IBM WebSphere Business Integration Adapters

Adapter for Exchange User Guide

Adapter Version 1.1.x

IBM WebSphere Business Integration Adapters

Adapter for Exchange User Guide

Adapter Version 1.1.x

Note!

Before using this information and the product it supports, read the information in "Notices" on page 73.

19December2003

This edition of this document applies to adapter version 1.1.x and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

Contents

About this document

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

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

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 Adapter installations, and includes reference material on specific components.

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

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

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

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

Typographic 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, *`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 path names are relative to the directory where the connector is installed on your machine.
ProductDir	Represents the directory where the IBM WebSphere Business Integration Adapters product is installed. The default directory is WebSphereAdapters.

Summary of Changes

This chapter contains information about changes to *the Adapter for Exchange User Guide* for the current release.

New for Adapter for Exchange 1.1.0

This book has been updated to include the changes listed below:

- Files installed with the connector have changed.
- Changes in function to event listener configuration properties CwEventUserName and CwAgentUserName.
- Changed instructions for configuring MAPI profile.
- Added two new configuration properties: RecipientBusinessObj andRecurrencePatternBusinessObj.
- Revised Polling Error section to include additional possible error message.
- Adapter installation information has been moved from this guide. See Chapter 2 for the new location of that information.
- Contains updated standard configuration properties in Appendix A.
- Contains updated Connector Configurator section in Appendix B.
- Contains improved instructions for Starting and Stopping the adapter.
- Contains new information about starting multiple adapters.

Chapter 1. Overview of the adapter

This chapter describes the IBM WebSphere Business Integration adapter for Exchange and the associated system architecture.

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

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

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

For more information about the relationship of the integration broker to the adapter, see the *IBM WebSphere Business Integration System Administration Guide*(for WebSphere InterChange Server), *Implementing Adapters with WebSphere Message Brokers*, or*Implementing Adapters with WebSphere Application Server*.

In addition to the adapter framework and the application-specific component, the adapter for Exchange contains a component called Event Listener. This component is a COM+ event sink application that runs on the same machine as Microsoft Exchange Server.

This chapter contains the following sections:

Adapter architecture

The adapter for Microsoft Exchange Server enables the exchange of business objects between your integration broker and an Exchange Server.

Exchange Server is a powerful electronic messaging back end that supports a variety of email protocols. It can host calendars, contact lists, and many other types of files for individual users and for entire organizations through the use of public and private folders. An application adapter permits the exchange of data between Exchange Server and other applications.

Figure 1 on page 2 illustrates the general architecture of the connector.

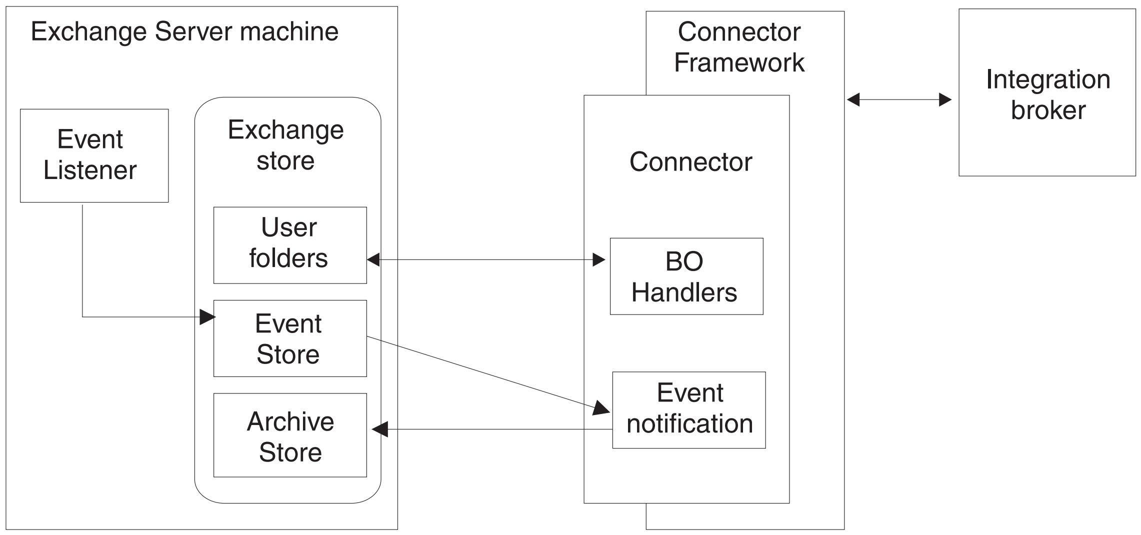


Figure 1. Exchange Server connector architecture

The application-specific component of the Exchange Server connector contains the event notification functionality and four business object handlers: MessageBOHandler, ContactItemBOHandler, TaskItemBOHandler, and AppointmentItemBOHandler.

The application-specific component performs the following tasks:

- Detects Exchange events (for example, the creation of a ContactItem business object)
- Archives Exchange events
- Builds application-specific business objects
- Processes integration broker requests
- Logs error and trace messages

This connector is single threaded. As a result, the connector cannot process integration broker requests and event notification tasks (including detecting and archiving events) concurrently; instead, these events are handled one at a time, in the order in which they occurred.

Event Listener

The adapter for Exchange Server contains an additional component called Event Listener. Unlike the connector itself, the Event Listener component is installed on the same machine as Exchange Server; it listens for all subscribed events (for example, the creation, modification, or deletion of a message) in Exchange Server and then logs them to the Event Store, where the connector can pick them up.

This component is installed as a COM+ event sink application. An *event sink* is an application that is launched by a defined trigger (in this case, the arrival of a subscribed event).

You must subscribe to changes by creating an event registration file in each folder for which events are going to be generated. The event registration is a hidden file that is bound to the event sink. When the Microsoft Information Store Service starts up, it searches for these hidden files and adds them to its list of folders to monitor. Whenever an item in a monitored folder is saved or deleted, the Information Store Service notifies the event sink that is associated with the event registration.

Event Listener performs the following tasks each time it receives a subscribed event:

- Retrieves its configuration properties.
- Extracts information from the event.
- Generates a unique ID for the event.
- Sends an email with event information to the CWEvent user specified during Event Listener configuration.
- Logs event messages to the Windows event log.

Data storage and access

Exchange Server organizes data into *stores*, which can be either public or private. These information stores can contain folders to further organize the data. Folders inside the private information store include individual user’s folders such as Inbox, Outbox, Contacts, and Tasks. Folders inside the public information store include those that are available to all members of an organization (for example, a Global Address List folder).

The connector for Exchange Server makes use of Exchange Server user folders, as well as the custom Event and Archive folders. The Event Store holds all of the events received by the Event Listener, and the Archive Store holds all of the archived events.

In an Exchange Server environment, user folders are accessed with either the Information Store Service or the Directory Service; these services are accessed with the Microsoft Mail API (MAPI). The connector for Exchange Server uses the Collaboration Data Objects (CDO) library as a wrapper for MAPI. Figure 2 illustrates this topology.

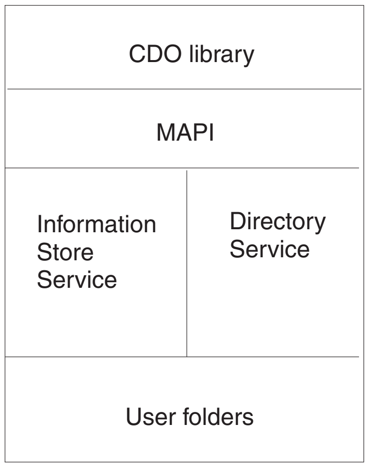
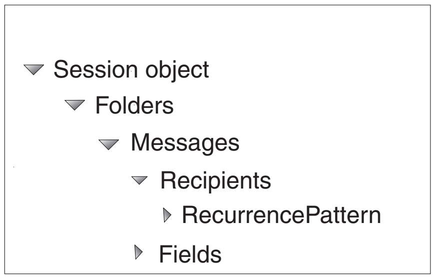


Figure 2. Using CDO to access user folders

Within Exchange Server, objects are organized into a tree structure, as shown in Figure 3.



The Session object is at the top of the hierarchy; you cannot access any other

Figure 3. Object Hierarchy within Exchange Server

objects until you have successfully logged in with the Session object's Logon() method.

All of the Exchange Server folders in the public and private information stores are contained under the Session object, and can be directly accessed by CDO.

Within each folder are messages. Messages are the fundamental business entities in Exchange Server, and include objects such as MailItem, ContactItem, and AppointmentItem. The CDO library provides direct access to messages.

MailItem messages can have recipient objects. The CDO library does not provide direct access to recipient objects; you must access them through the parent message object.

Each message has multiple field objects that define its content. For example, a ContactItem message object has FirstName, LastName, and MailingAddress fields.

