IBM WebSphere Business Integration Adapters

IBM

Adapter for e-Mail User Guide

Adapter Version 54.x

IBM WebSphere Business Integration Adapters

Adapter for e-Mail User Guide

Adapter Version 54.x

┌─ **Note!** ───┐
│ Before using this information and the product it supports, read the information in "Notices" on page 91. │
└──┘

30September2004

This edition of this document applies to IBM WebSphere Business Integration Adapter for e-Mail (5724-H04), version 5.4.x.

To send us your comments about this document, email 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

About this document

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

What this document includes

This document describes installation, connector property configuration, business object development, and troubleshooting for the IBM WebSphere Business Integration Adapter for e-Mail.

What this document does not include

This document does not describe deployment metrics and capacity planning issues such as server load balancing, number of adapter processing threads, maximum and minimum throughputs, and tolerance thresholds.

Such issues are unique to every customer deployment and must be measured within or close to the exact environment where the adapter is to be deployed. You should contact your IBM services representative to discuss the configuration of your deployment site, and for details on planning and evaluating these kinds of metrics, given your specific configuration.

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

What this document includes

This document describes installation, connector property configuration, business object development, and troubleshooting for the IBM WebSphere Business Integration Adapter for e-Mail.

What this document does not include

This document does not describe deployment metrics and capacity planning issues such as server load balancing, number of adapter processing threads, maximum and minimum throughputs, and tolerance thresholds.

Such issues are unique to every customer deployment and must be measured within or close to the exact environment where the adapter is to be deployed. You should contact your IBM services representative to discuss the configuration of your deployment site, and for details on planning and evaluating these kinds of metrics, given your specific configuration.

Audience

This document is for IBM consultants and customers. Users of this document should be familiar with the WebSphere business integration systems and e-mail technologies.

IBM(R) WebSphere(R) Business Integration Adapters supply integration connectivity for leading e-business technologies and enterprise applications.

This document describes the installation, configuration, and business object development for the adapter for e-Mail.

Related documents

The complete set of documentation available with this product describes the features and components common to all WebSphere Business Integration Adapters installations, and includes reference material on specific components.

You can install related documentation from the following sites:

- For general adapter information; for using adapters with WebSphere message brokers (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, WebSphere Business Integration Message Broker); and for using adapters with WebSphere Application Server, see the IBM WebSphere Business Integration Adapters InfoCenter:
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- For using adapters with WebSphere InterChange Server, see the IBM WebSphere InterChange Server InfoCenters:
  http://www.ibm.com/websphere/integration/wicserver/infocenter
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- For more information about WebSphere message brokers:
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- For more information about WebSphere Application Server:
  http://www.ibm.com/software/webservers/appserv/library.html

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

Note: Important information about this product may be available in Technical Support Technotes and Flashes issued after this document was published. These can be found on the WebSphere Business Integration Support Web site, http://www.ibm.com/software/integration/websphere/support/. Select the component area of interest and browse the Technotes and Flashes sections.

Typographic conventions

This document uses the following conventions:

courier font	Indicates a literal value, such as a command name, filename, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
italic, italic	Indicates a cross-reference or variable name.
blue text	Blue text, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click any blue text to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
[ ]	In a syntax line, square brackets surround an optional parameter.

...	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.
< >	In a naming convention, angle brackets surround individual elements of a name to distinguish them from each other, as in `<server_name><connector_name>tmp.log`.
/, \	In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes. All WebSphere business integration system product pathnames are relative to the directory where the product is installed on your system.
%text% and *$text*	Text within percent (%) signs indicates the value of the Windows `text` system variable or user variable. The equivalent notation in a UNIX environment is `$text`, indicating the value of the `text` UNIX environment variable.
ProductDir	Product family is WBIA: Represents the directory where the IBM WebSphere Business Integration Adapters product is installed. The CROSSWORLDS environment variable contains the ProductDir directory path, which is IBM\WebSphereAdapters by default.

New in this release

Version 5.4.x

The Adapter for e-Mail has been updated with the following items:
- The adapter, running on Windows, provides bidirectional script support
- Tivoli License Manager enablement
- Support for GB18030 Chinese code has been added for this release
- ReplyTo feature has been added to the e-Mail adapter
- Support for RFC822 specification for syntax of Internet addresses has been added
- For this release, the adapter runtime code has been split from the server code line

The 5.4.x version of the adapter is supported on the following platforms:
- WIN 2003
- AIX 5.1 and 5.2: this adapter supports 32-bit JVM on a 64-bit platform
- Solaris 9: this adapter supports 32-bit JVM on a 64-bit platform
- HP-UX, 11, 11i
- Linux RedHat AS 3.0, ES 3.0 and WS 3.0
- SUSE LINUX Standard Server 8.1 and Enterprise Server 8.1 SP3
- IBM JRE/JDK 1.4.2

Version 5.3.x

The Adapter for e-Mail has been updated with general maintenance fixes.

Beginning with the 5.3 version, the Adapter for e-Mail is no longer supported on the Solaris 7 platform.

Version 5.2.x

The Adapter for e-Mail has been updated with general maintenance fixes.

Beginning with the 5.2 version, the adapter for e-Mail is no longer supported on Microsoft Windows NT.

Adapter installation information has been moved from this guide. See Chapter 2, "Installing the e-Mail adapter and related files" on page 17, for the new location of that information.

Prior versions

New features and other changes in prior versions.

Version 5.1.x

The adapter can now use WebSphere Application Server as an integration broker. For further information, see "Broker compatibility" on page 13.

The adapter now runs on the following platforms:
- Solaris 7, 8
- AIX 5.x
- HP UX 11.i

Version 5.0.x

Updated in March, 2003. The "CrossWorlds" name is no longer used to describe an entire system or to modify the names of components or tools, which are otherwise mostly the same as before. For example "CrossWorlds System Manager" is now "System Manager," and "CrossWorlds InterChange Server" is now "WebSphere InterChange Server."

The IBM WebSphere Business Integration Adapter for e-Mail is being released with the same functionality as in previous releases.

The release of this document contains the following new or corrected information:
- Documentation of the DefaultVerb adapter property, which enables the adapter to provide a default verb for business objects it generates from a file if no verb information is available in the file. For more information, see "DefaultVerb" on page 21.
- Documentation of the AttachmentExtension meta-object attribute, which enables the adapter to specify an extension for the files it generates representing business objects contained in an e-mail. For more information, see "AttachmentExtension" on page 30.
- The child meta-object attribute previously known as `EventRecovery` has been renamed to InDoubtEvents. The possible values for this attribute have changed to `Reprocess`, `FailOnStartUp`, `LogError`, and `Ignore`. For more information, see "InDoubtEvents" on page 35.
- There is a new child meta-object attribute known as `Reconnect` that has been added to the `MO_PollLocation`. The possible values for this attribute are true or false. These values will be used to determine whether an attempt will be made to reconnect to the mail server upon loss of connection. For more information, see "Event notification" on page 4.

Version 4.3.x

IBM WebSphere Business Integration for e-Mail includes the adapter for Email. This adapter operates with both the InterChange Server and WebSphere MQ Integrator integration brokers. An integration broker, which is an application that performs integration of heterogeneous sets of applications, provides services that include data routing.

This adapter includes:
- An application-specific component specific to e-Mail technology.
- A sample, which is located in \connectors\EMail\samples.
- IBM WebSphere Adapter Framework, which consists of:
  - Adapter Framework
  - Development tools (including Business Object Designer and Connector Configurator)
  - APIs (including ODK, JCDK, and CDK)

This manual provides instructions about using this adapter with both integration brokers: InterChange Server and WebSphere MQ Integrator.

The adapter has been internationalized. For more information, see "Processing locale-dependent data" on page 10 and Appendix A, "Standard configuration properties for adapters," on page 41.

Chapter 1. Overview of the adapter

This chapter describes the adapter for e-Mail and its components. The adapter for e-Mail enables an integration broker to exchanges business objects, files, or messages with e-mail applications. The adapter also enables the IBM WebSphere InterChange Server integration broker to send e-mail to specified users when Error or Fatal Error messages occur. It contains the following sections:

- "Features of the adapter for e-Mail"
- "Adapter components"
- "How the adapter works" on page 3
- "Example scenario for the adapter for e-Mail" on page 8
- "Processing locale-dependent data" on page 10
- "Common Event Infrastructure" on page 11
- "Application Response Measurement" on page 11

Features of the adapter for e-Mail

The adapter for e-Mail has the following functionality:

- **Business object processing** – The adapter for e-Mail enables an integration broker to send e-mail messages containing business objects or files to e-mail applications.

  The adapter can also poll for new e-mail messages on one or more mail servers, and convert the messages into business objects that it sends to the integration broker. Encapsulating business objects in e-mail messages enables enterprises to integrate business processes and synchronize data across the Internet.

- **Error notification (InterChange Server integration broker only)** – The adapter for e-Mail enables the IBM WebSphere InterChange Server to use e-mail to deliver error messages. InterChange Server sends an e-mail message to specified users when Error or Fatal Error messages occur.

  The adapter for e-Mail works with the e-Mail collaboration, which is automatically created by InterChange Server. When an error occurs and e-mail notification is configured, InterChange Server creates an e-Mail Notification business object containing the message information and passes it to the adapter for e-Mail. The adapter extracts the message information from the business object, creates an e-mail message, and sends the message.

  Note: Business object processing and error notification are two distinct roles for the adapter. In order to enable both behaviors, you must configure the adapter for both. For more information, see Chapter 2, "Installing and configuring the adapter," on page 13.

Adapter components

Adapters consist of two parts: the adapter framework and the application-specific component. The adapter framework, whose code is common to all adapters, acts as an intermediary between the integration broker and the application-specific component. The application-specific component contains code tailored to a particular application or technology. 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 adapter framework and the application-specific component, which it refers to as the adapter. The WebSphere Business Integration Adapter for e-Mail also uses a data handler, which converts a business object to a specific data format, and converts e-mail attachments in a specific format into business objects. The adapter can use one of the IBM-delivered data handlers or a custom data handler. For information on data handlers, see Chapter 3, "Planning an e-Mail adapter implementation," on page 27 and the *Data Handler Guide*.

