
- AIX 5.1 with maintenance level 4
- AIX 5.2 with maintenance level 1
- HP-UX 11.11
- Sun Solaris 7.0 or Solaris 8.0

Beginning with the 1.0 version, the adapter for IndusConnect Framework is no longer supported on Microsoft Windows NT.

The following support software:
- IndusConnect Framework, version 9.0.3

Overview of the installation process

The installation process consists of installing components on the server and installing products on the client. The client and server can be the same physical system. This guide identifies components as either server or client.

Follow these steps to install the adapter for IndusConnect Framework.
1. Install the integration broker.
2. Install the adapter for IndusConnect Framework.
3. Configure and initialize the adapter for IndusConnect Framework.

Installing the adapter

The adapter installation software is available on CD or as an ESD (electronic software delivery) download from the Web. Use Passport Advantage to download the setup files for IBM WebSphere Business Integration IndusConnect Framework to your system. The product part number also refers to the ESD image. The procedures for installing the adapter for IndusConnect Framework are described in detail in the *Installation Guide for WebSphere Business Integration Adapters*, which along with the *Release Notes*, is located in the InfoCenter at the following address: `http://www.ibm.com/websphere/integration/wbiadapters/infocenter`.

Installing the adapter on a Windows system

If you are installing from an ESD image, the file download utility copies an executable file containing a set of installation files onto your machine. To install the adapter:
1. Run the setup file, setupwin32.exe, for your platform from the setup directory or from CD. The InstallShield appears.
2. Select the language and click **OK**.
3. Step through the screens by clicking **Next**.

   Note: You will be asked to select a file location.

- If you are running ICS, you must install these files in the same product directory in which you installed ICS, for example, `C:\IBM\WebSphereICS`. In these steps, the ICS directory is called *ICSdir*.
- If you are running WebSphere MQ Integrator Broker or WebSphere Business Integration Message Broker, you must install these files in the same product directory in which you installed the WebSphere Business Integration adapter package. This is `C:\IBM\WebSphereAdapters` by default.

Use the **Browse** button to locate the correct directory and select it.

Installing the adapter on a UNIX system

If you are installing the adapter on a UNIX platform:

1. Run the appropriate setup file for your system:
   - For AIX®, run setupAIX.bin
   - For HP-UX, run setupHP.bin
   - For Solaris, run setupsolarisSparc.bin
2. Specify a directory for the files:
   a. If you are running ICS, you must install these files in the same product directory in which you installed ICS. This is `/opt/IBM/WebSphereICS` by default.
   b. If you are running WebSphere® MQ Integrator broker or WebSphere Business Integration Message Broker, you must install these files in the same product directory in which you installed the WebSphere Business Integration adapter package. This is `/opt/IBM/WebSphereAdapters` by default.

Installed file structure

Table 3 shows the file structure used by the adapter 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 back slashes.
2. All product path names are relative to the directory where the product is installed on your system.

Table 3. Adapter files installed on system

Subdirectory of %*Indus*%	**Description**
	Contains: • CustDataHandler.jar – .jar file containing the WBIX-specific XML data handler. • Indus.jar – .jar file containing application-specific adapter classes. • start_Indus.bat and start_Indus.sh – startup scripts for the adapter. • WBIX.jar – .jar file containing WBIX-framework adapter class. • indusicl.zip – ICL .zip file for import into customer user project (in System Manager). • Indus.war – Web application archive containing adapter servlets to be installed on IndusConnect machine.
translations	Directory for object mapping translations. Contains schemaTypes.xml, which is a translation manager configuration file.
translations\xslt	Directory containing actual translation mapping files (XSL stylesheets).
lib	Directory containing Java library classes for the adapter.

Importing the libraries and the connector

You must import the libraries for the adapter. In these steps, the Integration Component Library is called *wbieuicl*. You can substitute your own library name.

Do the following steps to import the libraries.

1. Start the WebSphere Business Integration System Manager. Select **Start→Programs→IBM WebSphere InterChange Server→IBM WebSphere Integration Toolset→Administrative→System Manager**.

   Note: The first time you follow these procedures, you must select **Window→Open Perspective→Other→System Manager**, then select **OK**.

   The Projects window should contain:
   User Projects
   Integration Component Libraries
2. Right-click **Integration Component Libraries**.
3. Select **New Integration Component Library**.
4. In the Project name field, enter the library name: *wbieuicl* to create a folder named *wbieuicl*.
5. Click **Finish**.
6. Click **File→ Import** to open the Import window.
7. Select **Zip file**.
8. Click **Next**.
9. For the **Zip file** field, browse to indusicl.zip. If you are running ICS, the file is located in *ICSdir*/connectors/Indus.
10. For the **Folder** field, browse to the folder you created for *wbieuicl*.
11. Click **Finish** to import all the files to the folder.
12. Right-click on the *wbieuicl* folder.
13. Select **Refresh view**. The view shows all the business objects that were imported.

Creating the new User Project

In the following steps, the project is called ***INDUS***. You can specify any project name, but you must be consistent throughout the installation and configuration steps.