How the connector works

The connector receives a business object from the integration broker, processes that object based on its active verb, and then sends a request for operation to the Exchange Server.

It also polls the event store for subscribed events. If a subscribed event is found, the connector builds a business object and populates it with data retrieved from an API call to the Exchange Server. The connector then publishes the business object to the integration broker.

The following steps illustrate the basic sequence of events when a connector starts, polls for events, and processes those events:

1. During connector startup, the init() method is invoked and the connector is logged onto Exchange Server.
2. The resetReceivedCount() method is called to set the eventsReceived property to zero (0).
3. The pollForEvents() method is called to poll the Event Store for subscribed events.
4. If the connector finds subscribed events, it calls the processEvents() method to handle all of the event processing. The processEvents() method does the following:
   - Calls the processEvent() method, which in turn does the following:

    - Calls fetchEvent()
    - Creates a new, empty business object
    - Sets the FolderID, MessageID, and StoreID fields on the new business object
    - If the event is a Create or Update:
      - Sets the verb to Retrieve
      - Calls doVerbFor() to populate the business object
      - Resets the verb to its original value (Create or Update)
    - If the event is a Delete, it sets the verb to Delete
  - Calls the gotApplEvent() method to publish the event to the integration broker
  - Calls the addEvent() method to increment the eventsReceived count and to compare the values of eventsReceived to pollQuantity to determine if further processing is needed
5. Depending on the archive behavior specified in the ArchiveProcessed property, the archiveEvent(), deleteEvent(), or setEventStoreStatus() methods can be called to perform the necessary event archiving.

The following sections contain additional information about these processes.

Business object processing

When the connector receives a request from an integration broker to perform an application operation, the connector processes hierarchical business objects recursively (that is, it performs the same steps for each child business object until it has processed all individual business objects). The order in which the connector processes child business objects and the top-level business object depends on whether the child business objects are contained with or without ownership and whether they are contained with single or multiple cardinality.

Note: The term *hierarchical business object* refers to a complete business object, including all of the child business objects it contains. The term *individual business object* refers to a single business object, independent of any child business objects that it contains or that contain it. The term *top-level business object* refers to the individual business object at the top of the hierarchy; it does not have a parent business object.

Business object retrieval

When an integration broker asks the connector to retrieve a hierarchical business object from the database, the connector attempts to return a business object that exactly matches the current database representation of that business object. In other words, all simple attributes of each individual business object returned to the integration broker match the value of the corresponding fields in the database. Also, the number of individual business objects in each array contained by the returned business object match the number of children in the database for that array.

To perform such a retrieval, the connector uses the primary key values in the top-level business object received from the integration broker. These key values are used to recursively descend through the corresponding data in the database.

Business object creation: When an integration broker asks the connector to create a hierarchical business object in the database, the connector performs the following steps:

1. Recursively creates each single-cardinality child business object contained with ownership into the database.
2. Processes each single-cardinality child business object contained without ownership.
3. Creates the top-level business object in the database.
4. Creates each single-cardinality child business object that stores the parent/child relationship in the child.
5. Creates each multiple-cardinality child business object.

Business object modification: When an integration broker asks the connector to update a hierarchical business object in the database, the connector performs the following steps:
1. Uses the primary key values of the source business object to retrieve the corresponding entity from the database.
2. Recursively updates all single-cardinality children of the top-level business object.
3. Updates all simple attributes of the retrieved business object except those whose corresponding attribute in the source business object contains the value CxIgnore.
4. Processes all arrays of the retrieved business object.

Business object deletion: When an integration broker asks the connector to delete a hierarchical business object from the database, the connector deletes only the top-level business object.

Processing application events

The creation, modification, or deletion of any Exchange Server object is considered an event. The event is placed into the Event Store. When the connector invokes the pollForEvents () method, the event is retrieved and processed. The Retrieve operation is based on key attributes in the business object (typically the EntryID attribute, and sometimes the StoreID attribute).

Create

When the connector finds a Create event in the Event Store, it creates a business object of the type specified by the event and sets the FolderID, StoreID, and MessageID fields with the values found in the event. The connector then retrieves the business object from the database and publishes it to the integration broker with the Create verb.

Update

When the connector finds an Update event in the Event Store, it creates a business object of the type specified by the event and sets the FolderID, StoreID, and MessageID fields with the values found in the event. The connector then retrieves the business object from the database and publishes it to the integration broker with the Update verb.

Delete

When the connector finds a Delete event in the Event Store, it creates a business object of the type specified by the event and sets the FolderID, StoreID, and MessageID fields to the values found in the event (all other attributes are set to CxIgnore).

The connector supports only physical delete operations that are triggered by Exchange Server.

Event handling

Whenever an Exchange Server object is created, updated, or deleted, the connector's Event Listener places the event in the Event Store. When the connector invokes the pollForEvents() method, the events in the Event Store are detected and subsequently processed.

The PollQuantity connector property specifies the number of events the connector retrieves from the event store. For each event the connector retrieves, the following tasks are performed:

- The connector updates the event status to IN_PROGRESS.
- The connector determines if the integration broker is subscribed to events generated for the business object, then does one of the following:
  - If there is a subscription, the connector performs the necessary Create, Update, or Delete action.
  - If there is no subscription, the connector updates the event status to UNSUBSCRIBED and archives the event according to the settings for the ArchiveProcessed connector property.

Table 1 lists the status values used for events.

Table 1. Event status

Event status	Numerical value for status	Description
READY_FOR_POLL	0	The event is ready to be picked up by the next poll call.
IN_PROGRESS	1	The connector has picked up the event and is processing it.
UNSUBSCRIBED	2	There is no subscription for this event.
SUCCESS	3	The connector successfully processed the event.
ERROR_PROCESSING_EVENT	-1	The connector encountered an error while processing the event.
ERROR_POSTING_EVENT	-2	The connector encountered an error while publishing the business object to the integration broker.
ERROR_OBJECT_NOT_FOUND	-3	The business object for which the event was created cannot be found.

Event notification

In order to use event notification, you must create two Exchange Server user folders: one for storing events, and one for archiving events. For instructions on creating and configuring these folders, see "Connector installation" on page 13.

Event archiving

The ArchiveProcessed connector property determines whether an event is archived after its status is updated. Table 2 on page 8 describes the settings for ArchiveProcessed.

Table 2. Using the ArchiveProcessed property

ArchiveProcessed value	Event status	Connector behavior
`true`	Event successfully processed	The connector archives the event with a status of SUCCESS. After the event has been archived, it is deleted from the Event Store.
	Event is unsuccessfully processed	The connector archives the event with a status of ERROR_POSTING_EVENT or ERROR_OBJECT_NOT_FOUND. After the event has been archived, it is deleted from the Event Store.
	Event is not processed because there is no subscription for the business object	The connector archives the event with a status of UNSUBSCRIBED. After the event has been archived, it is deleted from the Event Store.
`false`	Event is successfully processed	The event remains in the Event Store.
	Event is unsuccessfully processed	The event remains in the event store with a status of ERROR_POSTING_EVENT, ERROR_OBJECT_NOT_FOUND, or ERROR_PROCESSING_EVENT.
	Event is not processed because there is no subscription for the business object	The event remains in the event store with a status of UNSUBSCRIBED.

Event recovery

The connector can recover events after a system failure. During startup, the connector invokes the recoverInProgressEvents() method to search for any events that had a status of IN_PROGRESS when the previous connector session ended.

The InDoubtEvents connector property determines what type, if any, of event recovery is performed. Refer to "Application-specific configuration properties" on page 20 for more information on setting this property.

Guaranteed Event Delivery

The connector supports Guaranteed Event Delivery (GED) through the use of the connector framework's duplicate event elimination (DEE) feature and the setDEEID() method. As a result, an event is sent to the integration broker only once, even if the connector terminates after delivering the event to the broker but before updating the event's status in the Event Store.

In order to take advantage of GED, you must set three connector configuration properties, as shown in Table 3.

Table 3. Setting connector properties to enable GED

Property name	Description	Value required for GED
DuplicateEventElimination	Standard configuration property; when set to `True`, the event ID is stored in the monitor queue to prevent delivery of duplicate events.	`True`

Table 3. Setting connector properties to enable GED (continued)

Property name	Description	Value required for GED
MonitorQueue	Standard configuration property; specifies the JMS queue where the connector stores the processed event ID.	Must be set to the appropriate queue (for example, MONITORQUEUE)
InDoubtEvents	Connector-specific configuration property; specifies how to handle events that had a status of IN_PROGRESS when the connector terminated.	`Reprocess`

For more information on setting connector configuration properties, see Appendix A, “Standard configuration properties for connectors,” on page 41.

Transaction support

Transactional support is provided by the Exchange Server Extensible Storage Engine (ESE). The ESE is a transaction logging system that ensures data integrity after a system failure. Because the Event Store and Archive Store are placed in Exchange Server mailboxes, the ESE provides support for aborting or rolling back an event processing transaction.