Figure 1 illustrates the architecture of the adapter for e-Mail.
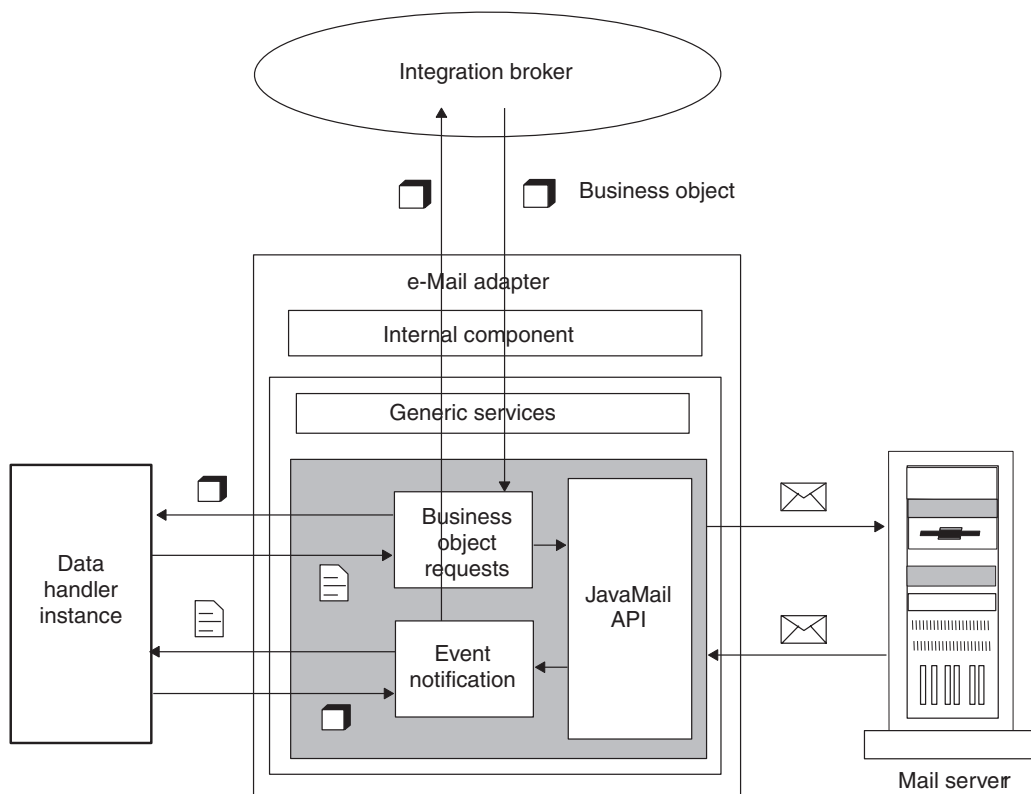
Figure 1. Adapter for e-Mail architecture

Transport protocols

The adapter interacts with mail servers using JavaMail. The adapter supplies the parameters that JavaMail requires to communicate with mail servers, and JavaMail handles the underlying SMTP (Simple Mail Transport Protocol) and IMAP (Internet Message Access Protocol) protocols.

SMTP is the Internet mail transport protocol. The adapter uses the SMTP protocol for the mail transport.

For polling, adapter for e-Mail supports the IMAP protocol via JavaMail for online access of e-mail messages and remote manipulation of the mailbox. IMAP is used for storing and manipulating messages on a mail server, thus enabling access of e-mail from anywhere. IMAP supports online query of messages, selective retrieval

of message contents, and server-side searches for messages. Note that the adapter does not currently support the POP protocol via JavaMail.

How the adapter works

The following sections describe how the adapter processes business object requests and handles events.

Business object processing

When an integration broker sends a business object to the adapter for e-Mail, the adapter processes the business object and generates an e-mail.

An e-Mail business object must contain e-mail routing information (from address, to address, subject, and content), the MIME type of the attachments, and the business objects to deliver. The business object can also specify the complete path of files to include in the e-mail as attachments. A single business object can result in an e-mail message containing multiple business objects and multiple files as attachments.

To process a business object request, the adapter:
1. Extracts the routing information from the top-level business object and composes an e-mail message.
2. Processes each contained business object or file as follows:
   - If there are contained business objects, the adapter calls a data handler to convert each business object into a business object string that is included as an attachment to the e-mail. The MIME type of the attachment is defined in the top-level business object. There is only one business object per attachment.
   - If there are files specified in the top-level business object, the adapter includes the files as attachments to the e-mail. Each file must be located at its specified location.
3. When the e-mail is complete, the adapter delivers it using JavaMail.

Figure 2 illustrates business object processing with the adapter for e-Mail. For information on creating business objects for the adapter, see Chapter 4, "Developing business objects for the adapter," on page 29.
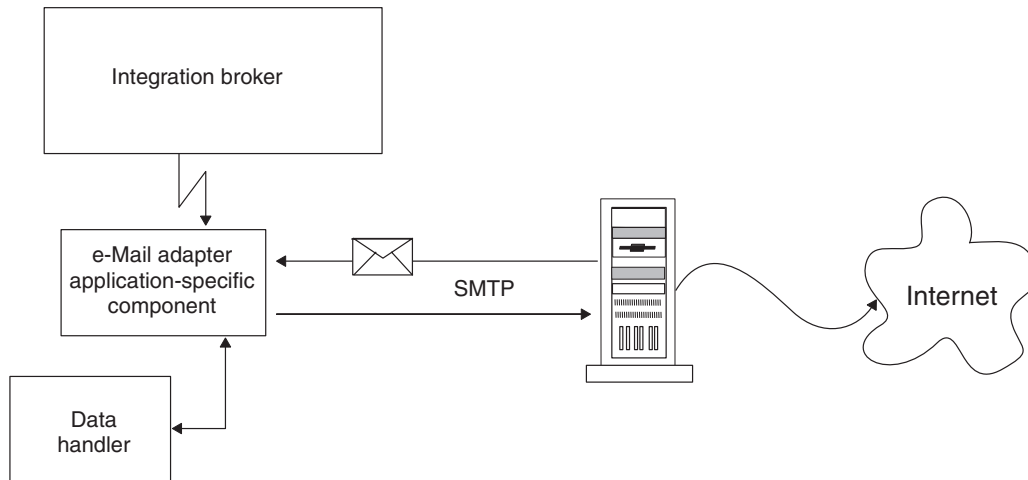
Figure 2. Adapter for e-Mail business object processing

Event notification

In event notification, the adapter polls for e-mail messages in specified folders on the mail server. The adapter checks the folders, retrieves new messages, converts the messages to business objects using the data handler, and sends the business objects to the integration broker.

The adapter can poll in multiple locations. The following folders are required per poll location:

- Poll folder – The folder in which the adapter polls.
- Archive folder – The folder in which the adapter archives successful events.
- Failed event folder – The folder in which the adapter archives failed events.

Although polling locations can be located on different IMAP stores, these three folders, if specified for a poll location, should be in the same mailbox.

The sections that follow describe the event mechanism is more detail.

Event triggering

A triggering event for the adapter is a new e-mail that arrives in a poll folder. The adapter checks each poll folder for new e-mails at the interval defined by the PollFrequency configuration property.

Event detection

Each IMAP mail server has a message store called an IMAP store. The IMAP store has a mailbox for each of the users of the mail server. A mailbox can have multiple folders, and a folder can contain multiple e-mail messages or folders.

Each poll folder must have a corresponding archive folder and failed event folder. The poll folder, archive folder, and failed event folder are together referred to as a **poll location**. An administrator must create the folders and specify the poll locations in the meta-object for the adapter.

The adapter polls for new messages in the poll folder of an IMAP store. When the adapter sees a new message in a poll folder, it retrieves the message. The adapter determines that a message is a new message by checking the status of the IMAP flag for the message.

Event status: IMAP supports flags that describe the state of e-mail messages in a mailbox. The event detection mechanism uses the flags to determine which e-mail messages to retrieve. The status of an event is specified both by the folder in which the message is located and by the message flag.

The event mechanism uses the IMAP flag SEEN to mark events that are "in progress". In polling, the adapter checks the messages in a poll folder and picks messages that are not marked with the SEEN flag. These events are "ready to be polled". When the adapter picks a message from the poll folder for processing, it sets the SEEN flag for the message. Events marked as SEEN are "in progress" events.

Note: The adapter does not poll messages marked as SEEN.

When processing is complete, and all the attachments of a message are successfully converted to business objects and sent to the integration broker, the e-mail is marked as FLAGGED and then moved to the archive folder. Messages that have attachments that cannot be converted to business objects, or messages that have unsubscribed business objects or business objects that could not be delivered to integration broker are moved to the failed event folder. Table 1 summarizes the event states for the adapter.

Table 1. Event states

Event state	e-Mail message state
ready to be polled	All messages in the poll folder with the SEEN flag not set.
in progress	All messages in the poll folder with the SEEN flag set.
sent to integration broker	All messages in the archive folder. Any message in the PollFolder that is marked with the FLAGGED flag set. **Note:** The adapter changes the flag of an IN_PROGRESS message to FLAGGED once it has been sent to the interchange broker and is set to be archived (moved to the archive folder).
unsubscribed business objects and errors	All messages in the failed event folder.

Event retrieval

To retrieve events, the adapter polls each poll folder in turn and picks one e-mail from each mailbox. This polling strategy ensures that e-mails in all poll folders get processed.

The adapter processes each of the e-mails retrieved from a poll location in the following way:

1. From the poll folder, the adapter picks an e-mail with the SEEN flag not set. The adapter then sets SEEN flag on this e-mail.
2. The adapter fetches the e-mail from the poll folder. The e-mail can have any number of attachments, and the adapter processes each attachment. There must be only one business object per attachment.

3. For each attachment, the adapter uses the MIME type of the attachment to determine which data handler to use to convert the attachment to a business object. The adapter retrieves the contents of the attachment and passes it to the data handler. Attachments in an e-mail can be in different formats. If the attachment has no MIME type, however, the adapter does not process the attachment.

4. The data handler generates a business object and returns it to the adapter.

5. If the business object is subscribed, the adapter delivers it to the integration broker; otherwise, the adapter considers this e-mail to be a failed event.

The adapter processes all message attachments in the same way. An e-mail message is considered to be a successful event if all the attachments were converted to business objects and delivered to the integration broker. In the cases of successful e-mails, the adapter marks the e-mail messages as FLAGGED, then, archives the successfully processed e-mails to the archive folder for this polling location. The adapter does not archive an e-mail until all of the attachments of the e-mail have been processed.

If one or more attachments cannot be converted to business objects or the business object provided by the data handler is unsubscribed, the event is a failed event. In this case, the adapter archives the e-mail in the failed event folder for this polling location.

The adapter delivers a number of events per folder that is less than or equal to the value configured for the PollQuantity adapter property. The adapter does not fetch and process new e-mail until it has finished processing the previous e-mail.

Failed events: An e-mail is considered to be a failed event in the following situations:

- An attachment could not be retrieved.
- The adapter was not able to determine which data handler to use for an attachment.
- The data handler was not able to convert an attachment to a business object.
- The business object returned by a data handler was not subscribed to.

The adapter processes all the attachments of a message and deliver business objects obtained from a message to the integration broker, even if the message is a failed event. As an example, assume that an e-mail has five attachments. The adapter processes attachments one and two, fails to process attachment three, and then succeeds in processing attachments four and five. Although the adapter delivers business objects for four of the attachments, the entire e-mail message is treated as failed event and archived in the failed event folder because attachment three could not be processed. The adapter logs the details of each attachment that it fails to convert to a business object.

Event archiving

To archive events, the adapter requires an archive folder and a failed event folder. Events are archived into these folders as follows:

- If an e-mail is a successful event, the adapter marks the e-mail messages as FLAGGED, then, moves the e-mail from the poll folder to the to the archive folder. If an archive folder is not specified, the adapter deletes the e-mail from the poll folder, and the message is lost.

- If the e-mail is a failed event, the adapter moves the e-mail from the poll folder to the failed event folder. If a failed event folder is not specified, the adapter deletes the event from the poll folder, and the message is lost.

Recovery mechanism

An administrator can configure the recovery mechanism of the adapter for each polling location specified by the PollConfigMO configuration property. The PollConfigMO property identifies a meta object that has an `InDoubtEvents` attribute for each polling location. This attribute can have the values `Reprocess`, `Ignore`, `LogError`, or `FailOnStartUp`.