Do the following steps using the System Manager:

1. If ICS is installed on your system, right-click **User Projects→New ICS project**, then go to step 4.
2. Right-click **User Projects→New User Project**.
3. Select either **New WAS project** or **New Message broker project**.
4. Specify the project name: ***INDUS***.
5. In the lower window, select the check box beside the *wbieuicl* library folder name. If you specified a different library name in step 4 in "Importing the libraries and the connector," use that library name here.
6. Click **Finish** to create the folder ***INDUS*** with all the business objects in it.

Configuring the adapter

The connector component of the adapter has two types of configuration properties: standard configuration properties, which apply to most adapters, and connector-specific configuration properties, which apply only to your adapter and provide a way for you to change static information or logic within the connector without having to recode and rebuild it. A connector obtains its configuration values at startup. You must set the values of these properties before running the connector.

To configure connector properties, use the Connector Configurator tool. Details are given in Appendix B, “Connector Configurator,” on page 49. This tool provides a graphical user interface for configuring the connector.

When you have finished specifying values for the connector’s configuration properties, the 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 and WebSphere Application Server).

Specifying properties

In this section, the steps required to specify properties are described. The project name for the folder used in these procedures is *INDUS*.

To specify properties, start the WebSphere Business Integration System Manager and do the following steps

1. Expand the **User Projects** folder.
2. Expand **InterChange Server Projects**.
3. Expand the *INDUS* folder.
4. Expand the **Connectors** folder.
5. Double-click **IndusConnector**::*wbieuicl*.
6. Click the **Standard Properties** tab to add or modify configuration properties.
7. Specify the AgentTraceLevel for the adapter. Selecting the value **5** generates detailed trace level; zero (**0**) generates no trace information.
8. Your application information might inform you which of the following transports to select for the DeliveryTransport property.
   MQ
   IDL (the default transport method)
   JMS

   Accept the defaults for the remainder of the configuration for standard properties.

   In the next few steps, you define the connector-specific properties for the transport.
9. Click the **Connector-Specific Properties** tab.
10. Click to expand **wbiadapter**.
11. Click to expand **transports**.
12. Expand **Indus**.
13. Expand **senders**.
14. Expand **IndusSenderReceiver**.
15. Expand **parameters**.

16. Specify the appropriate values for the properties listed in Table 4.

Table 4. Connector-specific configuration properties for adapter for IndusConnect Framework transport

Property / Parameter	Description	Value	Default value
sendUrl	This is the URL to which this sender component sends events. This URL must be fully qualified and resolvable over HTTP.	This parameter should point to the IndusConnect process servlet, for example, APIFWAdapterServlet	N/A
pollingInterval	Interval in milliseconds that specifies how long to wait before calling the poll() method to retrieve new events from IndusConnect. See “Polling parameter considerations” on page 16 for additional polling information.	Integer value in milliseconds or ″no″ to stop polling.	30000
prePollStatus	When polling, this is the status level of events that are searched for. Events of this status are returned in the poll() method.	The value of this option must be a single character.	R
postPollStatus	When polling, this is the status level that events are set to after polling. All events returned in the poll() method have their status level set to this value.	The value of this option must be a single-digit integer.	9
apifwVersion	The version of APIFW business objects. This value is used to construct the DTD and header for process requests.	String	V090000
dtdLocation	Location of business object DTD documents on the IndusConnect Framework server. This value is necessary to construct DTD headers for process requests.	String name of the xml directory on the server hosting the IndusConnect Framework and containing the IndusConnect.DTD files.	N/A
environment	Indus PassPort environment name needed for apiFW. IndusConnect Framework application environment. This value is necessary to construct DTD headers for process requests.		D903
username	Application (for example, Indus Passport) username that the adapter uses when conducting requests.	String This value must match the apifw.properties apiUsername property. See IndusConnect Framework documentation for this property.	N/A
DefaultEncoding	Default encoding to assume for documents received from the server when the ContentEncoding header is not set.	Valid Java encoding Strings	UTF-8
DefaultSendMethod	Parameter indicating whether to default to GET or POST as the method for accessing URLs.	GET or POST	POST

Table 4. Connector-specific configuration properties for adapter for IndusConnect Framework transport (continued)

Property / Parameter	Description	Value	Default value
generateErrorObject	Flag that indicates whether the adapter should generate a separate business object from the error and warning information received from the IndusConnect Framework. The object generated is named indusErrorObject.	True or False	False
OverrideServerEncoding	Flag indicating to override the set ContentEncoding header of the server with the DefaultEncoding property	True or False	False

17. Expand **eventstore**.
18. Expand **parameters**.
19. Specify the appropriate values for the properties listed in Table 5.