If the Event Listener is unable to process events (either because it is disabled or because of a system error), the Information Store Service maintains the events and delivers them to the Event Listener once it has been restored.

Adapter environment

Before installing, configuring, and using the adapter, you must understand its environment requirements as described in the following sections:

- “Broker compatibility”
- “Installation prerequisites” on page 10
- “Processing locale-dependent data” on page 10

Broker compatibility

The adapter framework that an adapter uses must be compatible with the versions of the integration brokers with which the adapter is communicating. Version 1.1 of the adapter for Exchange is supported on the following adapter framework and with the following integration brokers:

- **Adapter framework**: WebSphere Business Integration Adapter Framework versions 2.1.0, 2.2.0, 2.3.0, 2.3.1, and 2.4.0.
- **Integration brokers**:
  - WebSphere InterChange Server, versions 4.2.x
  - WebSphere MQ Integrator, version 2.1.0
  - WebSphere MQ Integrator Broker version 2.1.0
  - WebSphere Business Integration Message Broker 5.0 with CSD02
  - WebSphere Application Server Enterprise, version 5.0.2, with WebSphere Studio Application Developer Integration Edition, version 5.0.1

See the *Release Notes* for any exceptions.

Note: For instructions on installing the integration broker and its prerequisites, see the following documentation. For WebSphere InterChange Server (ICS), see the *System Installation Guide for UNIX* or *for Windows*. For message brokers (WebSphere MQ Integrator Broker, WebSphere MQ Integrator, and WebSphere Business Integration Message Broker), see *Implementing Adapters with WebSphere Message Brokers*, and the installation documentation for the message broker. Some of this can be found at the following Web site:

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

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

Installation prerequisites

You must have the following software installed and configured on the machine on which you plan to install the connector:

- Microsoft Windows 2000

  Note: Beginning with version 1.1, the Adapter for Exchange is no longer supported on Microsoft Windows NT.
- Microsoft Collaboration Data Objects (CDO) library version 1.21

  The CDO library is available with Exchange Server 2000 and Outlook 2000.
- Microsoft Outlook COM library
- Microsoft automation controller

  The automation controller is available in the Microsoft Office suite of products, and is typically installed as part of an application like Microsoft Excel or Microsoft Access.
- Adapter Application Development Kit (ADK)
- One of the integration brokers listed in "Broker compatibility" on page 9.

Note that this guide discusses only the installation of the adapter. Refer to the appropriate product documentation when installing any of the prerequisite software listed in this section.

Installing the adapter and related files

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

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

Processing locale-dependent data

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

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

Chapter 2. Installing the adapter

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

- "Connector installation" on page 13
- "Event Listener installation" on page 14

Connector installation

This section describes the tasks you must perform to install the connector and its associated business objects. Refer to "Adapter environment" on page 9 for software prerequisites and compatibility.

After you have installed all prerequisite software on the machine, you can install the connector and the business objects.

For complete instructions on installing the adapter (which includes the connector and business objects), refer to the *Installation Guide for WebSphere Business Integration Adapters*, located in the WebSphere Business Integration Adapters Infocenter at the following web site:

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

The following table describes the files that are installed with the connector. In the table, *ProductDir* represents the directory in which you have installed the IBM WebSphere Business Integration Adapter software.

Table 4. Files installed with the connector

Directory	Installed files
ProductDir\connectors\Exchange	start_Exchange.bat—The connector startup file BIA_Exchange.jar—The connector code BIA_COMProxy.dll—The COM wrapper library
ProductDir\connectors\Exchange\samples	BIA_Exchange.cfg—The sample connector configuration file
ProductDir\connectors\messages	BIA_ExchangeConnector.txt—The connector message file
ProductDir\repository\Exchange\2000	AppointmentItem.in, ContactItem.in, MailItem.in, Recipient.in, RecurrencePattern.in—Sample business objects

Installing business objects

The adapter provides a number of sample business objects that you can use when developing your own business objects. To access the sample business objects, do the following:

1. Navigate to the *ProductDir*\repository\Exchange\2000 directory.

2. Use the Connector Configurator to open the .in files and save them to your broker repository.
3. Ensure that the connector configuration property RepositoryDirectory specifies the directory in which you just placed the sample business objects.

Event Listener installation

This section describes the tasks you must perform to install the Event Listener. It also lists the prerequisite software you must have on the machine before you install the Event Listener.

Installation prerequisites for Event Listener

The following software must be installed and configured on the machine where you plan to install Event Listener:

- Microsoft Windows 2000
- Microsoft Exchange Server 2000
- Microsoft Collaboration Data Objects (CDO) library version 1.21, including the exoledb.dll executable

  The CDO library is available with Exchange Server 2000 and Outlook 2000.
- Microsoft Exchange Information Store Service and Directory Service

  These services are available in Microsoft Exchange Server 2000.
- One of the following tools for registering events:
  - Exchange Explorer—This GUI must be downloaded from the Microsoft Web site.

    Note: Exchange Explorer requires Microsoft Visual Basic 6.0 or later to be installed on the machine.
  - regevent.vbs—This utility is available with Microsoft Visual Studio, or as a download from the Microsoft Web site.

Installing Event Listener

You must install the Event Listener executable file (BIA_Exchange.dll) and configuration file (BIA_Exchange.ini) on the machine where Exchange Server is running, as follows:

1. Insert the WebSphere Business Integration Adapter CD-ROM and navigate to the CD's \connectors\Exchange\dependencies\eventListener directory.
2. Copy the BIA_Exchange.dll file from the CD-ROM to your Exchange Server machine. The recommended location for the file is C:\Program Files\Exchsrvr\bin, but you can install the file elsewhere. Ensure that the directory in which you plan to install BIA_Exchange.dll has adequate security to prevent users from accidentally deleting the file.
3. Copy the BIA_Exchange.ini file from the adapter CD-ROM (\connectors\Exchange\dependencies\eventListener) to the C:\WINNT directory on the Exchange Server machine.

Chapter 3. Configuration tasks

This chapter describes the configuration tasks necessary to enable the adapter to communicate with Exchange Server. It contains the following sections:

- "Configuration tasks required for Event Listener" on page 15
- "Configuration tasks required for the connector" on page 18
- "Registering for events" on page 21
- "Starting the adapter" on page 24
- "Stopping the adapter" on page 26
- "Creating multiple instances of adapters on one server" on page 27

Configuration tasks required for Event Listener

The Event Listener component is an event sink that is installed as a COM+ component to Exchange Server. You must perform the following configuration tasks on the machine running Exchange Server:

1. Create a user account for running event sinks and assign it the appropriate permissions. See "Setting up a user account and permissions for event sinks."
2. Register the Event Listener executable (BIA_Exchange.dll) on the Exchange Server machine. See "Registering Event Listener."
3. Create a COM+ application for Exchange Server and add the Event Listener component to it. See "Creating a COM+ application" on page 16.
4. Modify the Event Listener configuration file. See "Modifying the configuration file" on page 17.
5. Enable the Windows Event Viewer to display Event Listener messages. See "Configuring the Windows Event Viewer for use with Event Listener" on page 18.

Setting up a user account and permissions for event sinks

An Exchange Server event sink requires authentication before it can execute. Therefore, you must specify a Windows user account with permissions to execute the event sink. This document refers to this account by the general name *Exchange_User*.

Refer to the Microsoft Knowledge Base article 262054 (*XADM: How to Get Service Account Access to All Mailboxes in Exchange 2000*) for detailed information on creating the Windows user account.

Registering Event Listener

The Event Listener component must be registered on the Exchange Server machine with the Regsvr32 utility, as follows:

1. Click **Start —> Run**. The Windows Run dialog box is displayed.
2. In the **Open** field, type the following:

   ```
   regsvr32 installation_directory\BIA_Exchange.dll
   ```

   where *installation_directory* specifies the location in which you installed the BIA_Exchange.dll file (for example, C:\Program Files\Exchsrvr\bin).

Windows displays a dialog box to confirm the successful registration of the Event Listener file.

Creating a COM+ application

Perform the following steps to create a COM+ application on your Exchange Server machine. This COM+ application is needed to run the Event Listener event sink.

1. Click **Start —> Programs —> Administrative Tools —> Component Services**. The Component Services dialog box is displayed.
2. Expand the Component Services tree by clicking the plus sign (+) next to it.
3. Right-click the COM+ Applications folder under Component Services, and then click **New —> Application**. The COM Application Installation Wizard launches.
4. Click **Next** on the Welcome dialog box. The Install or Create a New Application dialog box is displayed.
5. Click **Create an empty application**. The Create Empty Application dialog box is displayed.
6. Type `BIA_Exchange` in the **Enter a name for the new application** field.
7. In the Activation type box, click **Server application**, and then click **Next**. The Set Application Identity dialog box is displayed.
8. In the Account box, click **This user**, and then click **Browse**. The Select User or Group dialog box is displayed.
9. Select the name of the Exchange user you set up in "Setting up a user account and permissions for event sinks" on page 15, and then click **OK**. The user name is inserted in the **User** field.
10. In the **Password** field, type the password for the Exchange user.
11. Retype the password in the **Confirm password** field, and then click **Next**.
12. Click **Finish** to create the application and exit the wizard. The COM+ Applications folder now contains a BIA_Exchange application folder, with corresponding Components and Roles subdirectories underneath.