If the adapter crashes between poll calls, the poll folders may have messages that are in-progress. These messages have a SEEN flag set and are in-doubt events for the adapter. The next time the adapter starts, the initialization procedure scans all the poll folders for the in-doubt transactions as follows:

- If `InDoubtEvents` for the poll location is set to `Reprocess`, the adapter resets the SEEN flag of the in-doubt events in that poll folder. The adapter picks these messages in the subsequent poll calls.
- If the InDoubtEvents setting for the poll location is FailOnStartUp, the adapter terminates after displaying error messages.
- If the InDoubtEvents setting for the poll location is LogError, the adapter continues after displaying error messages.
- If InDoubtEvents is Ignore, the adapter continues without processing the in-doubt transactions.

Before restarting the adapter, the administrator can view each of the poll folders using an e-mail client program, and determine how to handle in-doubt events. The administrator can either leave the in-doubt messages in the poll folder, reset the SEEN flag of in-doubt events, or delete the messages. When the adapter next starts, if there are in-doubt events, the adapter processes the in-doubt transactions in the initialization method as explained below.

Adapter initialization

When the adapter for e-Mail starts, the following occurs:

1. The adapter gets the subscription list for the business objects it supports.
2. Using the information in the adapter's meta-object, the adapter creates a list of poll locations.
3. The adapter tries to connect to each poll location. The adapter does not start if it fails to connect to any of the poll locations.
4. The adapter makes sure that all the folders for each poll location exist. The adapter will not start if any folder does not exist.
5. The adapter retrieves the value of the `SMTP_MAILHOST` property from the adapter properties. Using this value, the adapter connects to the mail host. If the attempt to connect to the mail host is not successful, the adapter does not start.
6. The adapter looks for in-doubt transactions in each of the poll folders. If the adapter finds in-doubt transactions, it performs recovery based on the value of the `InDoubtEvents` setting for the poll location. If the `InDoubtEvents` flag is `Reprocess`, the adapter resets the SEEN flag for the message. If it is `FailOnStartUp`, the adapter terminates. If it is `LogError`, the adapter logs the error and continues. If the `InDoubtEvents` flag is `Ignore`, the adapter continues and ignores the in-doubt transactions.

Example scenario for the adapter for e-Mail

Figure 3 on page 9 illustrates a hypothetical scenario for the adapter for e-Mail. The illustration shows a cross-enterprise solution that enables trading partners to exchange business data via e-mail. The trading partners have chosen to exchange business data via e-mail rather than communicate across firewalls or send data via FTP (File Transfer Protocol).

Trading partner A has an enterprise application generating business data. A WebSphere business integration adapter polls for events in the application, generates business objects, and sends the business objects to an integration broker. The integration broker is configured to forward the business objects to the adapter.

The adapter uses a data handler to convert business objects to a specified data format. The adapter composes an e-mail message, attaches files containing the converted data, and sends the message across the Internet to mail host MailServer1.

Concurrently, a legacy application generates e-mail messages containing business data formatted as text strings, and sends the messages to mail host MailServer2.

Trading partner B configures the adapter to poll both MailServer1 and MailServer2 for new e-mails. When new e-mails arrive, the adapter checks the MIME type of each attachment, calls the appropriate data handler to convert the attachments to business objects, and sends the business objects to the integration broker.
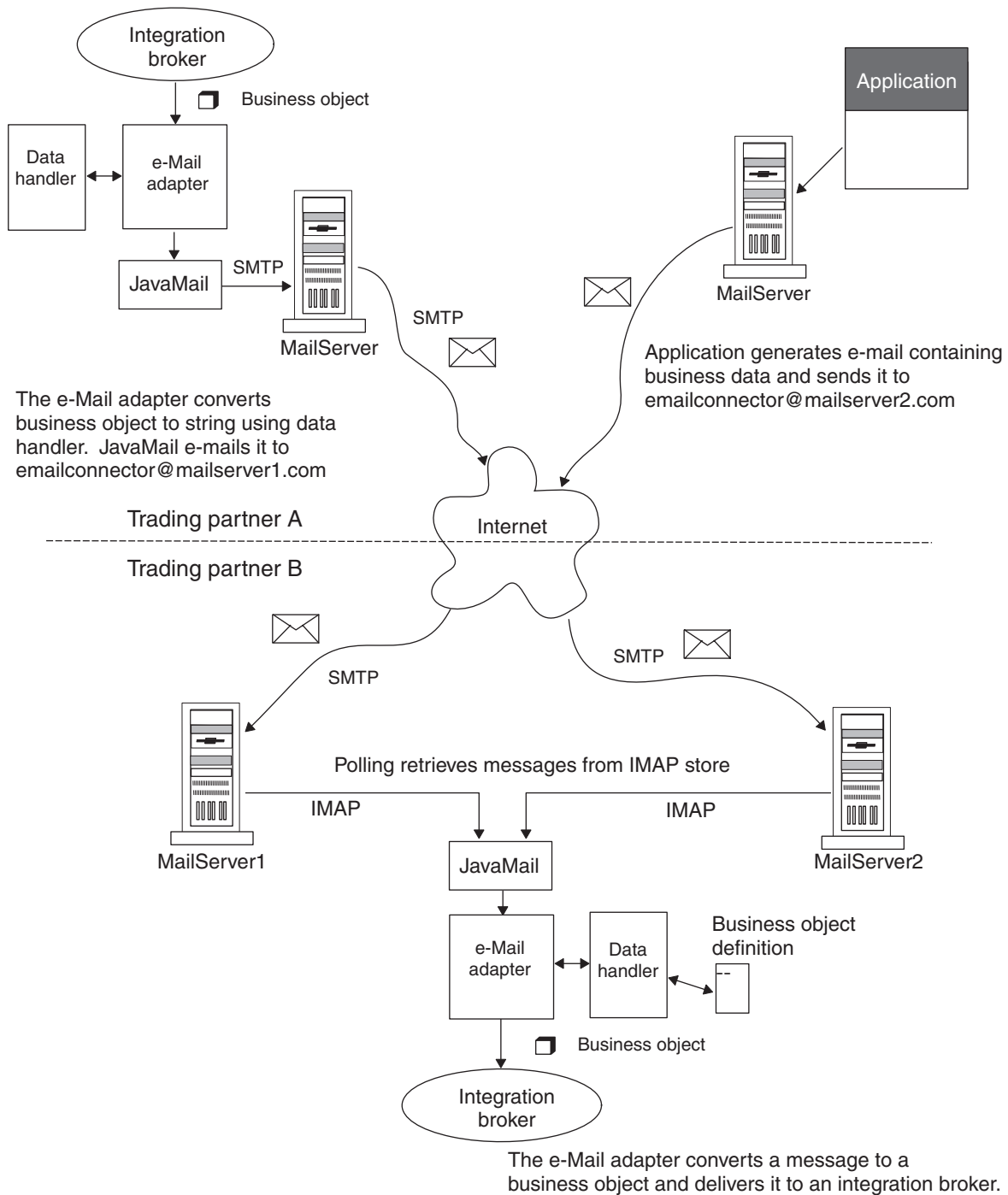
Figure 3. Example scenario for the adapter for e-Mail

As Figure 3 shows, trading partners can create new mailboxes at host mail servers and use the adapter to execute business processes over the Internet. For enterprises with firewalls between the Internet and local networks, or between internal department networks, the use of e-mail enables data exchange without requiring tunnels through firewalls. The adapter for e-Mail is also useful when applications are locked for direct access but can exchange data by means of e-mail.

Processing locale-dependent data

The adapter has been internationalized so that it can support double-byte character sets, and deliver message text in the specified language. When the adapter transfers data from a location that uses one character code set to a location that uses a different code set, it performs character conversion to preserve the meaning of the data.

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

To log error and informational messages in the appropriate language and for the appropriate country or territory, configure the Locale standard configuration property for your environment. For more information on these properties, see Appendix A, "Standard configuration properties for adapters," on page 41

Characteristics of e-Mail adapter locale processing

The adapter encodes the e-mails it exchanges according to character set and encoding information configured at the header, content, and business object levels. The following traits characterize the encoding activity:

- In the header, only the subject line is encoded; fields that contain information about the sender and receiver, for instance, are not encoded.
- Business objects contained in an e-mail can either be processed as parts of the message or as attachment files; for more information on this, see "Business object structures" on page 29.

Locale configuration order of precedence

There is an order of precedence to how the adapter attempts to apply encoding to the data it processes:

- To encode a business object that is contained in an e-mail, the adapter first checks if the MIME type specified for the business object is "text"; if it is not, then the adapter uses "UTF-8" as the MIME character set. Note that this does not apply to the message header or content, but only to a business object.
- The adapter searches for encoding information in the top-level wrapper business object for data it processes, which contains information about the content and routing requirements of an e-mail; if the necessary attributes have been defined and configured then the adapter uses these specifications. For more information about the structure of top-level objects, see "Business object structures" on page 29.

  If encoding information is not specified at the business object level, then the adapter checks if the optional properties are configured in the adapter definition; if they are then the adapter uses those properties to drive its behavior. For more information about adapter-specific properties, see "Adapter-specific properties" on page 20.

- If encoding information is not specified at either the business object level or in the adapter definition, then the adapter uses the default system locale information.

Common Event Infrastructure

This adapter is compatible with IBM's Common Event Infrastructure, a standard for event management that permits interoperability with other IBM WebSphere event-producing applications. If Common Event Infrastructure support is enabled, events produced by the adapter can be received (or used) by another Common Event Infrastructure-compatible application.

For more information refer to the Common Event Infrastructure appendix in Appendix D, "Common Event Infrastructure," on page 85.

Application Response Measurement

This adapter is compatible with the Application Response Measurement application programming interface (API), an API that allows applications to be managed for availability, service level agreements, and capacity planning. An ARM-instrumented application can participate in IBM Tivoli Monitoring for Transaction Performance, allowing collection and review of data concerning transaction metrics.

For more information refer to the Application Response Measurement appendix in Appendix C, "Application Response Measurement," on page 83this guide.

Chapter 2. Installing and configuring the adapter

This chapter describes how to install and configure the e-Mail adapter. It contains the following sections:

- "Adapter for e-Mail environment"
- "Determining installation and configuration tasks" on page 15
- "Installing the e-Mail adapter and related files" on page 17
- "Verifying installation" on page 17
- "Adapter configuration tasks" on page 18
- "Configuring the adapter definition" on page 19
- "Starting the connector" on page 23

Adapter for e-Mail environment

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

- "Broker compatibility"
- "Adapter platforms" on page 14
- "Locale-dependent data" on page 14

Broker compatibility

The adapter framework that an adapter uses must be compatible with the version of the integration broker (or brokers) with which the adapter is communicating. Version 5.4 of the adapter for e-Mail is supported on the following versions of the adapter framework and with the following integration brokers:

Adapter framework: WebSphere Business Integration Adapter Framework version 2.6.

Integration brokers:

- WebSphere InterChange Server, versions 4.2.2, 4.3
- WebSphere MQ Integrator, version 2.1
- WebSphere MQ Integrator Broker, version 2.1
- WebSphere Business Integration Message Broker, version 5.0.1
- WebSphere Application Server Enterprise, version 5.0.2, with WebSphere Studio Application Developer Integration Edition, version 5.1.1
- WebSphere Business Integration Server Foundation version 5.1.1

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.