Table 5. Connector-specific configuration properties for adapter for IndusConnect Framework event store

Property / Parameter	Description	Value	Default value
fetchURL	URL from which this store retrieves events.	{IBM servlet location}/getEventXML	N/A
fetchIDParameter	Parameter name that the remote site uses to identify the event to retrieve. When placed in the URL, the value of this parameter will be the eventID argument passed into the fetchRecords method.		eventID
fetchStatusParameter	Parameter name that the event store will use to retrieve events of a set status. When placed in the URL, the value of this parameter will be the status argument passed into the fetchRecords method.		status
postPollStatus	Event status to set after poll. This value is used to indicate when an event has been copied from the HTTP event store to the file system event store.	The value of the option should match that specified in the IndusSenderReceiver class and must be a single-digit integer.	9
updateURL	URL that allows updating of events in the event store.	{IBM servlet location}/setStatus	N/A
updateIDParameter	Parameter name that the remote site uses to identify the event to update. When placed in the URL, the value of this parameter is the eventID argument passed into the setEventStoreStatus method.		eventID
useGetMethod	Indicates whether the event store should use GET as the method for accessing URLs. Change this value to FALSE in a production environment.	TRUE (GET) or FALSE (POST)	TRUE
eventDir	Directory in which event store is located. This argument can be relative or fully qualified.		eventStore

Table 5. Connector-specific configuration properties for adapter for IndusConnect Framework event store (continued)

Property / Parameter	**Description**	**Value**	**Default value**
archive	Directory in which archive store is located. Set this argument to null or omit to not implement archival.		archive
archiveFailureFatal	Boolean indicating whether to return a fatal exception if the system fails to initialize the archive.	True or False	False
successCompletionStatus	Event status to set after the adapter has successfully consumed an event (for example, successfully passed the event on to the integration broker).	The value of this option must be a single-digit integer.	8
errorCompletionStatus	Event status to set after the adapter has failed to consume an event (for example, failure either in logic processing or in sending the event to the integration broker).	The value of this option must be a single-digit integer.	7
deleteOnArchive	Boolean indicating whether to delete an event when requested to archive an event.	True or False	False
updateStatusParameter	Parameter name that the remote site uses to identify the new status of the specified event. When placed in the URL, the value of this parameter is the status argument passed into the setEventStoreStatus method.		status
deleteURL	URL that supports deletion of events from the store.		N/A
deleteIDParameter	Parameter name that the remote site uses to identify the event to delete. When placed in the URL, the value of this parameter is the eventID argument passed into the deleteEvent method.		eventID

20. Expand **logicmodules**.
21. Expand **TranslationManager**.
22. Expand **parameters**.

    The adapter uses the XSLT logic module as the primary transformation mechanism for translating between the adapter's XML messages and the WBI XML. Translations between XML formats are direct, there is no chaining. To configure the logic module with properties other than the defaults, specify values for the properties listed in Table 6.

Table 6. Connector-specific configuration properties for adapter for IndusConnect Framework logic module

Property / Parameter	**Description**	**Value**	**Default value**
schemaConfig	Provides translation between application and integration broker data formats.	*repository_dir*\schemaTypes.xml	translations \schemaTypes.xml
repository	The xslt subdirectory is specified as the root for name resolution within the repository, versus the top-level translations directory.	*repository_dir*\xslt	translations\xslt
Indus	Used to configure routing path.		CW

Table 6. Connector-specific configuration properties for adapter for IndusConnect Framework logic module (continued)

Property / Parameter	Description	Value	Default value
CW	Used to configure routing path.	This value must match the routing paths.	Indus

23. If you selected MQ as the delivery transport in step 8 on page 11, you must specify the queue manager name:
    a. Click the **Messaging** tab.
    b. Specify the queue manager name in the **Queue Manager** field. The default is **server.queue.manager**.
24. After making the changes, click **File→Save→ To Project**.
25. Exit the Connector Configurator.

Updating the user project

To update the user project, do the following:

1. Under **User Projects**, right-click the *INDUS* folder.
2. Select **Update project**.
3. Expand the *wbieuicl* folder.
4. Expand the **Connectors** folder.
5. Check IndusConnector.
6. Click **Finish**.
7. Click **Yes** to the **Overwrite Confirmation** prompt.

Deploying the project to the server

In the InterChange Servers window of the System Manager, you should already have an instance of your local server. In these steps the server instance is called ***INDUS**server*. Do the following steps:

1. Ensure the WebSphere InterChange Server is running.
2. Expand the **InterChange Server Instances** folder.
3. Right-click ***INDUS**server*.

   Note: If no server instances appear under Servers, you must first register a server:
   a. Right-click **InterChange Server Instances**.
   b. Select **Register Server**.
   c. Specify the information needed to complete the Register new server dialog.
   d. Click **OK**.
4. Select **Connect**.
5. Specify **User name** and **Password** information in the appropriate fields.
6. Click **OK**.

   The status should change to connected.
7. Expand **User Projects**.
8. Expand **InterChange Server Projects**.
9. Select ***INDUS***.
10. Right-click ***INDUS*** → **Deploy user project** to open the project list.

11. Check ***INDUS***.
12. Select **Create schema** and **Compile**.
13. Select the destination server, ***INDUS****server*.
14. Click **Next**.
15. If an ***INDUS*** folder is displayed, select the check box.
16. Click **Finish**. This step might take several minutes to complete.
17. Click **OK**.
18. Shutdown and restart the InterChange Server as follows:
    a. Right-click ***INDUS****server*.
    b. Select **Shut Down**.
    c. Select **Gracefully**.
    d. Click **OK**.
    e. Click **OK**.
    f. Start the InterChange Server.