Adding Event Listener to the COM+ application

After you have created the new COM+ application, you must add the Event Listener component to it, as follows:

1. Navigate to the BIA_Exchange application folder in the Component Services dialog box.
2. Right-click the Components folder, and then click **New**. The COM Application Installation Wizard launches.
3. Click **Next** on the Welcome dialog box. The Install or Create a New Application dialog box is displayed.
4. Click **Install new component(s)**, and then click **Next**. The Install new components dialog box opens.
5. In the Files to install window, click **Add**.
6. Browse for the BIA_Exchange.dll file you installed on the machine, and then click **OK**. The wizard analyzes the file and lists the Events component in the Components found window.
7. Click **Next**, and then click **Finish** to exit the wizard. The BIA_Exchange.Events.1 component is placed in the BIA_Exchange application's Components folder.

Modifying the configuration file

The Event Listener requires a configuration file called BIA_Exchange.ini. This file contains configuration properties used by the Event Listener; Table 5 describes the properties you can need to edit or set.

Table 5. Event Listener configuration properties

Property	**Description**	**Default value**
CwEventID	Identifies the last event ID used.	zero (0)
CWEventUser	Specifies the alias name of the Event Listener user. All events are stored in this user’s Inbox.	CwEvent
CwAgentUsername	Specifies the full name of the user account whose events are to be filtered out. The Event Listener ignores events caused by this user.	CwAgent
CwAgentUserGUID	Specifies the GUID of the user account associated with the connector (for example, `{31A42120 - 49B8 - 4B8F - A135 - 915C1AFF2BEC}`). The Event Listener ignores events caused by this user.	N/A
CwEventUsername	Specifies the full name of the Event Listener user. The Event Listener ignores all events caused by this user.	CwEvent
EventsTraceLevel	Specifies the level of trace messages for the Event Listener.	0 (no tracing)

Make the following modifications to the file:

- Modify the CwEventUser entry to specify the name of the Exchange user account you set up in “Setting up a user account and permissions for event sinks” on page 15. When the Event Listener receives notification of an event, it sends an email message with event information to this user account.
- Set the CwAgentUserGUID entry to specify the GUID of the user account associated with the connector.

  Note: A convenient way of obtaining this value is to trigger the Event Listener and then examine its event log entries. For example:

  1. Set the EventsTraceLevel to 5.
  2. Create an event registration in the Tasks folder (See“Registering for events” on page 21.)
  3. Create a new task using Outlook. This will trigger the Event Listener.
  4. Examine the Application Log using the Event Viewer.

  The thirteenth message logged by BIA_Exchange will contain the GUID.
- If additional filtering is desired, modify the CwEventUsername and CWAgentUserName entries to specify the display name of the Exchange user accounts to be filtered out.

Configuring the Windows Event Viewer for use with Event Listener

The Event Listener logs its error and trace messages to the Windows Event Viewer. You must manually edit the Windows Registry to enable the Windows Event Viewer to find and properly display Event Listener messages.

Caution

Incorrectly editing the Windows Registry can cause problems with your machine. Use caution when editing the Registry. If you are unfamiliar with the Registry, seek assistance from a Windows system administrator.

Perform the following steps to update the Windows Registry:

1. If the Windows Event Viewer is open, close it now.
2. Click **Start —> Run**, and type `regedit` in the Run dialog box. The Windows Registry opens.
3. Navigate to and highlight the key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\Application.
4. Right-click to bring up the context menu, and then click **New —> Key**. A new key is added to the Registry under the Application key.
5. Name the key `BIA_Exchange`.
6. Right-click in the right pane of the Windows Registry to bring up the context menu. Click **New —> String Value**. A new entry is added.
7. Name the new entry `EventMessageFile`.
8. Double-click the EventMessageFile entry to open the Edit String dialog box.
9. In the **Value data** field, specify the full path name for the BIA_Exchange.dll file (for example, *ProductDir* \connectors\Exchange\dependencies\eventListener\BIA_Exchange.dll), and then click **OK**.
10. Right-click in the right pane of the Windows Registry to bring up the context menu. Click **New —> DWORD Value**. A new entry is added to the key.
11. Name the new entry `TypesSupported`.
12. Double-click the TypesSupported entry to open the Edit DWORD Value dialog box.
13. Type 7 in the Value data field, and leave the default base (Hexadecimal) selected. Click **OK**.
14. Click **Registry —> Exit** to close the Windows Registry.

Configuration tasks required for the connector

You must perform the following configuration tasks for the connector:

- Create a Microsoft Outlook profile for the connector. See "Configuring the connector MAPI profile" on page 19.
- Create the Archive Store. See "Creating the Archive Store" on page 19.
- Set connector configuration properties. See "Setting connector configuration properties" on page 19.

Configuring the connector MAPI profile

A Microsoft Outlook profile must be created on the machine where the Outlook client and the Exchange connector are installed. This profile ensures that all necessary Exchange folders are configured correctly.

Create the Outlook profile for the connector as follows:

1. Start the Outlook client on the machine where the connector is installed.
2. When Outlook prompts you to specify a mailbox, enter the name of the Exchange user you created in "Setting up a user account and permissions for event sinks" on page 15.
3. When prompted by Outlook, register Outlook as the default client for e-mail messages and contacts.

Outlook creates a profile that is named Ms Exchange Settings by default.

Note: The value you specify for the connector's ProfileName configuration property must exactly match the Outlook MAPI profile name.

Creating the Archive Store

When you created the Outlook profile in "Configuring the connector MAPI profile," the Event Store was automatically created. You must now create the Archive Store to hold archived events, as follows:

1. Open the Outlook client and select **Outlook Today — [Mailbox -** *user_name*].
2. Create a new folder with the following values. Refer to your Outlook documentation for more information on creating new folders.

Table 6. Field values for the Archive Store folder

Field name	Value to enter
Name	`Archive` **Note:** This value must be used for the connector's ArchiveFolder configuration property.
Folder contains	`Mail Items`
Select where to place the folder	`Mailbox—`*`user_name`*, where *user_name* is the name of the account you want to use for the Archive Store. You can use any account, as long as the Exchange user has access to it.

Setting connector configuration properties

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

Standard configuration properties

To configure standard connector properties, use the Connector Configurator tool. Details are given in Appendix B, "Connector Configurator," on page 57. This tool provides a graphical user interface for configuring the connector. Click on the **Standard Config Properties** tab to add or modify configuration properties.

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

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

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

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

Application-specific configuration properties

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

To configure these properties, use Connector Configurator. Click the **Application Config Properties** tab to add or modify configuration properties. For more information, see Appendix B, "Connector Configurator," on page 57.

Table 7 lists the application-specific configuration properties for the connector, along with their descriptions and possible values.

Table 7. Application-specific configuration properties for Exchange Server

Property	Description	Possible values	Default value	Required?
ArchiveFolder	Fully qualified name of directory where events are archived	The full path name for the event archive directory (for example, Mailbox - User1/Archive)		Yes
ArchiveProcessed	Specifies how events are archived.	true, false For a detailed description of archive behavior, see "Event archiving" on page 7.	true	Yes
DateFormat	Specifies the format of Date attributes in the business object. The format is based on the java.text.SimpleDateFormat class	Any valid combination of date symbols (for example, MM/dd/yyyy HH:mm:ss). Common symbols include: • y—year • M—month • d—day • h—hour in A.M. or P.M. (1—12) • H—hour in the day (0—24) • m—minute • s—second • S—millisecond • a—designation of A.M. or P.M.	MM/dd/yyyy HH:mm:ss	Yes

Table 7. Application-specific configuration properties for Exchange Server (continued)

Property	Description	Possible values	Default value	Required?
EventFolder	Fully qualified name of directory where events are stored	The full path name for the event directory (for example, Mailbox - User1/Inbox)		Yes
InDoubtEvents	Specifies how to handle events that were IN_PROGRESS when the connector terminated	• FailOnStartup—A fatal error is logged, email notification is sent, and FAIL is returned • Reprocess—The events' status is changed to READY_FOR_POLL so they can be picked up during the next poll • LogError—An error is logged • Ignore—The events are ignored	Reprocess	Yes
ProfileName	Name of MAPI profile used to log on to Exchange Server	Any valid MAPI profile name		Yes
RecipientBusiness Obj	Determines the name of the Recipient child business object	any text string	Recipient	No
RecurrencePattern BusinessObj	Determines the name of the RecurrencePattern child business object t	any text string	RecurrencePattern	No

Registering for events

You must create an event registration for each Exchange folder in which events are generated. The event registration is stored in a hidden file that is bound to the Event Listener; when the Microsoft Information Store Service starts, it searches for event registration files and adds the specified folders to the list of monitored folders.