Adapter platforms

In addition to a broker, this adapter requires one of the following operating
systems:

- All operating system environments require the Java compiler (IBM JDK 1.4.2for
  Windows 2000) for compiling custom adapters
- **AIX:**
  AIX 5.1 with Maintenance Level 4
  **AIX 5.2 with Maintenance Level 1. This adapter supports 32-bit JVM on a
  64-bit platform.**
- **Solaris:**
  Solaris 8 (2.8)with Solaris Patch Cluster dated Feb. 11, 2004 or later
  Solaris 9 (2.9) with Solaris Patch Cluster dated February 11, 2004 or later. This
  adapter supports 32-bit JVM on a 64-bit platform.
- **HP-UX:**
  HP-UX 11.i (11.11) with June 2003 GOLDBASE11i and June 2003 GOLDAPPS11i
  bundles
- **Linux:**
  Red Hat Enterprise Linux AS 3.0 with Update 1
  Red Hat Enterprise Linux ES 3.0 with Update 1
  Red Hat Enterprise Linux WS 3.0 with Update 1
  SUSE Linux Enterprise Server x86 8.1 with SP3
  SUSE Linux Standard Server x86 8.1 with SP3
-

  Note: The TMTP (Tivoli Monitoring for Transaction Performance) component of
  the WebSphere Business Integration Adapter FrameworkV2.6 is not
  supported on Linux Red Hat.

- **Windows:**
  Windows 2000 (Professional, Server, or Advanced Server) with Service Pack 4
  Windows XP with Service Pack 1A, for WebSphere Business Integration Adapter
  Framework (administrative tools only)
  Windows 2003 (Standard Edition or Enterprise Edition)

Locale-dependent data

The adapter has been internationalized so that it can support double-byte character
sets, and deliver message text in the specified language. When the adapter
transfers data from a location that uses one character code set to a location that
uses a different code set, it performs character conversion to preserve the meaning
of the data.

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

This adapter supports the processing of bidirectional script data for the Arabic and Hebrew languages when the adapter is run in a Windows environment. Bidirectional processing is not supported in non-Windows environments. To log error and informational messages in the appropriate language and for the appropriate country or territory, configure the Locale standard configuration property for your environment. For more information on these properties, see Appendix A, "Standard configuration properties for adapters," on page 41

This adapter supports the processing of bidirectional script data for languages such as Arabic, Hebrew, Urdu, Farsi and Yiddish. To use the bidirectional capacity, you must configure the bidirectional standard properties. For more information, refer to the standard configuration properties for connectors in Appendix A, "Standard configuration properties for adapters," on page 41.

Determining installation and configuration tasks

The installation and configuration tasks for the e-Mail adapter depend on the roles it must fill in the WebSphere business integration system. Read the sections below to determine what tasks you must perform based on how you will be using the adapter.

Note: If you are installing the e-Mail adapter as a stand alone system, see the installation guides for further instructions, *System Installation Guide for Windows* or the *Implementation Guide for WebSphere MQ Integrator Broker*.

Using the adapter for event notification

If the adapter must poll for new e-Mail messages in an application and send them to the integration broker for processing, perform the following steps:

1. Install the adapter as described in "Installing the e-Mail adapter and related files" on page 17.
2. Configure the mail server and mailboxes for poll locations. For information, see "Configuring mail servers for polling" on page 18.
3. Create the business objects required to support the interface. For more information, see Chapter 4, "Developing business objects for the adapter," on page 29.
4. Create or modify a meta-object for the data handler that the adapter uses to process the data. For more information, see "Data handler meta-objects" on page 33.
5. Create and configure a poll meta-object to contain a child meta-object for each poll location. For more information, see "Using e-Mail adapter meta-objects" on page 34.
6. Configure the mail servers that the adapter will be polling as described in "Configuring mail servers for polling" on page 18.
7. Configure the adapter definition as described in "Configuring the adapter definition" on page 19. Keep the following points in mind:
   - You must add the business objects developed for the interface in 3 as supported business objects for the adapter.
   - You must add the poll meta-object created in 5 as a supported object for the adapter.
   - You must add the data handler meta-object created in 4 as a supported object for the adapter.

- You must configure the properties of the adapter; because the adapter will be polling, you must be sure to set the value of the PollConfigMO property to the name of the poll meta-object created in 5.

8. Run the adapter as described in "Starting the connector" on page 23.

Using the adapter for business object processing

If the adapter must receive business objects from the integration broker and process them, perform the following steps:

1. Install the adapter as described in "Installing the e-Mail adapter and related files" on page 17.

2. Create the business objects required to support the interface. For more information, see Chapter 4, "Developing business objects for the adapter," on page 29. These business object will include a top-level wrapper business object that contains routing information for the adapter. For more information, see "Using e-Mail adapter business objects" on page 29.

3. Create or modify a meta-object for the data handler that the adapter uses to process the data. For more information, see "Data handler meta-objects" on page 33.

4. Configure the adapter definition as described in "Configuring the adapter definition" on page 19. Keep the following points in mind:
   - You must add the business objects developed for the interface in 3 as supported business objects for the adapter.
   - You must add the data handler meta-object created in 4 as a supported object for the adapter.
   - You must configure the properties of the adapter.

5. Run the adapter as described in "Starting the connector" on page 23.

Using the adapter for error notification

> **Note:** Using the adapter for Error Notification (InterChange Server Integration Broker Only)

If your integration broker is InterChange Server and the adapter will be used for e-mail notification of errors, perform the following steps:

1. The e-Mail adapter is automatically installed if your integration broker is InterChange Server, for the reason that it plays such a significant role in error notification, so you do not have to perform any installation tasks. Proceed to configure the adapter properties as described in "Configuring the adapter definition" on page 19.

2. Configure the Email Notification business object as described in "EmailNotification business object" on page 36.

3. You must configure other InterChange Server components for e-mail notification. For information on setting up e-mail notification of errors, refer to the *System Administration Guide*.

4. Run the adapter as described in "Starting the connector" on page 23.

Installing the e-Mail adapter and related files

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

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

Verifying installation

The sections below describe the paths and filenames of the product after installation and how to verify your adapter installation.

Verifying installation on a Windows system

Before you begin: Install the adapter. The Installer copies the standard files associated with the adapter into your system. The utility installs the connector into the *ProductDir*\connectors directory, and adds a shortcut for the connector to the Start menu.

Perform the following step to verify adapter installation on a Windows system: Change to the directory where you installed the adapter *ProductDir*\ and compare the contents to those listed in Table 2.

Table 2 describes the Windows file structure used by the adapter, and shows the files that are automatically installed when you choose to install the adapter through Installer.

Table 2. Windows file structure for the connector

Subdirectory of *ProductDir*	Description
\connectors\EMail	Contains the adapter application-specific component class files CWEmail.jar and the adapter start file start_Email.bat
\connectors\messages	Contains the error codes and description of the errors used by the connector. Contains the EMailConnector.txt file, as well as the EMailConnector_ll_TT.txt files (message files specific to a language (_ll) and a country or territory (_TT)).
\connectors\EMail\samples	Contains the files used for testing the connector
\connectors\EMail\JavaMail-Sun	Contains the mail.jar archive from JavaMail and the activation.jar from Java Activation Framework.

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

Verifying installation on a Unix system

Before you begin: Install the adapter. The Installer copies the standard files associated with the adapter into your system. The utility installs the connector agent into the *ProductDir*/connectors directory.

Perform the following step to verify adapter installation on a UNIX system: Change to the directory where you installed the adapter `ProductDir/` and compare the contents to those listed in Table 3.

Table 3 describes the UNIX file structure used by the adapter, and shows the files that are automatically installed when you choose to install the adapter through Installer.

Table 3. UNIX file structure for the connector

Subdirectory of *ProductDir*	Description
/connectors/EMail	Contains the adapter application-specific component class files CWEmail.jar and the adapter start file start_Email.sh
/connectors/messages	Contains the error codes and description of the errors used by the connector. Contains the EMailConnector.txt file, as well as the EMailConnector_ll_TT.txt files (message files specific to a language (_ll) and a country or territory (_TT)).
/connectors/EMail/samples	Contains the files used for testing the connector
/connectors/EMail/JavaMail-Sun	Contains the mail.jar archive from JavaMail and the activation.jar from Java Activation Framework.

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

Adapter configuration tasks

There are many configuration tasks for the e-Mail adapter, though only some of them are required, depending on how the adapter is being used. Refer to "Determining installation and configuration tasks" on page 15 to determine which of the following configuration tasks are required for your situation, and then refer to the sections below for specific task information.

This section describes:
- "Configuring mail servers for polling" on page 18
- "Configuring the adapter definition" on page 19
- "Standard configuration properties" on page 19
- "Adapter-specific properties" on page 20

Configuring mail servers for polling

If the e-Mail adapter must poll for e-mail messages to send to the integration broker, you must configure the mail servers to provide the mailboxes and poll locations that the e-Mail adapter uses for polling. To configure mail servers, do the following:

1. Create user accounts that can access the poll locations. Users can be located on a single mail server or on multiple mail servers.
2. Configure the poll locations for each user by creating a poll folder, archive folder, and failed event folder for each poll location. Although users may have multiple poll locations, and poll locations can be located on different mail

servers, the three folders for a poll location must be located in the same mailbox. The adapter connects to these folders using the user name and password of the mailbox owner.

Note that the adapter does not use these mailboxes when sending business object requests. Instead, it uses the mail server name as specified by the SMTP_MailHost adapter configuration property.

Configuring the adapter definition

You must configure the e-Mail adapter before running it. To configure the adapter:
- Set the standard and adapter-specific configuration properties.
- If the adapter must poll for events, then you must configure the required meta-objects for the e-Mail adapter. For information on configuring meta-objects, see "Using e-Mail adapter meta-objects" on page 34.

After configuring the meta-objects, you must add support for them to the adapter definition. To configure an adapter definition, use Connector Configurator (which you launch from System Manager, if your integration broker is ICS), or Connector Configurator (if your integration broker is WebSphere MQ Integrator Broker).

An adapter obtains its configuration values at startup. During a runtime session, you may want to change the values of one or more adapter properties. Changes to some adapter configuration properties may take effect immediately or may require a restart of the adapter or of the whole integration system, depending on the property and the integration system. To determine when changes to the adapter configuration take effect, refer to the Update Method column on the relevant properties tab in the adapter configuration tool.

Standard configuration properties

Standard configuration properties provide information that all adapters use. See Appendix A, "Standard configuration properties for adapters," on page 41 for documentation of these properties.

Important: Because this adapter supports all integration brokers, configuration properties for all brokers are relevant to it.

Table 4 provides information specific to this adapter about configuration properties in the appendix.

Table 4. Property information specific to this adapter

Property	Note
CharacterEncoding	This adapter does not use this property.
ApplicationName	You must provide a value for this configuration property before running the adapter.
Locale	Because this adapter has been internationalized, you can change the value of this property. See release notes for the adapter to determine currently supported locales.

Adapter-specific properties

Adapter-specific configuration properties provide information for the adapter that is specific to the application or technology that the adapter integrates. Adapter-specific properties enable you to change adapter behavior without having to recode and rebuild the application-specific component.

Table 5 lists the adapter-specific configuration properties for the e-Mail adapter. See the sections that follow for explanations of the properties.