Polling parameter considerations

The polling parameters control how often the IndusConnector polls for business objects to transmit to the integration broker. You might poll less often if the application at the other end of the collaboration processes requests in a batch mode. If you set the polling frequency too high, performance might suffer. Aim to balance polling between the applications at both ends of the collaboration. The connector specific property, pollingInterval is used to control this polling interval.

The standard connector property, PollFrequency, controls how often the server receives events from the adapter, however, you do not need to change the value for this standard connector property.

Configuration and internationalization

The data handler is invoked from the adapter and requires configuration. This configuration includes locale-specific properties that define number and date format conversion. Within any adapter using the data handler, the DataHandlerMetaObjectName property must be defined. The value of this property is the name of the meta-business object that defines the appropriate data handlers for each MIME type. The meta-object specified for the text/xml type designates the WBIX XML data handler and its configuration options.

To configure these options, you must edit the business object MO_Indus_XMLConfiguration. Using the System Manager, you can edit the properties in the Business Object designer according to the following parameters. When you are finished, you must redeploy this object to the server.

The options are:

UseDefaults
: Boolean indicating whether GBOs should be populated with their default values. If the parameter does not exist within the meta object, the default behavior is to use defaults.

AppDateFormat
: The date format that the data handler expects from the application and passes to the application. The default date format for the Indus adapter is `yyyy-MM-dd hh:mm:ss`.

See the section "Date formats" for additional formatting details.

EnforceRequired
Specifies whether to enforce requirements. If **TRUE** is specified, events containing required fields that contain no data are rejected by the data handler. If **FALSE** is specified, required fields that contain no data are accepted by the data handler.

ClassName
Java class: com.ibm.wbix.datahandlers.XMLDataHandler

ICSDateFormat
The date format that the data handler expects from the integration broker and passes to the integration broker. The default date format for the Indus adapter is `MM/dd/yyyy`.

See the section "Date formats" for additional formatting details.

XmlEncoding
Specifies the encoding to use when producing XML from an ICS object. Valid encodings are as follows:

US-ASCII
Seven-bit ASCII, also referred to as ISO646-US or the Basic Latin block of the Unicode character set.

ISO-8859-1
ISO Latin Alphabet No. 1, also referred to as ISO-LATIN-1.

UTF-8 Eight-bit Unicode Transformation Format.

Date formats

The date formats follow:

yyyy	The year in four-digit format. For example, 2003.
MM	The month in two-digit format. For example, October is 10.
dd	The date in two-digit format. For example, the eighteenth is 18.
hh	The hour in 24-hour format. For example, 11:00 p.m. is 23.
mm	The minutes.
ss	The seconds.
SSSS	Thousandths of a second.

For further details, see the Java 1.3.1 SimpleDateFormat specification.

Starting the System Monitor

Start the System Monitor and connect to the server. You should have a view of each connector. Look for the IndusConnector. The yellow icon means it is ready to start.

Right-click the IndusConnector to start the connector. The icon turns to green when it has started successfully.

Starting the connector

Windows
```
start_Indus.bat Indus name_of_ICServer
```

UNIX
```
/opt/IBM/WebSphereAdapters/bin/connector_manager -start Indus
```

Using the adapter with WMQIB or WBIMB

When using either WebSphere MQ Integrator Broker (WMQIB) or WebSphere Business Integration Message Broker (WBIMB) as your integration broker, you must make some modifications to this adapter. The modifications enable the broker to successfully process the default wbieuAsset business object, which is recursively defined by default. The changes you make must be made to the wbieuAsset business object and XSL translations that use the wbieuAsset business object.

Changing business object definitions

Perform the following steps to make the changes to the wbieuAsset business object:

1. Open the Business Object Designer. For information on launching and using the Business Object Designer, see *IBM WebSphere Business Integration Adapters Business Object Development Guide*.
2. Open the wbieuAsset business object.
3. Save a copy of the wbieuAsset business object under a different name, for example, subAsset.
4. Change the type of the subAsset attribute in the original wbieuAsset business object. The new type must be the type created in step 3.
5. Refresh all business objects that depend on the wbieuAsset business object (by default, wbieuWorkOrder and wbieuWorkTask business objects).
6. Redeploy all three business objects.

After completing these changes to the business object, you must make some XSL translation changes.

Changing XSL translations

This section explains how to make changes to the XSL translations that use the wbieuAsset business object.

1. Open the following convert.xsl XSL translations which reside in the translations directory:

   Note: Before you begin, save backup copies to another directory or with a different file extension.

   xslt/Indus/cw/Equipment

   xslt/Indus/cw/Schedule

   xslt/Indus/cw/UniqueTrackedCmmdty

   xslt/Indus/cw/WorkOrder
2. For each translation, search for the text ″subAsset″. This text should appear in the **Attribute** elements and the text ″subAsset″ should appear as the **name** attribute of the XML element.
3. For each instance of the subAsset attribute, change the XML attribute named **class** from wbieuAsset to reflect the new business object created in step 3 in "Changing business object definitions."
4. Save all new translations.