You can create an event registration for a single folder (shallow registration), for a parent folder and all of its subfolders (deep registration), or for all of the folders in an organization (store-wide registration).

Event registrations are easily created with the Exchange Explorer GUI. They can also be created with the RegEvent.vbs utility. The following sections describe how to create event registrations with these tools.

Creating event registrations with Exchange Explorer

Exchange Explorer is a GUI for creating event registrations. It is available as a download from the Microsoft Web site.

Tip

You can locate the Exchange Explorer download by specifying the following search criteria on the Microsoft download page (www.microsoft.com/downloads):

- Specify that you want to search for Exchange products, and specify the keyword `SDK`.
- When the search results are returned, select the Exchange SDK Tools category.
- Locate and download the file exchangesdktools.exe. Follow the instructions provided on the Microsoft Web site for downloading and installing.

Start Exchange Explorer by clicking **Start —> Programs —> Exchange SDK —> Exchange SDK Development Tools —> Exchange Explorer**. When you are prompted to log on in the Authorization dialog box, enter the following values:

- User Name—The name of the user account where events are to be registered
- Password—The password for the user account
- Exchange store URL—http://*machine_name*/exchange/*user_name*/, where *machine_name* specifies the name of the machine running Exchange Server and *user_name* specifies the value you entered in the **User Name** field.

Creating a shallow event registration

Perform the following steps to create a shallow event registration for a folder:

1. Ensure that you have started Exchange Explorer and logged on.
2. In the Exchange Store Hierarchy pane, select the folder you want to register.
3. Click **File —> Add Event Registration**. The Exchange Store Event Sink Registration Wizard opens, and prompts you to choose a name for the event registration.
4. In the **Enter a name for this event sink registration field**, type the name you want to use, and then click **Next**.
5. The wizard asks you to specify the type of events you want to register for. Click **Synchronous**, and then click **Next**.
6. The wizard asks you to specify the types of synchronous events you want to register for. Click both **OnSyncSave** and **OnSyncDelete**, and then click **Next**.
7. The wizard asks you to specify the scope of your event registration. Click **Shallow**, and then click **Next**.
8. The wizard asks you to specify the type of event sink you are registering. Click **COM Event Sink**, and then click **Next**.
9. The wizard asks you to specify the event sink class. In the **What is the ProgID of your event sink class?** field, type `BIA_Exchange.Events.1`, and then click **Next**.
10. The wizard prompts you to verify the choices you have made. When you are satisfied with your event registration, click **Finish** to exit the wizard.

The event registration is created. Every time an item is created, updated, or deleted in the registered folder, an event is sent to Event Listener.

Note: If you encounter an error creating the registration item, see Chapter 5, "Troubleshooting and error handling," on page 35.

Creating a deep event registration

Perform the following steps to create a deep event registration (the parent folder and all of its subfolders are registered):

1. Ensure that you have started Exchange Explorer and logged on.
2. In the Exchange Store Hierarchy pane, select the folder you want to register.
3. Click **File —> Add Event Registration**. The Exchange Store Event Sink Registration Wizard opens, and prompts you to choose a name for the event registration.
4. In the **Enter a name for this event sink registration field**, type the name you want to use, and then click **Next**.
5. The wizard asks you to specify the type of events you want to register for. Click **Synchronous**, and then click **Next**.
6. The wizard asks you to specify the types of synchronous events you want to register for. Click both **OnSyncSave** and **OnSyncDelete**, and then click **Next**.
7. The wizard asks you to specify the scope of your event registration. Click **Deep**, and then click **Next**.
8. The wizard asks you to specify the type of event sink you are registering. Click **COM Event Sink**, and then click **Next**.
9. The wizard asks you to specify the event sink class. In the **What is the ProgID of your event sink class?** field, type `BIA_Exchange.Events.1`, and then click **Next**.
10. The wizard prompts you to verify the choices you have made. When you are satisfied with your event registration, click **Finish** to exit the wizard.

The event registration is created. Every time an item is created, updated, or deleted in the registered folder or any of its subfolders, an event is sent to Event Listener.

Note: If you encounter an error creating the registration item, see Chapter 5, “Troubleshooting and error handling,” on page 35.

Creating a store-wide event registration

Store-wide (or global) event registration for private mailbox stores can be done only in the following folder, where *domain* specifies your email domain and *GUID* specifies the SystemMailbox global unique ID: file://./backofficestorage/*domain*/mbx/SystemMailbox{*GUID*} /StoreEvents/GlobalEvents.

In addition, you must have the appropriate permissions to create a store-wide event registration. Global events can be registered only by a member of the Administrator or Domain Administrators group, or by a user in the Exchange Administrators role.

Perform the steps described in “Creating a shallow event registration” on page 22 or “Creating a deep event registration” to create the appropriate store-wide event registration.

Creating event registrations with RegEvent.vbs

The RegEvent.vbs utility can create shallow, deep, and store-wide event registrations. For detailed information on using the RegEvent.vbs utility to create shallow and deep event registrations, refer to the Microsoft Developer Network (MSDN) Library and Knowledge Base (www.msdn.microsoft.com), and search for the keyword `RegEvent`.

Creating a store-wide event registration with RegEvent.vbs

Store-wide (or global) event registration for private mailbox stores can be done only in the following folder, where *domain* specifies your email domain and *GUID* specifies the SystemMailbox global unique ID: file://./backofficestorage/*domain*/mbx/SystemMailbox{*GUID*} /StoreEvents/GlobalEvents.

In addition, you must have the appropriate permissions to create a store-wide event registration. Global events can be registered only by a member of the Administrator or Domain Administrators group, or by a user in the Exchange Administrators role.

Enter the following command to create a store-wide event registration:

```
cscript RegEvent.vbs Add "OnSyncSave;OnSyncDelete" \
BIA_Exchange.Events.1 "file://./backofficestorage/domain/mbx \
/SystemMailbox{GUID}StoreEvents/GlobalEvents/globalSynchReg.eml" \
-m ANY
```

Starting the adapter

To start the adapter for Exchange Server, you must:

1. Start the connector.
2. Start the Event Listener.

Starting the connector

Startup script and methods

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

```
ProductDir\connectors\connName
```

where *connName* identifies the connector, and *ProductDir* is the directory in which you installed the product. For the Exchange adapter, this file is `ProductDir\connectors\Exchange\start_Exchange.bat.`

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

- From the **Start** menu

  Select **Programs>IBM WebSphere Business Integration Adapters>Adapters>Connectors**. By default, the program name is "IBM WebSphere Business Integration Adapters". However, it can be customized. Alternatively, you can create a desktop shortcut to your connector.
- From the command line
  - `start_connName connName brokerName [-cconfigFile ]`

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

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

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

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

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

Note: Some WebSphere Business Integration adapters can be run as a Windows service. However, the adapter for Exchange must be started from a command prompt. It cannot be started as a Windows service. It can be started as a Windows task.

For more information on how to start a connector, including the command-line startup options, refer to one of the following documents:
- For WebSphere InterChange Server, refer to the *System Administration Guide*.
- For WebSphere message brokers, refer to *Implementing Adapters with WebSphere Message Brokers*.
- For WebSphere Application Server, refer to *Implementing Adapters with WebSphere Application Server*.

Note that the connector logs onto the Exchange Server with a MAPI profile that must already exist on the machine where the connector is started. This profile directs the Outlook client to the name of the appropriate mailbox. If the user who starts the connector does not have access to this mailbox, the system prompts the user for a login name and password.

Tasks performed during connector startup

When the connector is started, it performs the following tasks:
- Retrieves configuration information
- Logs on to Exchange Server with the retrieved configuration information
- Retrieves the supported business object definitions
- Retrieves a list of event subscriptions the connector supports
- Returns the connector version
- Returns a pointer to the business object handler
- Optionally, polls the event store based on priorities
- Builds a business object for any subscribed event found during startup

Starting Event Listener

By default, the Event Listener is started automatically by the Information Service when the trigger (the arrival of a registered event) occurs. If you need to start the Event Listener manually, however, use the Windows Component Services Administrative Tool, as follows:
1. Click **Start —> Programs —> Administrative Tools —> Component Services**. The Component Services dialog box opens.
2. Expand the Component Services and COM+ Applications folders.
3. Right-click the BIA_Exchange application in the COM+ Applications folder to bring up the context menu, then click **Start**.