Table 5. Adapter-specific configuration properties

Name	Possible values	Default value	Required
ApplicationPassword	password that corresponds to ApplicationUserName		Yes, if Application UserName is set
ApplicationUserName	username required by the SMTP server		No
BusinessObjectMimeCharset	a valid character set value		No
BusinessObjectMimeEncoding	a valid data encoding value		No
DataHandlerConfigMO	Data handler meta-object name	MO_DataHandler_Default	Yes
DebugMode	true or false	false	No
DefaultVerb	a verb value that is valid for the business objects handled by the adapter	Create	No
MailsPerMailBox	Any positive integer	1	No
MessageContentMimeCharset	a valid character set value		No
MessageContentMimeEncoding	a valid data encoding value		No
MessageHeaderMimeCharset	a valid character set value		No
MessageHeaderMimeEncoding	a valid data encoding value		No
PollConfigMO	Meta-object name		No
PollQuantity	Any positive integer	1	No
SMTP_MailPort	Port number for SMTP host		No
UseDefaults	sets the default value for a required attribute	False	No

ApplicationPassword

The password that is associated with the value entered for ApplicationUserName.

ApplicationUserName

The username required for authentication at the SMTP server.

BusinessObjectMimeCharset

The character set for business objects contained in the e-mail. Reference the e-mail RFC documents to determine the proper value for the environment. An example is iso-8859-1, which is the Latin 1 character set of Western European languages.

BusinessObjectMimeEncoding

The encoding for business objects contained in the e-mail. Reference the e-mail RFC documents to determine the proper value for the environment. As an example, you specify Q for "Quoted-Printable" and B for "Base64". If this property is set to a value then business objects are sent as message parts; if it is not specified then business objects are sent as attached files.

DataHandlerConfigMO

Name of the meta-object that the Data Handler uses to set configuration properties. Also used by the DataHandler base class to determine which DataHandler class to use for a particular content type. For information on this meta-object, see "Data handler meta-objects" on page 33.

DebugMode

Specifies whether JavaMail debug messages are sent to STDOUT.

DefaultVerb

Specifies the value that is inserted in the Verb attribute for top-level business objects created by the adapter during event notification if the Verb attribute does not contain a value. In some situations the adapter may poll an event and create a business object, but the original data source (such as an XML document) might not have information in it that directly corresponds to the Verb attribute of the business object definition. If business objects are delivered to the integration broker without a valid verb then they are regarded as not being subscribed to, and do not get processed. You can specify a valid verb (such as `Create`) for this property to populate the Verb attribute of the top-level business objects handled by the adapter in the case that the verb is blank or null.

The default value is `Create`.

MailsPerMailBox

Specifies the number of messages processed in each mailbox before the adapter proceeds to process the next mailbox.

MessageContentMimeCharset

The character set for the message content. Reference the e-mail RFC documents to determine the proper value for the environment. An example is iso-8859-1, which is the Latin 1 character set of Western European languages.

MessageContentMimeEncoding

The encoding for the message content. Reference the e-mail RFC documents to determine the proper value for the environment. As an example, you specify Q for "Quoted-Printable" and B for "Base64".

MessageHeaderMimeCharset

The character set for the message header. Reference the e-mail RFC documents to determine the proper value for the environment. An example is iso-8859-1, which is the Latin 1 character set of Western European languages.

MessageHeaderMimeEncoding

The encoding for the message header. Reference the e-mail RFC documents to determine the proper value for the environment. As an example, you specify Q for "Quoted-Printable" and B for "Base64".

PollConfigMO

The name of the meta-object that the e-Mail adapter uses for polling. For information on the e-Mail adapter meta-object, see "Using e-Mail adapter meta-objects" on page 34.

PollQuantity

Specifies the maximum number of events polled during a single poll call.

SMTP_MailPort

The port number used by the SMTP mail host to send e-mail.

UseDefaults

When this is set to true, default values will be used to set values for required attributes. When this is set to false, or not set, default values will not be used. The default setting is False.

Creating multiple connector instances

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

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

Create a new directory

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

```
ProductDir\connectors\connectorInstance
```

where `connectorInstance` uniquely identifies the connector instance.

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

```
ProductDir\repository\connectorInstance
```

Create business object definitions

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

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

2. Files for the initial connector should reside in the following directory:

   ```
   ProductDir\repository\initialConnectorInstance
   ```

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

Create a connector definition

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

1. Copy the initial connector's configuration file (connector definition) and rename it.

2. Make sure each connector instance correctly lists its supported business objects (and any associated meta-objects).

3. Customize any connector properties as appropriate.

Create a start-up script

To create a startup script:

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

   dirname
2. Put this startup script in the connector directory you created in "Create a new directory" on page 22.
3. Create a startup script shortcut (Windows only).
4. Copy the initial connector's shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

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

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

Starting the connector

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

ProductDir\connectors*connName*

where *connName* identifies the connector.

On UNIX systems the startup script should reside in the *ProductDir*/bin directory.

The name of the startup script depends on the operating-system platform, as Table 6 shows.

Table 6. Startup scripts for a connector

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

When the startup script runs, it expects by default to find the configuration file in the *Productdir* (see the commands below). This is where you place your configuration file.

Note: You need a local configuration file if the adapter is using JMS transport.

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

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

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

    start_*connName connName brokerName* [-c*configFile*]
  - On UNIX-based systems:

    connector_manager -start *connName brokerName* [-c*configFile*]

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 (available only when the broker is WebSphere Application Server or InterChange Server), which is launched when you start System Manager

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

- From System Manager (available for all brokers)

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

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

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

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

Stopping the connector

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

- If you started the connector from the command line, with its connector startup script:

  – On Windows systems, invoking the startup script creates a separate "console" window for the connector. In this window, type "Q" and press Enter to stop the connector.

  – On UNIX-based systems, connectors run in the background so they have no separate window. Instead, run the following command to stop the connector:

  `connector_manager_connName -stop`

  where *connName* is the name of the connector.

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

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

- From System Monitor (WebSphere InterChange Server product only)

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

- On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector stops when the Windows system shuts down.

Chapter 3. Planning an e-Mail adapter implementation

The e-Mail adapter provides for the integration of applications within and across enterprise firewalls using e-mail. This chapter provides information on planning an implementation of the e-Mail adapter. It contains the following sections:

- "Overview of e-Mail adapter implementation"
- "How the e-Mail adapter and data handlers work"
- "Sample e-Mail adapter files" on page 28

Overview of e-Mail adapter implementation

The process of implementing the e-Mail adapter includes a variety of tasks. Here is an overview of the tasks that you must complete before using the e-Mail adapter to send and receive business objects:

1. Determine the format of the data that will be sent and received in e-mail attachments.

2. When you understand the data formats, you can determine what data handlers the adapter will need to convert business objects to e-mail messages and messages to business objects. For information on data handlers, see "How the e-Mail adapter and data handlers work" on page 27.

3. If the adapter must handle business object requests, create the business object definitions that the e-Mail adapter will use. For information, see Chapter 4, "Developing business objects for the adapter," on page 29.

4. Configure the e-Mail adapter with the other components of the integration broker to send or receive the desired business objects via e-mail.

5. If your integration broker is InterChange Server, you may need to modify the data handler top-level meta-object. For adapters, the default top-level meta-object is named MO_DataHandler_Default. To modify this meta-object, use Business Object Designer. For information on data handler meta-objects, see "Data handler meta-objects" on page 33.

6. If polling must be implemented, determine what mail servers the adapter will connect with, and create the mailboxes for the poll locations. For information, see "Configuring mail servers for polling" on page 18.

7. For polling, configure the e-Mail adapter meta-object. This meta-object defines the poll locations for the e-Mail adapter. For more information, see "Using e-Mail adapter meta-objects" on page 34.

8. Configure the adapter by setting adapter configuration properties. For example, specify the name of the top-level data handler meta-object in the e-Mail adapter configuration attribute DataHandlerConfigMO. For information on adapter configuration properties, see "Configuring the adapter definition" on page 19.

How the e-Mail adapter and data handlers work

The e-Mail adapter is designed to use data handlers to convert business objects to specified formats and convert data in specified formats to business objects. A data handler is not part of the adapter but is a separate module that can be customized or replaced as needed.

You can either use data handlers provided by IBM or write your own data handlers. If the data is in a proprietary format or the IBM-delivered data handlers

do not support your application's data format, you must write a custom data handler. For example, IBM provides a data handler that converts between business objects and XML data. If your application generates HTML documents rather than XML documents, you would need to implement a custom data handler that would convert between HTML data and business objects.

Before you begin to work with e-Mail adapter, take time to analyze the data formats that your implementation will need to support. Then use the information in Table 7 to determine whether you can use the data handlers provided by IBM.

Table 7. WebSphere business integration system-delivered data handlers

IBM-delivered data handlers	Description
XML data handler	Converts business objects to and from XML documents. Supports XML 1.0.
NameValue data handler	Parses text data based on named fields, for example, fields that identify the business object type.
FixedWidth data handler	Parses text data using fixed-length fields. The field lengths are specified by the MaxLength property of each business object attribute.
Delimited data handler	Parses text data based on a specified delimiter that separates the individual fields of a business object's data.

For information on data handlers, see the *Data Handler Guide.* If you determine that you must write a custom data handler, see the *Data Handler Guide* for information on how to do this.

Sample e-Mail adapter files

You may want to load and run the e-Mail adapter sample files for an example of the use of the e-Mail adapter. The sample presents a scenario in which the e-mail is used to integrate applications across a firewall.

The files include business objects, collaborations (for use with the InterChange Server integration broker), and a readme file containing instructions on how to configure and run the sample. The sample files are located in connector\Email\samples. To run the sample, you need to configure a mail server and a client e-mail program.

Chapter 4. Developing business objects for the adapter

This chapter describes the structure of the business objects required by the e-Mail adapter. It contains the following sections:

- "Using e-Mail adapter business objects"
- "Data handler meta-objects" on page 33
- "Using e-Mail adapter meta-objects" on page 34
- "EmailNotification business object" on page 36
- "Specifying valid e-mail addresses" on page 38

Using e-Mail adapter business objects

Using the e-Mail adapter, an integration broker can e-mail a message and include business objects or files as attachments to the e-mail. The business objects or files are wrapped in a top-level hierarchical business object that contains routing information for the adapter. The business objects to be sent are defined as children of the top-level business object. If the e-Mail adapter will be used for business object request processing, you must define these wrapper business object definitions.

Note that wrapper business objects need only be designed for request processing. No wrapper business objects are needed for polling. In polling, the adapter looks at the MIME type of each of the e-mail message attachments, instantiates the appropriate data handler, and passes the attachment to the data handler. The data handler determines the name of the business object from the attachment, finds the corresponding business object definition in the list of supported business objects for the adapter, and converts the contents of the attachment into a business object.

Business object structures

The e-Mail adapter top-level business object must contain a set of attributes that describe the content and routing information for an e-mail message. The adapter composes the e-mail message using these attributes. These attributes are listed in Table 8.

Table 8. Required attributes in a top-level e-mail business object

Name	Description
RecipientName	Recipient e-mail address or addresses. For more information, see "Specifying valid e-mail addresses" on page 38.
MessageSubject	Description of the e-mail message. This value is set in the business object by the integration broker.
MessageContent	Content of the e-mail message. This value is set in the business object by the integration broker.
FromAddress	Sender's e-mail address. The sender receives any undelivered messages. For more information, see "Specifying valid e-mail addresses" on page 38.