The following examples show the business object definition before the changes are made and what the business object definition looks like after the changes are made. In this example, assume the subAsset attribute has been changed from wbieuAsset to type newAsset in the Business Object Designer:

Before:

```
<xsl:element name="Attribute">
  <xsl:attribute name="class">wbieuAsset</xsl:attribute>
  <xsl:attribute name="name">subAsset</xsl:attribute>
  <xsl:attribute name="type">SINGLECARDSTRING</xsl:attribute>
  <xsl:element name="BusinessObject">
  <xsl:attribute name="class">wbieuAsset</xsl:attribute>
  <xsl:attribute name="type">SINGLECARDSTRING</xsl:attribute>
  <xsl:element name="Attribute">
```

Becomes the following:

```
<xsl:element name="Attribute">
  <xsl:attribute name="class">newAsset</xsl:attribute>
  <xsl:attribute name="name">subAsset</xsl:attribute>
  <xsl:attribute name="type">SINGLECARDSTRING</xsl:attribute>
  <xsl:element name="BusinessObject">
  <xsl:attribute name="class">newAsset</xsl:attribute>
  <xsl:attribute name="type">SINGLECARDSTRING</xsl:attribute>
  <xsl:element name="Attribute">
```

Converting message files for non-Latin 1 and non-Unicode languages

The WBIX adapter framework extensions contain messages and property files for the English language. If you want to translate these messages to other languages with non-Latin 1 and non-Unicode characters, do the following:

1. Translate the appropriate message file, for example, the WBIXResource.properties file, to the target language. Save this file in the system-default encoding.
2. Using the Java Development Kit, run the native2ascii tool on the translated property file to convert the characters to Unicode-encoded characters. Refer to the Javadoc documentation for information on the native2ascii tool and its command-line syntax.
3. Using the Java Development Kit, run the jar tool on the Unicode-encoded property file to place it in the WBIX.jar file. Place the translated, Unicode-encoded property file in the same path location as the original WBIX property file. Refer to the Javadoc documentation for information on the jar tool and its syntax.

The WBIX adapter framework extensions should now produce messages in the target language and Unicode characters.

Chapter 3. Understanding business objects

This chapter discusses the IndusConnect Framework business objects and their mappings. The following topics are discussed:

- "Business objects"
- "Overview of business object structure" on page 22
- "The IndusConnect Framework business object mappings" on page 22

Business objects

Collaborations and connectors interact by sending and receiving business objects through the InterChange Server.

A business object reflects a data entity—a collection of data that can be treated as an operative unit. For example, a data entity can be equivalent to a form, including of all of the form's fields. The form might typically be used in an application, or over the Web, to contain business information about customers, or employees, or invoices.

Business objects are cached in memory during collaboration execution for fast access, and also stored in a persistent transaction state store to provide robust recovery, rollback, and re-execution of collaborations upon server restarts after failures.

The WBI system creates business objects that reflect the information contained in entities. In this manual, a data entity is often referred to in the context of the kind of business information it contains—for example, an employee entity or a customer entity.

Role of a business object

A business object can act as an event, a request, or a response. See Table 7 on page 22 for a summary of the IndusConnect Framework business objects.

Event

A business object can report the occurrence of an application event, an operation that affected a data entity in an application. The application event might be the creation, deletion, or change in value of that collection of data. When a connector detects an application event and sends a business object to an interested collaboration, the business object has the role of representing the event, and so it is called an event in the WBI system. For example, a connector might poll an application for new employee entities on behalf of a collaboration. If the application creates a new employee entity, the connector sends an event business object to the collaboration.

Request

Requests are typically generated in one of two ways:

- A collaboration can send a business object as a request to a connector, instructing the connector to insert, change, delete, or retrieve some data in an application.
- The Server Access Interface can send a business object as a request to a collaboration, if that collaboration has been designed or customized to accept the Retrieve verb as a trigger.

Response

When a connector finishes processing a request, it usually returns a response. For example, after a connector receives a request to retrieve employee data from an application, it sends a business object containing the employee data.

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 contains only 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.

The IndusConnect Framework business object mappings

The IndusConnect Framework objects are the business objects that can be sent to WBI. The WBI business objects are those that are sent from WBI to IndusConnect Framework.

Table 7. The IndusConnect Framework business object mappings

IndusConnect Framework object	WBI object	Verbs from IndusConnect Framework object to WBI object	Verbs from WBI object to IndusConnect Framework object
Schedule	wbieuWorkSchedule	update	create, update, delete
SchedRefData	Crew[1] and CrewMember[1]	update	–
WorkOrder	wbieuWorkOrder	replace	create, update, delete, retrieve
ServiceRequest	wbieuWorkRequest	–	create, update, delete,

Table 7. The IndusConnect Framework business object mappings (continued)

Equipment	wbieuAsset[2]	insert, update, delete, replace	create, update, retrieve
UniqueTrackedCmmdty	wbieuAsset[2]	insert, update, delete, replace	create, update, retrieve
Catalog	indusCatalog	insert, update, delete, replace	create, update, delete
Manufacturer	indusManufacturer	insert, update, delete	create, update, delete
MaterialRequest	wbieuMaterialRequest	insert, update, delete, replace	create, update, delete
Vendor	Vendor	insert, update, delete	create, update, delete
EntityAddress	wbieuPartner	insert, update, delete	–
ClientData	wbieuConsumer	–	create, update, delete

Notes:

1. Multiple Crew/CrewMember objects can result from one ScheduleRef object
2. wbieuAsset is mapped to the Equipment or UniqueTrackedCmmdty object based on the presence of the utc or serialNumber attribute

Chapter 4. Troubleshooting and error handling

This chapter contains information to help you isolate problems when they occur. The following topics are discussed:

- "Tracing messages"
- "Problems constructing an adapter from a profile" on page 26
- "Tips for troubleshooting" on page 27
- "Using the error log to isolate problems" on page 27
- "Error handling" on page 28

Tracing messages

Tracing is an optional debugging feature you can turn on to closely follow a connector's behavior. Tracing levels 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 8 describes the types of tracing messages that the connector for the IndusConnect Framework adapter 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 8. 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.
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	Messages that provide information regarding message-to-business object and business object-to-message conversions.
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.