Stopping the adapter

To stop the adapter for Exchange Server, you must:

1. Stop the connector.
2. Stop the Event Listener.

Stopping the connector

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

- If you started the connector from the command line, with its connector startup script:
  - Invoking the startup script creates a separate "console" window for the connector. In this window, type "Q" and press Enter to stop the connector.
- From Adaptor Monitor (WebSphere Business Integration Adapters product only), which is launched when you start System Manager

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

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

Stopping the Event Listener

By default, the Event Listener is shut down automatically after it has been idle for a specified period of time (the default value is three minutes). You can change this timeout value, or specify that the Event Listener should run until it is manually shut down. To specify shutdown options, do the following:

1. Open the Windows Component Services Administrative Tool (click **Start —> Programs —> Administrative Tools —> Component Services**).
2. Expand the Component Services and COM+ Applications folders.
3. Right-click the BIA_Exchange application in the COM+ Applications folder to bring up the context menu, then click **Properties**.
4. Navigate to the Advanced tab of the Properties dialog box.
5. If you want to prevent the Event Listener from shutting down automatically, click **Leave running when idle**.
6. If you want to specify a timeout value different from the default, enter the value (in minutes) in the **Minutes until idle shutdown** field.
7. Click **OK**.

If you want to manually shut down the Event Listener, do the following:

1. Open the Windows Component Services Administrative Tool (click **Start —> Programs —> Administrative Tools —> Component Services**).
2. Expand the Component Services and COM+ Applications folders.
3. Right-click the BIA_Exchange application in the COM+ Applications folder to bring up the context menu, then click **Shut down**.

Creating multiple instances of adapters on one server

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

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

Create a new directory

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

```
ProductDir\connectors\connectorInstance
```

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

```
ProductDir\repository\connectorInstance
```

Create business object definitions

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

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

   ```
   ProductDir\repository\initialConnectorInstance
   ```

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

Create a connector definition

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

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

Create a start-up script

To create a startup script:

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

   dirname
2. Put this startup script in the connector directory you created in "Create a new directory" on page 27.

3. Create a startup script shortcut (Windows only).
4. Copy the initial connector’s shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

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

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

Chapter 4. Understanding business objects

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

The chapter contains the following sections:

Defining metadata

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

A metadata-driven adapter handles each business object that it supports according to the metadata encoded in the business object definition. This enables the adapter to handle new or modified business object definitions without requiring modifications to the code.

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

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

Note: If you modify the Outlook model to add fields to a business object, you must then use Business Object Designer to add the new fields to the business object definition.

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

Overview of business object structure

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

- A type name
- Supported verbs
- Attributes

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

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

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

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

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

Business objects for Exchange Server

The adapter for Exchange Server supports application-specific business objects based on the Exchange Server for Outlook object model. The supported business objects include the following:

- AppointmentItem—Represents a one-time or recurring appointment in the Calendar folder.
- ContactItem—Represents a person in your Contacts folder.
- MailItem—Represents a mail message in a mailbox folder (for example, Inbox). The MailItem object is the default business object.
- TaskItem—Represents a task in the Tasks folder. A task can be assigned, delegated, or self-imposed, and must be performed within a specified time frame.
- Recipient—Represents a single address to which a message is addressed or sent.
- RecurrencePattern—Represents the recurrence pattern of an AppointmentItem or TaskItem business object.

Note: Business object definitions may be updated in new releases. Refer to the Release Notes to determine if this is the case. If you are running a previous version of this adapter, you will need to refresh any updated business object definitions. To do so, load the file into the BO Designer (refer to "Connector installation" on page 13 for the directory path) and save to your broker repository.

Note that the current release of the adapter does not support the Attachment business object.

If the Outlook object has no children, the WebSphere Business Integration Adapter business object represents the Outlook business entity. For example, an appointment without a recurrence pattern is represented by the AppointmentItem business object.

If the Outlook object has children, then the WebSphere Business Integration Adapter business object represents the parent business entity and the business object's attributes references the children. For example, a recurring appointment would be represented by the parent AppointmentItem business object and the RecurrencePattern child business object. The RecurrencePattern object is referenced by an attribute in the parent AppointmentItem business object.

Simple attributes in a WebSphere Business Integration Adapter business object correspond to the fields in the matching Outlook business entity.

Required fields for business objects

Table 8 describes the fields required for business objects.

Table 8. Required fields for business objects

Field	Notes
StoreID	Required for all business objects
MessageID	Required for business objects with an Update or Delete verb
FolderID	Required for business objects with a Create verb

Adding fields to a business object

You can add user-defined fields to any of the supported business objects through Microsoft Outlook. For example, you can add a field called Access_Passcode to a MeetingItem business object.

If you use Outlook to add a field to a business object, you must modify that business object's definition in Business Object Designer. Follow these guidelines when adding user-defined fields:

- Ensure that the attribute type matches that of the type property library.
- Ensure that the attribute's application-specific information contains the type library property name (for example, Access_Passcode) and its access mode (for example, Get/Put).

For more information on using Business Object Designer, see the *WebSphere Business Integration Adapters Business Object Development Guide*.

Attribute properties

The connector has various properties that you can set on its business object attributes. Table 9 shows the properties for simple attributes.

Table 9. Simple attributes for business objects

Attribute	Description
Name	Specifies the name of the attribute
Type	Specifies the data type of the attribute (boolean, date, integer, or string)
MaxLength	Specifies the maximum length allowed for attributes that have a string data type
IsKey	Specifies whether the attribute is a key field
IsForeignKey	N/A
IsRequired	Specifies whether the attribute is a required field in the business object

Table 9. Simple attributes for business objects (continued)

Attribute	Description
AppSpecificInfo	Specifies the Exchange-specific information used to access this attribute.
DefaultValue	If specified, this value is used by the connector if one is not set in the inbound business object, and if the connector's UseDefaults property is set to true.

Table 10 shows the properties for child object attributes.

Table 10. Attributes for child business objects

Attribute	Description
Name	Specifies the name of the child business object
Type	Specifies the data type of the child business object. The following data types are valid: • Recipient • RecurrencePattern
ContainedObjectVersion	Specifies the version of the child business object.
Relationship	If the child business object is a container attribute, the Relationship value is set to Containment.
IsKey	N/A
IsForeignKey	N/A
IsRequired	Specifies whether the attribute is required for the business object.
AppSpecificInfo	N/A
Cardinality	Specifies the number of child records that can be chosen for a FolderID record. The value can be 1 (if only one record can be chosen) or N (if multiple records can be chosen).

Application-specific information

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

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

Application-specific text for business objects

The application-specific information for the business object handles one key property for the adapter: the Outlook message class. The OutlookMessageClass parameter specifies the name of the Exchange business object that corresponds to the current WebSphere Business Integration Adapter business object. Valid values include the following:

- IPM.Appointment

- IPM.Contact
- IPM.Note
- IPM.Task

Application-specific text for simple attributes

The connector uses application-specific text for simple attributes, as described in Table 12.

Table 11. Application-specific text for simple attributes

Parameter	Description
CdoPropTagName	Set the value to `CdoPropTagValue` to specify the CDO property tag name and value that correspond to the current attribute.
PropertyName	Set the value to `PropertyNameValue` to specify the Outlook property name and value that correspond to the current attribute.
Access	Specifies whether the current attribute is read/write (`Get/Put`) or read-only (`Get`).

Application-specific text for verbs

The connector uses application-specific text for verbs, as described in Table 12:

Table 12. Application-specific text for verbs

Parameter	Description
SetToNullOnCxIgnore	Determines the behavior when CxIgnore and CxBlank are encountered while processing a service call request. This value is either true or false. If true and CxIgnore or CxBlank is encountered, then the value is set to null. If false and CxIgnore is encountered, then the value is not changed. If false and CxBlank is encountered, then the value is set to null.

Chapter 5. Troubleshooting and error handling

This chapter describes how the adapter handles errors. The chapter contains the following sections:

- "Error handling in the connector"
- "Error handling in Event Listener" on page 37
- "Tracing messages" on page 37

Error handling in the connector

The connector logs any abnormal condition that it encounters during processing, regardless of the trace level. It writes the error text to the connector log file; the name and location of this file are set by the LogFileName connector configuration property.

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

For a complete list of error messages, refer to the BIA_ExchangeConnector.txt file installed in the *ProductDir*\connectors\messages directory.

Table 13 describes some of more common errors and how the connector handles those errors.

Table 13. Connector errors

Error description	Error type	Error handling
The Exchange Server error MAPI_E_LOGON_FAILED is returned during Session login. The error occurs because the value for UserProfile or UserPassword is invalid, or because Exchange Server is not running properly.	Fatal error	The connector detects the error during initialization. The connector logs a fatal error and then terminates.
The connector detects a connection error when processing a business object service call request.	Fatal error	The connector logs a fatal error, sends the return code APPRESPONSE_TIMEOUT to trigger email notification, and then terminates.
The connector detects that the connection is lost while polling for events. The Exchange Server error MAPI_E_NETWORK_ERROR is returned while attempting to retrieve an event from the Event Store.	Fatal error	The connector tries to retrieve the event as many times as specified in the RestartRetryCount configuration property. If the connection has not been restored, the connector logs a fatal error, sends the return code APPRESPONSE_TIMEOUT to trigger email notification, and then terminates.