Table 8. Required attributes in a top-level e-mail business object (continued)

Name	Description
MimeType	Mime type of the attachments. If not specified, the adapter does not process child business objects.
AttachmentExtension	Set the Default Value property of this attribute to the extension that should be given to files that represent business objects contained in the e-mail. For instance, if a business object is sent as an attachment in an e-mail and the Default Value property of this attribute is set to the value txt then the attachment that represents the business object will have an extension of .txt.
BusinessObjectMimeCharset	The character set for business objects contained in the e-mail. For more information about this property, see the description for the identical property at the level of the adapter definition in "BusinessObjectMimeCharset" on page 20.
BusinessObjectMimeEncoding	The encoding for business objects contained in the e-mail. For more information about this property, see the description for the identical property at the level of the adapter definition in "BusinessObjectMimeEncoding" on page 20.
MessageContentMimeCharset	The character set for the message content. For more information about this property, see the description for the identical property at the level of the adapter definition in "MessageContentMimeCharset" on page 21.
MessageContentMimeEncoding	The encoding for the message content. For more information about this property, see the description for the identical property at the level of the adapter definition in "MessageContentMimeEncoding" on page 21.
MessageHeaderMimeCharset	The character set for the message header. For more information about this property, see the description for the identical property at the level of the adapter definition in "MessageHeaderMimeCharset" on page 21.
MessageHeaderMimeEncoding	The encoding for the message header. For more information about this property, see the description for the identical property at the level of the adapter definition in "MessageHeaderMimeEncoding" on page 21.
ReplyTo	ReplyTo address or addresses of e-mail header. For more information, see "Specifying valid e-mail addresses" on page 38.

In addition to the attributes listed in Table 8, the top-level business object can contain the optional attributes listed in Table 9 on page 31.

Table 9. Optional attributes in a top-Level e-mail business object

Name	Description
PriorityLevel	Valid values include Normal, High, and Low. The adapter uses this value to set the priority of outgoing mail.
attributes that represent hierarchical business objects	The adapter converts the child business objects into attachments in the e-mail message.
attributes that specify the names of files	The adapter includes the files as attachments in the e-mail message.

Figure 4 illustrates the basic structure of a top-level business object named Email_TLO_Customer_Wrapper for the e-Mail adapter. This business object contains a hierarchical Customer business object as a child and includes an attribute pointing to a file. The Customer business object and the file are added to the e-mail as attachments.
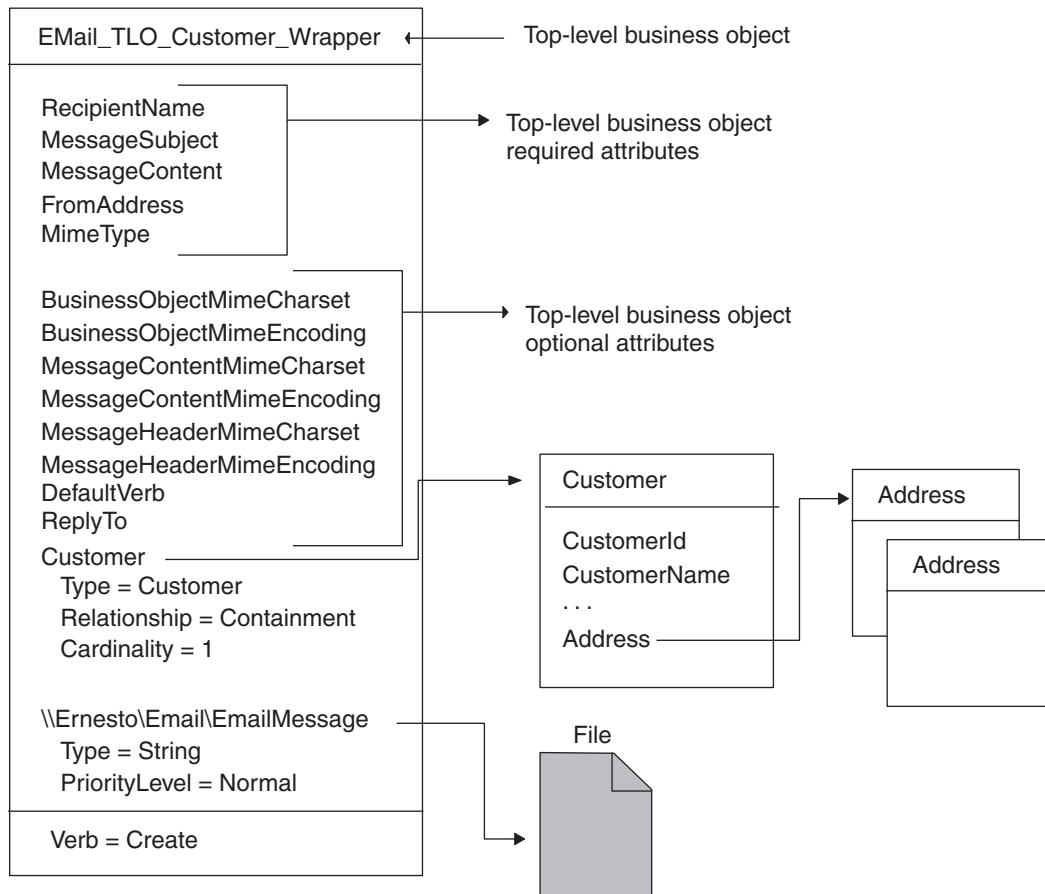

Figure 4. Example e-Mail adapter business object

Attributes specifying business objects

Any business object can be included in the top-level Email business object, as long as the contained business object conforms to the requirements imposed by the configured data handler. If an attribute is a business object, its cardinality must be 1.

When creating business object definitions for the e-Mail adapter, keep in mind that a data handler for the e-Mail adapter must place each business object contained in a top-level business object for the e-Mail adapter in a single attachment. This contained business object may be a large hierarchical business object containing many child business objects of its own, and these children are typically included in the serialized business object in the attachment.

There may be instances when you want to place multiple hierarchical business objects in a single attachment in the top-level e-Mail adapter business object. To do this, wrap the business objects in a parent business object, and define a complex attribute for that parent in the e-Mail adapter top-level business object.

Attributes specifying files

To include a file attachment with an e-mail message, specify the complete filename including the path name. If the file resides on another machine, specify the file name using the UNC naming convention. For example, if the file resides on a machine named `Ernesto`, specify the path name as:
`\\Ernesto\Email\EmailMessage.txt`. You can also map a network drive to the machine and specify the file name as `F:\Email\EmailMessage.txt`.

How business objects and files are processed

When the adapter has processed the basic set of attributes in the top-level business object, it looks for additional attributes. These can be either of type String or type business object. If an attribute is of any other type, the adapter ignores it. The adapter processes each additional non-null attribute as follows:

- If the type of an attribute is string, the adapter treats this as a complete file name. The adapter attaches the file as an attachment to the e-mail message. If the file is not at the location specified in the file name, the adapter returns `BON_FAIL`.

- If the type is a business object, the adapter gets the MimeType attribute of the top-level business object to determine which data handler to use to convert the business object to a string. If the adapter is not able to determine which data handler to use based on the MIME type, or if the MimeType attribute is null, the adapter returns `BON_FAIL`. When a data handler has been instantiated, the adapter passes the business object to the data handler, and the data handler returns a business object string. The adapter puts the business object string in an attachment in the e-mail message. The name of the attachment is the name of the business object, and the MIME type of the attachment is specified by the MimeType attribute of the top-level business object.

After processing all the non-null attributes of the top-level business object successfully, the adapter sends the e-mail.

Business object conformance with data handler requirements

Although you can include any business object in the top-level wrapper business object for the e-Mail adapter, the contained business objects must deliver data in a form that conforms with the requirements of the data handler used to convert the data.

For example, for the BySize data handler, a business object definition must specify a value for the MaxLength attribute property for each business object attribute. For the XML data handler, the business object definition must include application-specific text that enables the data handler to generate an XML document.

A good practice, therefore, is to create your own business object definitions for each type of data to be processed. In the business object definition, provide only the data required by the application and the information required by the data handler. You can then include these business objects in the top-level e-Mail adapter business object.

See the *Data Handler Guide* for information specific to each data handler.

Business object verb processing

When processing business object requests, the e-Mail adapter processes only the Create verb. It returns failure for any other verb. The adapter retains the verb of the child business objects.

For event notification, each e-mail can result in multiple business objects. The application sending the e-mail message is responsible for setting the verb of each business object. The data handler does not process the verbs of these business objects, but sets them in the business objects that it generates.

Business object attribute properties

Business object attributes have properties that can affect how the adapter and integration broker treat those attributes. Table 10 describes how the e-Mail adapter uses these properties for the attributes in the top-level business object.

Table 10. Business object attribute properties

Attribute property	Description
Required	For business object requests, the e-Mail adapter checks whether the Required property is set to True.
Default Values	If the business object does not provide a value for a Required attribute and a default value is specified, the adapter uses the default value.
Max Length	Not used
Type	Not used
Key	Not used
Foreign Key	Not used
Application-Specific Information	Not used

For event notification, the e-Mail adapter does not use business object attribute properties.

Data handler meta-objects

If the e-Mail adapter will be processing business object requests or performing event notification you must set up a data handler meta-object that specifies what data handlers the adapter must use to convert data.

The top-level meta-object for a data handler is a hierarchical business object that can contain any number of child objects. Each child object is a flat business object that represents a specific data handler instance. Child meta-objects have attributes that provide configuration values that enable a data handler instance to do its work. Different types of data handlers require different configuration properties, so the child meta-objects that support specific handlers have different attributes.

To configure a data handler for the e-Mail adapter, do the following:

- Set up the top-level data handler meta-object to have an attribute for each MIME type that the e-Mail adapter must support. The attribute name should be the name of the MIME type. The attribute represents a child meta-object for a data handler instance. The meta-object typically used for adapters is named MO_DataHandler_Default.

  Note that for attachments received in event notifications, the e-Mail adapter converts the MIME type of the attachment into lower case before instantiating the data handler. Therefore, when specifying the MIME type in the data handler meta-object, make sure that the MIME type is in lower case font. Otherwise, the e-Mail adapter is not be able to instantiate the data handler.

- Set the default attribute values for each child meta-object. The attributes for IBM-delivered data handlers are described in the *Data Handler Guide*.

For detailed information on setting up meta-objects for individual data handlers, see the *Data Handler Guide*.

Note: In order for the e-Mail adapter to instantiate a data handler, the data handler top-level meta-object must be configured in the list of business objects supported by the adapter.

Using e-Mail adapter meta-objects

Meta-objects are business objects that are designed to contain configuration information for adapters. A meta-object is required to configure the e-Mail adapter for event notification.

The e-Mail adapter gets the name of its top-level meta-object from the adapter property PollConfigMO. You must create the definition of this meta-object to match the structure described in "Structure of the e-Mail adapter meta-object" on page 34. Then specify its name in the PollConfigMO property. For information on creating a business object definition, see the *Business Object Development Guide*.

Important: In order for the e-Mail adapter to use the configuration information in the meta-object, you must create the meta-object's definition and add it to the list of business objects that the adapter supports.