Table 8. Tracing messages (continued)

Tracing Level	Tracing Messages
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).

Problems constructing an adapter from a profile

Loading errors

During the construction of the adapter from the profile, the core will report both fatal and non-fatal errors. Fatal errors are caused if the adapter could not be constructed or there was a fatal inconsistency in the profile.

Fatal errors

Fatal errors include:

Could not load class
The class could not be found or there was an error in attempting to load it.

Could not configure class
The loaded class threw an exception during loading. This could result from not passing the correct parameters.

Could not start class
The class could not correctly start its operation, possibly resulting from the inability to connect to an application.

Routing paths conflict
At least two routing paths contain the same origin and object.

Routing destination specified does not exist
One of the routing paths specified a destination that did not map to any sender in the profile.

Routing origin specified does not exist
One of the routing paths specified an origin that did not map to any receiver in the profile.

Routing logic module specified does not exist
One of the routing paths specified a logic module that does not exist in the profile.

Non-fatal errors

Non-fatal errors during loading include warnings about correctly configured adapters that could cause inadvertent errors. Examples of non-fatal errors include:

Objects not supported from CW controller side.
If WBI is the broker, the connector controller on ICS may expect the adapter to support certain business objects. This warning is created if an object cannot be sent into the adapter from ICS.

Specific route overrides a Generic Route
: Warns if the profile has a specific route that overrides a generic route. Warns the user that an object from the particular receiver will take the specific route and not the generic route.

The non-fatal errors do not stop operation of the adapter because they do not interfere with the performance of the adapter. However, warning messages are sent in case the user has configured the adapter incorrectly.

Routing errors

At runtime, the adapter core will attempt to recover from all errors. However, the core will report errors to the transport receivers if their objects caused an error in routing. Each transport receiver must implement the handleError() function to attempt to recover from failed routing. Here are some errors that could occur during routing:

Route not found for object
: There could be several reasons for this error. One could be that the MessageEvent did not correctly specify the receiver who requested routing. Each transport receiver is expected to set its name in the MessageEvent before it routes the event. If the name specified is malformed this could also cause an error. Finally, a routing path may not exist in the profile, so the core does not have an route to apply.

Logic Module processing failure
: One of the logic modules in the path threw an exception during processing.

Tips for troubleshooting

Follow these tips for troubleshooting problems:
- Check that the connector is running.
- Check that the server is running.
- Be sure the business object structure is consistent with the operation.

Using the error log to isolate problems

When an error has occurred, the adapter writes an error record to the error log. To investigate the error, review the error log and locate the particular log message to determine where processing failed and the event status at the time of failure. To examine event contents, you must find the record in the file event store. Most error records that indicate a failure has occurred are located in the archive with status FAIL, that is a file exists ending with the characters _3 in the file name. Some records might still be in the event store directory with an error status (file ending with a negative number). To examine the file contents, find the log message that correlates the event ID with the event record file name:

```
(wbix_transport_msg0023=Translating eventID {0} to record name {1}).
```

Then examine the record file to see the event contents that caused the error.

The actions you take for a failed event are different depending on the event type and customer situation. However, a failed event can often be recovered and resubmitted to the hub.

Use the transaction data in the event store to understand the failure. It is not recommended that you manually resubmit the data to the adapter. Events should be recovered either by setting the InDoubtEvents standard connector property in the adapter or through the target application that originated the event.

You can resubmit failed events from the IndusConnect Framework in two ways. The preferred method is to re-edit the object in the Indus PassPort application so that it is republished by the IndusConnect Framework. The second method is by resetting the event status to PrePollStatus in the external event store table so that the event is processed by the adapter in the next poll.

The archive contains information on all events, both successful and failed. The exception to this rule are events with duplicated IDs. Only the first event of a specific status is archived. Therefore, if an event enters the adapter with ID 123 and another event with the ID 123 is sent in before the first event is sent to the broker, the archive contains two records: one record for the successful event and a second record for the failed duplicate. These will be the only two events in the archive even if more events with the same ID are sent. Error messages for the successive events continue to be written to the log however.