Table 13. Connector errors (continued)

Error description	Error type	Error handling
The connector attempts to retrieve an unsubscribed event.	Warning	The connector logs the warning and stores the event in the Archive Store with a status of UNSUBSCRIBED.
The connector encounters a problem with the gotApplEvent() method when trying to publish a business object to the integration broker.	Error	The connector logs the error. The event is stored in the Archive Store with a status of ERROR_POSTING_EVENT.
The Create, Update, or Delete operation for a message fails during service call requests.	Error	The connector logs the error and returns CWConnectorConstants.FAIL to the calling collaboration or flow.
The Exchange Server error MAPI_E_NOT_FOUND is returned during event processing because a folder, item, or field does not exist.	Error	The connector logs the error. The event is stored in the Archive Store with a status of ERROR_OBJECT_NOT_FOUND.
The Exchange Server error E_ACCESSDENIED is returned when the connector attempts to view or update an object on which it does not have permissions.	Error	The connector logs the error. The event is stored in the Archive Store with a status of ERROR_PROCESSING_EVENT.
The Exchange Server error E_FAIL is returned when an item cannot be updated because the mailbox quota has been exceeded.	Error	The connector logs the error. The event is stored in the Archive Store with a status of ERROR_PROCESSING_EVENT.

Event polling errors

When the connector attempts to poll the Event Store, one or more of the following Outlook messages might be displayed:

```
A program is trying to automatically send e-mail on your behalf. Do you
want to allow this? If this is unexpected, it may be a virus and you
should choose "No."
```

```
A program is trying to access e-mail addresses you have stored in Outlook. Do you
want to allow this? If this is unexpected, it may be a virus and you
should choose "No."
```

This warning is caused by the security features in Outlook. Microsoft provides a method of customizing security settings with a custom form so that this does not occur. For more information refer to the articles at http://support.microsoft.com. In particular:

- If you are running Outlook 2000, refer to the Microsoft Knowledge Base articles 262631 (*OL2000: Information about the Outlook E-mail Security Update*) and 263297 (*OL2000: Administrator Information About the Outlook E-mail Security Update*).
- If you are running Outlook 2002, refer to the Microsoft Knowledge Base articles 290498 (*OL2002: Add-in or Custom Solution Causes a Warning to Appear*) and 290499 (*OL2002: Administrator Information About E-mail Security Features*).

Error handling in Event Listener

The Event Listener logs any abnormal condition that it encounters during processing, regardless of the trace level. It writes the error text to the Windows Event Log as an Error event.

Table 14 describes the most common errors that occur in Event Listener.

Table 14. Event Listener errors

Error description	Error type	Error handling
The Event Listener is unable to open the configuration file (BIA_Exchange.ini).	Warning	The Event Listener uses default values instead of those specified in the configuration file.
Email cannot be sent to the CwEvent user because the user does not exist or because the mailbox is full.	Error	The error is logged with all known information.

Event registration errors

When using Exchange Explorer to create event registrations, you can receive the following error:

```
An error has occurred creating the registration item.
Event sinks must be registered as a COM+ Application prior to
creating the registration event item, or there may be issues
connecting to the network or server. Also check the data for
accuracy. Choose Yes to retry your request, No to abort, or
Cancel to check your data and try again.
```

Verify that the event registration data is accurate and that the network and server are functioning. If the error still occurs, ensure that you have registered your event sink as a COM+ application, as described in “Adding Event Listener to the COM+ application” on page 16.

Tracing messages

Tracing is an optional debugging feature you can turn on to closely follow the connector or Event Listener’s behavior. Tracing messages are configurable and can be changed dynamically. You set various levels depending on the desired detail. The following sections describe tracing for each component of the Exchange adapter.

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

Using tracing with the connector

Trace messages, by default, are written to STDOUT (screen). You can also configure tracing to write to a file.

Table 15 on page 38 describes the types of tracing messages that the connector outputs at each trace level. All the trace messages appear in the file specified by the connector property TraceFileName. These messages are in addition to any tracing messages output by the IBM WebSphere Business Integration Adapter architecture.

Table 15. Tracing messages for the connector

Tracing level	Tracing messages
Level 0	Message that identifies the connector version. No other tracing is done at this level. This is the default tracing level.
Level 1	Tracing messages include: • Status messages • Messages that provide identifying (key) information for each business object processed
Level 2	Business object handler messages that contain information about the arrays and child business objects the connector encounters or retrieves while processing a business object.
Level 3	Messages that provide information about business object processing. For example, these types of messages are delivered when the connector finds a match between business objects, or finds a specific business object in an array of business objects.
Level 4	Tracing messages include: • Application-specific text messages (for example, messages showing the values returned by the functions that parse the business object's application-specific text fields) • Messages that identify when the connector enters or exits a function, which can help trace the connector's process flow
Level 5	Tracing messages include: • Messages that indicate connector initialization (for example, messages that show the value of each configuration property retrieved) • Messages that comprise a representation of a business object before the connector processes it (in other words, displaying the state of the business object as the connector receives it from the collaboration) and after the connector completes its processing (in other words, displaying the state of the business object as the connector returns it to the collaboration) • Messages that comprise a business object dump during processing

Using tracing with the Event Listener

Trace messages, by default, are written to the Windows Event Log as Information events. Table 16 describes the types of tracing messages that the Event Listener outputs at each trace level.

Table 16. Tracing messages for Event Listener

Tracing level	Tracing messages
Level 0	The Event Listener identification message (for example, `2002/07/10 15:01:46.812:  This is version 1.0 of the Exchange adapter`). This message is always displayed the first time the event sink fires.
Level 1	Tracing messages include • Status messages • Messages that provide identifying (key) information for each processed event • Messages for each execution of the onSave(), onDelete(), onSyncSave(), onSyncDelete(), onTimer(), onMDBStartUp(), and onMDBShutDown() methods

Table 16. Tracing messages for Event Listener (continued)

Tracing level	Tracing messages
Level 2	Messages that indicate each time an email message is sent to the CwEvent user.
Level 3	Messages that display event-specific flags (for example, EVT_NEW_ITEM, EVT_COPY, or EVT_MOVE).
Level 4	Messages that display the Event Store field values. Fields include EventID, Event type, FolderID, MessageID, and Priority.
Level 5	Tracing messages include: • Initialization messages (for example, messages that show the value of each configuration property that is retrieved) • Messages that trace individual method entry and exit points

Appendix A. Standard configuration properties for connectors

This appendix describes the standard configuration properties for the connector component of WebSphere Business Integration adapters. The information covers connectors running on the following integration brokers:

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

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

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

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

New and deleted properties

These standard properties have been added in this release.

New properties

- XMLNameSpaceFormat

Deleted properties

- RestartCount
- RHF2MessageDomain

Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties
- Connector-specific configuration properties

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

Using Connector Configurator

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

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

Setting and updating property values

The default length of a property field is 255 characters.

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

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

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

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

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

Summary of standard properties

Table 17 on page 43 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker, as standard property dependencies are based on `RepositoryDirectory`.

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

Table 17. 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 17. 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 17. Summary of standard configuration properties (continued)

Property name	**Possible values**	**Default value**	**Update method**	**Notes**
OADMaxNumRetry	A positive number	`1000`	Dynamic	Repository Directory must be <REMOTE>
OADRetryTimeInterval	A positive number in minutes	`10`	Dynamic	Repository Directory must be <REMOTE>
PollEndTime	`HH:MM`	`HH:MM`	Component restart	
PollFrequency	A positive integer in milliseconds `no` (to disable polling) `key` (to poll only when the letter `p` is entered in the connector's Command Prompt window)	`10000`	Dynamic	
PollQuantity	`1-500`	`1`	Agent restart	JMS transport only: Container Managed Events is specified
PollStartTime	`HH:MM`(HH is 0-23, MM is 0-59)	`HH:MM`	Component restart	
RepositoryDirectory	Location of metadata repository		Agent restart	For ICS: set to <REMOTE> For WebSphere MQ message brokers and WAS: set to C:\crossworlds\repository
RequestQueue	Valid JMS queue name	*CONNECTORNAME*`/REQUESTQUEUE`	Component restart	Delivery Transport is JMS
ResponseQueue	Valid JMS queue name	*CONNECTORNAME*`/RESPONSEQUEUE`	Component restart	Delivery Transport is JMS: required only if Repository directory is <REMOTE>
RestartRetryCount	`0-99`	`3`	Dynamic	
RestartRetryInterval	A sensible positive value in minutes: 1 - 2147483547	`1`	Dynamic	
SourceQueue	Valid WebSphere MQ name	*CONNECTORNAME*`/SOURCEQUEUE`	Agent restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		*CONNECTORNAME*`/SYNCHRONOUSREQUESTQUEUE`	Component restart	Delivery Transport is JMS

Table 17. Summary of standard configuration properties (continued)

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

Standard configuration properties

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

AdminInQueue

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

The default value is `CONNECTORNAME/ADMININQUEUE`.