Structure of the e-Mail adapter meta-object

The meta-object for the e-Mail adapter is a hierarchical business object whose top-level object can contain any number of child objects. Each child object represents one poll location. To configure the adapter to check one or more poll locations, you must set up the meta-object with a corresponding number of child meta-objects. For an illustration of the meta-object definition, see Figure 5 on page 36.

Top-level meta-object attributes

All attributes of the top-level meta-object are container attributes of cardinality 1. For example, if a adapter has two poll locations, Location1 and Location2, the attribute types might be MO_PollLocation1 and MO_PollLocation2.

Child meta-object attributes

Each child meta-object is a flat object with the attributes listed in Table 11.

Table 11. Child meta-object attributes

Name	Description
PollHostName	Name of the host machine running the mail server. The mailbox is located on this server. JavaMail requires this name to connect to the mail store.
UserName	The name of the user. The adapter polls in the mailbox of this user. JavaMail requires this to authenticate the user.
Password	The password of the user. JavaMail requires this to authenticate the user.
PollFolder	Name of the folder in the mailbox. The adapter polls in this folder. This folder should exist in the mailbox of the user. The default value is PollFolder. BiDi supported for Windows. All BiDi enabled properties will be transformed from Windows BiDi format into BiDi.Metadata format, which is an e-Mail connector standard property.
ArchiveFolder	Name of the folder in the mailbox. The adapter archives the successful messages (successful events) in this folder. This folder should exist in the mailbox of the user. If a value is not specified for this attribute, the adapter deletes successful messages after processing them. BiDi supported for Windows. All BiDi enabled properties will be transformed from Windows BiDi format into BiDi.Metadata format, which is an e-Mail connector standard property.
FailFolder	Name of the folder in the mailbox. The adapter archives the failed messages (failed events) in this folder. If a value is not specified for this attribute, the adapter deletes failed messages. BiDi supported for Windows. All BiDi enabled properties will be transformed from Windows BiDi format into BiDi.Metadata format, which is an e-Mail connector standard property.
InDoubtEvents	Valid values for this attribute are `Reprocess`, `FailOnStartUp`, `LogError`, and `Ignore`. If there are in-doubt events in this poll location, during initialization the adapter processes them based on the default value set for this attribute. If the value is set to `Reprocess`, the adapter resets the SEEN flag of the in-doubt events in the poll folder, and picks up these messages in the subsequent poll calls. If it is set to `FailOnStartUp` and there are in-doubt transactions in this poll folder, the adapter fails. If it is set to `LogError` and there are in-doubt transactions in this poll folder, the adapter logs the error and continues. If it is set to `Ignore` and there are in-doubt transactions in the poll folder, the adapter simply ignores them. The default value is `Reprocess`.

Table 11. Child meta-object attributes (continued)

Name	Description
Reconnect	Values for this attribute are True or False. This value is used to determine whether a reconnect attempt should be made for each poll cycle in the event of a connection loss.

Meta-object example

Figure 5 shows an example of a top-level meta-object named MO_Email_Default. The e-Mail adapter uses this meta-object to poll two locations, which are specified in the two child meta-objects, MO_PollLocation1 and MO_PollLocation2.

In the example, the value of the EventRecovery attribute in MO_PollLocation1 is set to Resubmit, and the value of this attribute in MO_PollLocation2 is set to Ignore. The value of the EventRecovery attribute informs the adapter administrator to resubmit in-doubt events for Location1 and ignore in-doubt events for Location2.
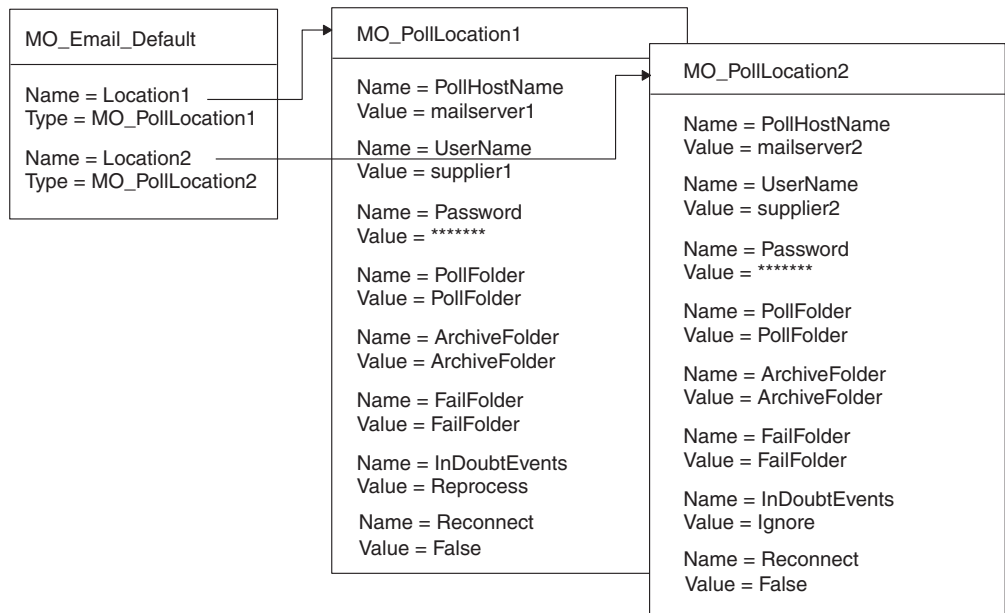
```
┌──────────────────────┐      ┌──────────────────────┐
│ MO_Email_Default     │      │ MO_PollLocation1     │
├──────────────────────┤      ├──────────────────────┐     ┌──────────────────────┐
│                      │      │                      │     │ MO_PollLocation2     │
│ Name = Location1     │──┐   │ Name = PollHostName  │     ├──────────────────────┤
│ Type = MO_PollLocation1 │  │ Value = mailserver1  │     │                      │
│                      │  │   │                      │     │ Name = PollHostName  │
│ Name = Location2     │  │   │ Name = UserName      │──┐  │ Value = mailserver2  │
│ Type = MO_PollLocation2 │  │ Value = supplier1    │  │  │                      │
└──────────────────────┘  │   │                      │  │  │ Name = UserName      │
                          │   │ Name = Password      │  │  │ Value = supplier2    │
                          │   │ Value = *******      │  │  │                      │
                          │   │                      │  │  │ Name = Password      │
                          │   │ Name = PollFolder    │  │  │ Value = *******      │
                          │   │ Value = PollFolder   │  │  │                      │
                          │   │                      │  │  │ Name = PollFolder    │
                          │   │ Name = ArchiveFolder │  │  │ Value = PollFolder   │
                          │   │ Value = ArchiveFolder│  │  │                      │
                          │   │                      │  │  │ Name = ArchiveFolder │
                          │   │ Name = FailFolder    │  │  │ Value = ArchiveFolder│
                          │   │ Value = FailFolder   │  │  │                      │
                          │   │                      │  │  │ Name = FailFolder    │
                          │   │ Name = InDoubtEvents │  │  │ Value = FailFolder   │
                          │   │ Value = Reprocess    │  │  │                      │
                          │   │                      │  │  │ Name = InDoubtEvents │
                          │   │ Name = Reconnect     │  │  │ Value = Ignore       │
                          │   │ Value = False        │  │  │                      │
                          │   └──────────────────────┘  │  │ Name = Reconnect     │
                          │                             │  │ Value = False        │
                          └─────────────────────────────┘  └──────────────────────┘
```

Figure 5. e-Mail adapter meta-object example

EmailNotification business object

> **Note:** EmailNotification Business Object (InterChange Server Integration Broker Only)

If your integration broker is InterChange Server then a business object named EmailNotification is automatically created by the system to support error notification. When an error occurs, the EmailNotification business object is instantiated by the system and sent to the e-Mail adapter. The adapter extracts the message information from the business object, creates the e-mail message, and sends the message.

The EmailNotification business object contains the attributes listed in Table 12 on page 37. The only supported verb for this business object is Create.

Table 12. EmailNotification business object attributes

Name	Description	Required?
RecipientName	Message recipient. For more information, see "Specifying valid e-mail addresses" on page 38.	Yes
MessageSubject	Description of e-mail content. Value set in business object by collaboration.	Yes. If empty, the adapter uses the default value.
MessageContent	Content of the e-mail message. Value set in business object by collaboration.	Yes. If empty, the adapter uses the default value.
MessageAttachment	File to be attached with the message.	Optional. If empty, no attachment is sent with the message. To include a file attachment with an e-mail message, set this attribute to the complete filename including the path.
FromAddress	Recipient of undelivered messages. Value defined in business object as part of configuration. For more information, see "Specifying valid e-mail addresses" on page 38.	Optional. If empty, no undelivered messages are returned. To have undelivered messages delivered to an administrator, specify the desired e-mail address in this attribute and enable its Required attribute property.
BusinessObjectMimeCharset	The character set for business objects contained in the e-mail. For more information about this property, see the description for the identical property at the level of the adapter definition in "BusinessObjectMimeCharset" on page 20.	No
BusinessObjectMimeEncoding	The encoding for business objects contained in the e-mail. For more information about this property, see the description for the identical property at the level of the adapter definition in "BusinessObjectMimeEncoding" on page 20.	No
MessageContentMimeCharset	The character set for the message content. For more information about this property, see the description for the identical property at the level of the adapter definition in "MessageContentMimeCharset" on page 21.	No
MessageContentMimeEncoding	The encoding for the message content. For more information about this property, see the description for the identical property at the level of the adapter definition in "MessageContentMimeEncoding" on page 21.	No
MessageHeaderMimeCharset	The character set for the message header. For more information about this property, see the description for the identical property at the level of the adapter definition in "MessageHeaderMimeCharset" on page 21.	No
MessageHeaderMimeEncoding	The encoding for the message header. For more information about this property, see the description for the identical property at the level of the adapter definition in "MessageHeaderMimeEncoding" on page 21.	No

Table 12. EmailNotification business object attributes (continued)

Name	Description	Required?
ReplyTo	ReplyTo address(es) of the e-mail message Inserts the value of the MessageContent attribute into the CONTENT field. Inserts the value of the MessageAttachment attribute as an attachment to the mail. Inserts the value of the FromAddress attribute into the FROM field. Inserts the value of the ReplyTo attribute into the REPLYTO field.	No

Note: No e-mail is sent if an attachment filename is incorrect, an attachment file does not exist, or an attachment is not readable. If the file resides on another machine, specify the file name using the UNC naming convention. For example, if the file resides on a machine named Ernesto, the path name should be specified as: `\\Ernesto\Email\EmailMessage`. You can also map a network drive to the machine and specify the file name as `F:\Email\EmailMessage`.

At runtime, the adapter extracts the attribute values from the business object and inserts the values into an e-mail message as follows:

- Inserts one or more the message recipients as defined in the RecipientName attribute into the TO field.
- Inserts the value of the MessageSubject attribute into the SUBJECT field.
- Inserts the value of the MessageContent attribute into the CONTENT field.
- Inserts the value of the MessageAttachment attribute as an attachment to the mail.
- Inserts the value of the FromAddress attribute into the FROM field.