Error records have the following format:

```
[timestamp][system][component][thread label][error message: error exception:
errorexception message]
```

The following is a sample error which maps to the previous format:

```
[Time: 2003/06/25 15:37:05.715] [System: ConnectorAgent] [SS: IndusConnector]
[Thread: Indus.IndusSenderReceiver (#6276108)] Unable to extract
events from HTTP store: com.ibm.wbix.adapter.transports.http.HTTPTransportException:
XML parsing failed: org.xml.sax.SAXParseException:
The element type "MRAccounting" must be terminated by the matching end-tag "
</MRAccounting>"
```

Error handling

There are three broad categories of errors that can occur in the adapter:

- **Adapter configuration error**

  Upon startup or on an attempt to process a business object, errors can occur that are fatal or nonfatal:

  - Fatal configuration errors during startup prevent the adapter from starting and an error message is immediately logged.
  - Similarly, configuration or routing errors that occur during object processing are immediately logged. If the error occurs during a pollForEvents() call, the error returned is only an error code consistent with the method description. If the error occurs during a doVerbFor(), the error message, error return code, and message explanation are returned to the calling collaboration in the response object.

- **Logic processing error**

  Errors can also occur during logic processing (object translation) though these are usually caused by errors in configuration or improperly written XSL translation documents. These errors cannot occur during a pollForEvents() call. If the error occurs during a doVerbFor(), the error message, error return code, and message explanation are returned to the calling collaboration in the response object.

- **Transport sending/receiving error**

The majority of errors occur when an object is sent or received from the target application.

Sending errors occur in the doVerbFor() method:

- Communication errors can be fatal to the adapter when I/O errors occur. This results in the shutdown of the adapter with an APPRESPONSETIMEOUT return code.
- When a processing or target application error occurs, the collaboration exception object contains the return code and necessary error messages. These messages are embedded in the document returned from the application, and are handled by the receiving component of the adapter.

Receiving errors can occur during a poll of the application for new events, or when getting the response to an object sent:

- Fatal errors are I/O errors where the adapter fails to correctly connect to the application using HTTP. These errors are logged. On the next pollForEvents() and doVerbFor() method, the APPRESPONSETIMEOUT return code is sent.
- Nonfatal errors occur when processing of an individual message occurs (usually due to improper XML structure). These errors are logged in the pollForEvents() method. It is assumed that these methods are persisted in the event store and are handled again.
- When a nonfatal error occurs in the doVerbFor() method, any error messages provided by the target application are logged by the adapter. If the generateErrorObject property is set to true, the adapter attempts to create a separate error object that contains the target application error and warning messages and sends that to the server. Processing errors and their messages are logged by the adapter. No additional object is created. All error messages are included in the collaboration exception object returned to the collaboration.

Chapter 5. Extending and customizing the adapter

This chapter describes the XML grammar used by the adapter.

XML grammar

The XML grammar is an open definition containing the attribute = value pairs of a business object. The XML grammar supports the concept of embedded business objects and business object containers.

This adapter uses an XML format for sending business objects into and out of the integration broker. This format is defined generically to support any new business objects created inside the integration broker through the business object designer. This section describes what that format is (XML schema) and how the XML instance reflects the integration broker business object. Understanding this format enables you to add new objects for your adapter to support as well as modify existing business objects.

An object definition begins with a BusinessObject tag. The attributes for this tag are as follows:

BusinessObject	
class	The business object type of the object.
type	The cardinality of the object; the value is always SINGLECARDSTRING.
verb	The verb associated with this object. This attribute is necessary only in the initial document tag.

The contents of each BusinessObject element are Attributes. The attributes are as follows:

Attribute	
class	The class of the business object. Used only when the Attribute value is a business object and thus refers to the class of that object as in a BusinessObject tag.
name	The attribute name within the object definition.
type	The type of the attribute value. The value of this attribute can be any of business object simple types (for example. String or Date). In this case, the content of the tag is the String representation of the attribute value.

To define a business object as the attribute, the value of type is SINGLECARDSTRING (for single cardinality objects) or MULTIPLECARDSTRING (for *n*-cardinality objects). In this case, the content of the tag is a BusinessObject element (or multiple elements in the case of *n*-cardinality). With this grammar, arbitrary depth business objects can be expressed.

The schema is included in the following example for reference with a sample object definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="BusinessObject">
        <xsd:annotation>
            <xsd:documentation> Grammar definition for WBI Business Objects
</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element maxOccurs="unbounded" minOccurs="1" ref="Attribute"/>
            </xsd:sequence>
            <xsd:attribute name="type" type="xsd:string" use="required"/>
            <xsd:attribute name="verb" type="xsd:string" use="required"/>
            <xsd:attribute name="class" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Attribute">
        <xsd:complexType mixed="true">
            <xsd:choice maxOccurs="unbounded" minOccurs="0">
                <xsd:element ref="BusinessObject"/>
            </xsd:choice>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
            <xsd:attribute name="type" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

The following is a simple example.

```
<?xml version="1.0" encoding="UTF-8"?>
<BusinessObject class="EMail" type="SINGLECARDSTRING" verb="Create">
 <Attribute class="Header" name="Header" type="SINGLECARDSTRING">
  <BusinessObject class="Header" type="SINGLECARDSTRING">
   <Attribute name="sendDate" type="Date">2003-1-18</Attribute>
   <Attribute name="subject" type="String">Greetings</Attribute>
   <Attribute class="Name" name="Recipients" type="MULTIPLECARDSTRING">
    <BusinessObject class="Name" type="SINGLECARDSTRING">
     <Attribute name="FirstName" type="String">John</Attribute>
     <Attribute name="LastName" type="String">Doe</Attribute>
    </BusinessObject>
    <BusinessObject class="Name" type="SINGLECARDSTRING">
     <Attribute name="FirstName" type="String">Jane</Attribute>
     <Attribute name="LastName" type="String">Doe</Attribute>
    </BusinessObject>
   </Attribute>
  </BusinessObject>
 </Attribute>
 <Attribute name="body" type="String">Hello All</Attribute>
</BusinessObject>
```

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)
- The WebSphere message brokers: WebSphere MQ Integrator (WMQI) and Integrator Broker (WMQIB), and WebSphere Business Integration Message Broker (WBIMB).
- 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 9 on page 35 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker.