AdminOutQueue

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

The default value is `CONNECTORNAME/ADMINOUTQUEUE`.

AgentConnections

Applicable only if `RepositoryDirectory` is <REMOTE>.

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

Identifies the integration broker type that you are using. The options are ICS, WebSphere message brokers (WMQI, WMQIB or WBIMB) or WAS.

CharacterEncoding

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

Note: Java-based connectors do not use this property. A C++ connector currently uses the value `ascii7` for this property.

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

ConcurrentEventTriggeredFlows

Applicable only if `RepositoryDirectory` is <REMOTE>.

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

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

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

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

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

ContainerManagedEvents

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

The default value is No value.

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

- PollQuantity = 1 to 500
- SourceQueue = CONNECTORNAME/SOURCEQUEUE

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

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

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

ControllerStoreAndForwardMode

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

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

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

The default is true.

ControllerTraceLevel

Applicable only if RepositoryDirectory is <REMOTE>.

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

DeliveryQueue

Applicable only if DeliveryTransport is JMS.

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

The default value is CONNECTORNAME/DELIVERYQUEUE.

DeliveryTransport

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

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

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

WebSphere MQ and IDL

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

- Asynchronous communication:
  WebSphere MQ allows the application-specific component to poll and persistently store events even when the server is not available.
- Server side performance:
  WebSphere MQ provides faster performance on the server side. In optimized mode, WebSphere MQ stores only the pointer to an event in the repository database, while the actual event remains in the WebSphere MQ queue. This saves having to write potentially large events to the repository database.
- Agent side performance:
  WebSphere MQ provides faster performance on the application-specific component side. Using WebSphere MQ, the connector's polling thread picks up an event, places it in the connector's queue, then picks up the next event. This is faster than IDL, which requires the connector's polling thread to pick up an event, go over the network into the server process, store the event persistently in the repository database, then pick up the next event.

JMS

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

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

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

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

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

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

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

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

This line restricts heap memory usage to a maximum of 768 MB (3 segments * 256 MB). If the process memory grows more than this limit, page swapping can occur, which can adversely affect the performance of your system.

- The `IPCCBaseAddress` property to a value of 11 or 12. For more information on this property, see the *System Installation Guide for UNIX*.

DuplicateEventElimination

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

This property can also be set to `false`.

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

FaultQueue

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

The default value is `CONNECTORNAME/FAULTQUEUE`.

JvmMaxHeapSize

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

The default value is 128m.

JvmMaxNativeStackSize

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

The default value is 128k.

JvmMinHeapSize

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

The default value is 1m.

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

The default is `crossworlds.queue.manager`.

jms.NumConcurrentRequests

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

The default value is 10.

jms.Password

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

There is no default.

jms.UserName

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

There is no default.

ListenerConcurrency

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

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

Locale

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

A locale name has the following format:

```
ll_TT.codeset
```

where:

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

By default, only a subset of supported locales appears in the drop list. To add other supported values to the drop list, you must manually modify the

\Data\Std\stdConnProps.xml file in the product directory. For more information, see the appendix on Connector Configurator.

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

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

LogAtInterchangeEnd

Applicable only if RespositoryDirectory is <REMOTE>.

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

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

MaxEventCapacity

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

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

MessageFileName

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

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

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

MonitorQueue

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

The default value is CONNECTORNAME/MONITORQUEUE

OADAutoRestartAgent

Valid only when the RepositoryDirectory is <REMOTE>.

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

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

The default value is `false`.

OADMaxNumRetry

Valid only when the `RepositoryDirectory` is <REMOTE>.

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

The default value is `1000`.

OADRetryTimeInterval

Valid only when the `RepositoryDirectory` is <REMOTE>.

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

The default is `10`.

PollEndTime

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

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

PollFrequency

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

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

The default is `10000`.

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

PollQuantity

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

PollStartTime

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

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

RequestQueue

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

The default value is `CONNECTOR/REQUESTQUEUE`.

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

The default is `3`.

RestartRetryInterval

Specifies the interval in minutes at which the connector attempts to restart itself. When used for a parallel connector, specifies the interval at which the master connector application-specific component attempts to restart the slave connector application-specific component. Possible values ranges from 1 to 2147483647.

The default is 1.

SourceQueue

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

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see "ContainerManagedEvents" on page 47.

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 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" on page 57
- "Starting Connector Configurator" on page 58
- "Creating a connector-specific property template" on page 59
- "Creating a new configuration file" on page 61
- "Setting the configuration file properties" on page 64
- "Using Connector Configurator in a globalized environment" on page 70

Overview of Connector Configurator

Connector Configurator allows you to configure the connector component of your adapter for use with these integration brokers:
- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator, WebSphere MQ Integrator Broker, and WebSphere Business Integration Message Broker, collectively referred to as the WebSphere Message Brokers (WMQI)
- WebSphere Application Server (WAS)

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

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see "Running Configurator in stand-alone mode" on page 58).

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 59 to set up a new one.

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

Starting Connector Configurator

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

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

Running Configurator in stand-alone mode

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

To do so:

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

You may choose to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see "Completing a configuration file" on page 63.)

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 59.
- To use an existing file, simply modify an existing template and save it under the new name.

Creating a new template

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

To create a template:

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

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

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

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

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

If you do not see any template that displays the connector-specific properties used by your connector, you will need to create one.

Specifying general characteristics

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

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

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

Specifying values

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

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

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

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

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

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

Setting dependencies

When you have made your changes to the **General** and **Value** tabs, click **Next**. The **Dependences - Connector-Specific Property Template** dialog box appears.

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

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

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

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

Creating a new configuration file

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

To select a broker:

- In the Connector Configurator home menu, click **File>New>Connector Configuration**. The **New Connector** dialog box appears.
- In the I**ntegration Broker** field, select ICS, WebSphere Message Brokers or WAS connectivity.
- Complete the remaining fields in the **New Connector** window, as described later in this chapter.

You can also do this:

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

Creating a configuration file from a connector-specific template

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

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

     Enter the name of the connector. Names are case-sensitive. The name you enter must be unique, and must be consistent with the file name for a connector that is installed on the system.

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

     Click ICS or WebSphere Message Brokers or WAS.
   - **Select Connector-Specific Property Template**

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

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

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

Using an existing file

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

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

- A previous configuration file for the connector.
  Such a file typically has the extension *.cfg.

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

To use an existing file to configure a connector, you must open the file in Connector Configurator, revise the configuration, and then resave the file.

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

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

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

Follow these steps to open a connector configuration from a System Manager project:
1. Start System Manager. A configuration can be opened from or saved to System Manager only if System Manager has been started.
2. Start Connector Configurator.
3. Click **File>Open>From Project**.

Completing a configuration file

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

The title of the configuration screen displays the integration broker and connector name as specified in the file. Make sure you have the correct broker. If not, change the broker value before you configure the connector. To do so:
1. Under the **Standard Properties** tab, select the value field for the `BrokerType` property. In the drop-down menu, select the value `ICS`, `WMQI`, or `WAS`.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 66..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

Setting the configuration file properties

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

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

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

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

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

- Associated Maps
- Resources
- Messaging (where applicable)

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

Standard properties differ from connector-specific properties as follows:

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

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

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

Setting standard connector properties

To change the value of a standard property:

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

Setting application-specific configuration properties

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

1. Right-click in the top left portion of the grid. A pop-up menu bar will appear. Click **Add** to add a property. To add a child property, right-click on the parent row number and click **Add child**.
2. Enter a value for the property or child property.
3. To encrypt a property, select the **Encrypt** box.
4. Choose to save or discard changes, as described for "Setting standard connector properties."

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

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

Encryption for connector properties

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

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

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

Update method

Refer to the descriptions of update methods found in the *Standard configuration properties for connectors* appendix, under "Setting and updating property values" on page 42.

Specifying supported business object definitions

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

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

If ICS is your broker

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

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

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

To delete a business object from the supported list:

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

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

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

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

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

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

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

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

If a WebSphere Message Broker is your broker

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

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

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

If WAS is your broker

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

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

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

Associated maps (ICS only)

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

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

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

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

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

- **Business Object Name**

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

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

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

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

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

  To explicitly bind a map:

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

Resources (ICS)

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

Messaging (ICS)

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

Setting trace/log file values

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

To change the logging and tracing values:

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

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

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

Data handlers

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

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

Saving your configuration file

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

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

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

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

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

Changing a configuration file

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

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

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

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

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

IBM WebSphere Business Integration Adapter FrameWork V2.4.0

Index

Printed in USA