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

Table 9. 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 9. 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 9. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
OADMaxNumRetry	A positive number	`10000`	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 9. 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`	Component restart	CwXML if Repository Directory is not <REMOTE>: CwBO if Repository Directory is <REMOTE>
WsifSynchronousRequest Timeout	0 - any number (millisecs)	`0`	Agent 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 `RepositoryDirectory` is <REMOTE>.

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 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 `10000`.

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 39.

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 is`CwBO`.

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 update method is Component Restart.

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 (WMQI), WebSphere MQ Integrator Broker (WMQI) and WebSphere Business Integration Message Broker (WBIMB), collectively referred to as the MQ message brokers.
- 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 50).

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 51 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 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 55.)

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 51.
- 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 the 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`, `WebSphere Message Brokers`, 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 58..
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.
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 34.

Specifying supported business object definitions

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

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

- Business object definitions and map definitions should be saved into System Manager projects.
- Business object definitions and MQ message set files should exist if are using a WebSphere Message Broker as your integration broker.

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

If ICS is your broker

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

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

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

To delete a business object from the supported list:

1. To select a business object field, click the number to the left of the business object.

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

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

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

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

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

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

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

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

If a WebSphere Message Broker is your broker

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

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

To specify supported business objects:

1. Select the **Supported Business Objects** tab and, if System Manager is running, use the drop list to specify a business object name.
2. If System Manager is not running, enter the **Business Object Name**.
3. Enter the **Message Set ID (optional)**.

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

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

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

For WMQI:
- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder.
- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder, if you are using the file as a local configuration file.
- In a directory that you specify.

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

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

For details about using projects in System Manager, and for further information about deployment, see the following implementation guides:
- For ICS: *Implementation Guide for WebSphere InterChange Server*
- For 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. Connector configuration hierarchy

The following example shows the hierarchy for the adapter configuration. This view correlates with the structure shown when you view the connector properties with the Connector Configurator.

```
wbiadapter                                     Must start with wbiadapter tag
  transports                                   Must have transports tag with >= two child transports
      Broker Transport Name                    Name of transport is arbitrary
         senderclass                           Transport must have senderclass defined
         receiverclass                         Transport must have receiverclass defined
         eventstoreclass                       Transport can have one eventstore defined
         isbroker        TRUE                  WBI transport must have TRUE for isbroker
         receivers                             If a transport has receivers, must have receivers tag
            Receiver                           Name of Receiver is arbitrary
               parameters                      If parameters are provided, must have parameters tag
                   name    value               Each child is a name-value association
                   nnn     nnn                 Can have N number of parameters
            nnn                                Can have N receivers, each with parameters
         senders                               If a transport has senders, must have senders tag
            Sender                             Name of Sender is arbitrary
               parameters                      If parameters are provided, must have parameters tag
                   name    value               Each child is a name-value association
                   nnn     nnn                 Can have N number of parameters
            nnn                                Can have N senders, each with parameters
         eventstore                            Can only have one eventstore per transport, and is optional
            parameters                         If parameters are provided, must have parameters tag
                name    value                  Each child is a name-value association
                nnn     nnn                    Can have N number of parameters
            nnn                                Can have N number of transports

  logicmodules                                 Must have logicmodules tag, but can be empty below
      Name of logic module                     Arbitrary name for logicmodule
         parameters                            If parameters are provided, must have parameters tag
            name    value                      Each child is a name-value association
            nnn     nnn                        Can have N number of parameters
      nnn                                      Can have N number of logicmodules

  routingpaths                                 Must have routingpaths tag with at least  one working route
      routingpath1                             Arbitrary name for routingpath
          origin                               Each route must have an origin tag
             transport                         Each origin must have transport tag
             receiver                          Each origin must have receiver tag, but value can be empty
          destination                          Each route must have a destination tag
             transport                         Each destination must have transport tag
             sender                            Each destination must have sender tag, but value can be empty
          logicmodulerefs                      Each route must have a logicmodulerefs tag, but cannot have child
             one     logic module              Arbitrary name ("one" for order reminder); value must match logicmodule
             nnn     nnn                       Can have N number of logicmodule references
          includeobjects                       Each route must have an includeobjects tag, but cannot have child
             MyBusObjectQuery                  List of objects
             nnn                               Can have N number of objects supported
       nnn                                     Can have N number of routingpaths
```

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 Corporation
Department LZKS
11400 Burnet Road

Austin, TX 78758
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:

Index

Printed in USA