Specifying valid e-mail addresses

A valid e-mail address entry can be one or more fully qualified Internet addresses separated by commas as specified by RFC 822. For example, a valid entry for two recipients is:

`JohnDoe@company.com,"/FredSmith/" <FredSmith@company.com>`

The adapter does not resolve personal address aliases, such as the `EngineeringDepartment` alias defined in a personal address book. However, a valid address can be an alias defined in a mail server, such as `EngineeringDepartment` or `EngineeringDepartment@company.com`. In this case, the mail server will decode the alias and send e-mail to all members of the alias.

The e-Mail adapter parses each individual address to check whether its complaint to RFC 822. Examples of valid address entries are:

- `name@company.com`
- `EngineeringDepartment` or `EngineeringDepartment@company.com`, where `EngineeringDepartment` is an alias defined in the mail server. The mail server decodes the alias and sends e-mail to all members of the alias.
- `person1@some_company.com, person2@another_company.com`. Multiple addresses can be entered and must be separated by commas. For example,   `"NAME1" <name1@company.com>, <name2@company.com>`.

If an e-mail address requires an at (@) sign, it must be included in the address, since the adapter will not add characters to an e-mail address.

Appendix A. Standard configuration properties for adapters

This appendix describes the standard configuration properties for the connector component of WebSphere Business Integration adapters. The information covers connectors running with 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 (and shown as WMQI in the Connector Configurator).
- Information Integrator (II)
- WebSphere Application Server (WAS)

If your adapter supports DB2 Information Integrator, use the WMQI options and the DB2 II standard properties (see the Notes column in Table 13 on page 43.)

The properties you set for the adapter depend on which integration broker you use. You choose the integration broker using Connector Configurator. After you choose the broker, Connector Configurator lists the standard properties you must configure for the adapter.

For information about properties specific to this connector, see the relevant section in this guide.

New properties

These standard properties have been added in this release:

- AdapterHelpName
- BiDi.Application
- BiDi.Broker
- BiDi.Metadata
- BiDi.Transformation
- CommonEventInfrastructure
- CommonEventInfrastructureContextURL
- ControllerEventSequencing
- jms.ListenerConcurrency
- jms.TransportOptimized
- ResultsSetEnabled
- ResultsSetSize
- TivoliTransactionMonitorPerformance

Standard connector properties overview

Connectors have two types of configuration properties:
- Standard configuration properties, which are used by the framework
- Application, or connector-specific, configuration properties, which are used by the agent

These properties determine the adapter framework and the agent run-time behavior.

This section describes how to start Connector Configurator and describes characteristics common to all properties. For information on configuration properties specific to a connector, see its adapter user guide.

Starting 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 sections on Connector Configurator in this guide.

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.

Configuration property values overview

The connector uses the following order to determine a property's value:
1. Default
2. Repository (valid only if WebSphere InterChange Server (ICS) is the integration broker)
3. Local configuration file
4. Command line

The default length of a property field is 255 characters. There is no limit on the length of a STRING property type. The length of an INTEGER type is determined by the server on which the adapter is running.

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.

The update characteristics of a property, that is, how and when a change to the connector properties takes effect, depend on the nature of the property.

There are four update methods for standard connector properties:
- **Dynamic**
  The new value takes effect immediately after the change is saved in System Manager. However, if the connector is in stand-alone mode (independently of System Manager), for example, if it is running with one of the WebSphere message brokers, you can change properties only through the configuration file. In this case, a dynamic update is not possible.
- **Agent restart (ICS only)**
  The new value takes effect only after you stop and restart the connector agent.
- **Component restart**
  The new value takes effect only after the connector is stopped and then restarted in System Manager. You do not need to stop and restart the agent or the server process.

- **System restart**
  The new value takes effect only after you stop and restart the connector agent and the server.

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 Table 13 on page 43.

There are three locations in which a standard property can reside. Some properties can reside in more than one location.

- **ReposController**
  The property resides in the connector controller and is effective only there. If you change the value on the agent side, it does not affect the controller.

- **ReposAgent**
  The property resides in the agent and is effective only there. A local configuration can override this value, depending on the property.

- **LocalConfig**
  The property resides in the configuration file for the connector and can act only through the configuration file. The controller cannot change the value of the property, and is not aware of changes made to the configuration file unless the system is redeployed to update the controller explicitly.

Standard properties quick-reference

Table 13 provides a quick-reference to the standard connector configuration properties. Not all connectors require all of these properties, and property settings may differ from integration broker to integration broker.

See the section following the table for a description of each property.

Note: In the Notes column in Table 13, the phrase "RepositoryDirectory is set to <REMOTE>" indicates that the broker is InterChange Server. When the broker is WMQI or WAS, the repository directory is set to *<ProductDir>*\repository

Table 13. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdapterHelpName	One of the valid subdirectories in *<ProductDir>*\bin\Data \App\Help\ that contains a valid *<RegionalSetting>* directory	Template name, if valid, or blank field	Component restart	Supported regional settings. Include chs_chn, cht_twn, deu_deu, esn_esp, fra_fra, ita_ita, jpn_jpn, kor_kor, ptb_bra, and enu_usa (default).
AdminInQueue	Valid JMS queue name	*<CONNECTORNAME>* /ADMININQUEUE	Component restart	This property is valid only when the value of DeliveryTransport is JMS
AdminOutQueue	Valid JMS queue name	*<CONNECTORNAME>* /ADMINOUTQUEUE	Component restart	This property is valid only when the value of DeliveryTransport is JMS

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
AgentConnections	1 through 4	1	Component restart	This property is valid only when the value of DeliveryTransport is MQ or IDL, the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
AgentTraceLevel	0 through 5	0	Dynamic if broker is ICS; otherwise Component restart	
ApplicationName	Application name	The value specified for the connector application name	Component restart	
BiDi.Application	Any valid combination of these bidirectional attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (five letters)	Component restart	This property is valid only if the value of BiDi.Transforma tion is true
BiDi.Broker	Any valid combination of these bidirectional attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (five letters)	Component restart	This property is valid only if the value of BiDi.Transformation is true. If the value of BrokerType is ICS, the property is read-only.
BiDi.Metadata	Any valid combination of these bidirectional attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (five letters)	Component restart	This property is valid only if the value of BiDi.Transformation is true.
BiDi.Transformation	true or false	false	Component restart	This property is valid only if the value of BrokerType is not WAS.
BrokerType	ICS, WMQI, WAS	ICS	Component restart	
CharacterEncoding	Any supported code. The list shows this subset: ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437 .	ascii7	Component restart	This property is valid only for C++ connectors.

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
CommonEventInfrastructure	true or false	false	Component restart	
CommonEventInfrastructureURL	A URL string, for example, corbaloc:iiop: host:2809.	No default value.	Component restart	This property is valid only if the value of CommonEvent Infrastructure is true.
ConcurrentEventTriggeredFlows	1 through 32,767	1	Component restart	This property is valid only if the value of RepositoryDirectory is set to <REMOTE> and the value of BrokerType is ICS.
ContainerManagedEvents	Blank or JMS	Blank	Component restart	This property is valid only when the value of Delivery Transport is JMS.
ControllerEventSequencing	true or false	true	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ControllerStoreAndForwardMode	true or false	true	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ControllerTraceLevel	0 through 5	0	Dynamic	This property is valid only if the value of RepositoryDirectory is set to <REMOTE> and the value of BrokerType is ICS.
DeliveryQueue	Any valid JMS queue name	<CONNECTORNAME> /DELIVERYQUEUE	Component restart	This property is valid only when the value of Delivery Transport is JMS.
DeliveryTransport	MQ, IDL, or JMS	IDL when the value of RepositoryDirectory is <REMOTE>, otherwise JMS	Component restart	If the value of RepositoryDirectory is not <REMOTE>, the only valid value for this property is JMS.
DuplicateEventElimination	true or false	false	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
EnableOidForFlowMonitoring	true or false	false	Component restart	This property is valid only if the value of BrokerType is ICS.
FaultQueue	Any valid queue name.	<CONNECTORNAME> /FAULTQUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
jms.FactoryClassName	CxCommon.Messaging.jms .IBMMQSeriesFactory, CxCommon.Messaging .jms.SonicMQFactory, or any Java class name	CxCommon.Messaging. jms.IBMMQSeriesFactory	Component restart	This property is valid only if the value of DeliveryTransport is JMS.

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
jms.ListenerConcurrency	1 through 32767	1	Component restart	This property is valid only if the value of jms.TransportOptimized is `true`.
jms.MessageBrokerName	If the value of jms.FactoryClassName is IBM, use `crossworlds.queue. manager.`	`crossworlds.queue. manager`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS`.
jms.NumConcurrent Requests	Positive integer	10	Component restart	This property is valid only if the value of DeliveryTransport is `JMS`.
jms.Password	Any valid password		Component restart	This property is valid only if the value of DeliveryTransport is `JMS`.
jms.TransportOptimized	`true` or `false`	`false`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS` and the value of BrokerType is ICS.
jms.UserName	Any valid name		Component restart	This property is valid only if the value of Delivery Transport is `JMS`.
JvmMaxHeapSize	Heap size in megabytes	128m	Component restart	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
JvmMaxNativeStackSize	Size of stack in kilobytes	128k	Component restart	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
JvmMinHeapSize	Heap size in megabytes	1m	Component restart	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ListenerConcurrency	1 through 100	1	Component restart	This property is valid only if the value of DeliveryTransport is `MQ`.
Locale	This is a subset of the supported locales: `en_US, ja_JP, ko_KR, zh_CN, zh_TW, fr_FR, de_DE, it_IT, es_ES, pt_BR`	`en_US`	Component restart	

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
LogAtInterchangeEnd	true or false	false	Component restart	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
MaxEventCapacity	1 through 2147483647	2147483647	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
MessageFileName	Valid file name	InterchangeSystem.txt	Component restart	
MonitorQueue	Any valid queue name	`<CONNECTORNAME>` `/MONITORQUEUE`	Component restart	This property is valid only if the value of DuplicateEventElimination is true and ContainerManagedEvents has no value.
OADAutoRestartAgent	true or false	false	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
OADMaxNumRetry	A positive integer	1000	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
OADRetryTimeInterval	A positive integer in minutes	10	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
PollEndTime	HH = 0 through 23 MM = 0 through 59	HH:MM	Component restart	
PollFrequency	A positive integer (in milliseconds)	10000	Dynamic if broker is ICS; otherwise Component restart	
PollQuantity	1 through 500	1	Agent restart	This property is valid only if the value of ContainerManagedEvents is JMS.
PollStartTime	HH = 0 through 23 MM = 0 through 59	HH:MM	Component restart	
RepositoryDirectory	<REMOTE> if the broker is ICS; otherwise any valid local directory.	For ICS, the value is set to <REMOTE> For WMQI and WAS, the value is `<ProductDir` `\repository`	Agent restart	

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
RequestQueue	Valid JMS queue name	`<CONNECTORNAME>` `/REQUESTQUEUE`	Component restart	This property is valid only if the value of DeliveryTransport is JMS
ResponseQueue	Valid JMS queue name	`<CONNECTORNAME>` `/RESPONSEQUEUE`	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
RestartRetryCount	0 through 99	3	Dynamic if ICS; otherwise Component restart	
RestartRetryInterval	A value in minutes from 1 through 2147483647	1	Dynamic if ICS; otherwise Component restart	
ResultsSetEnabled	`true` or `false`	`false`	Component restart	Used only by connectors that support DB2II. This property is valid only if the value of DeliveryTransport is JMS, and the value of BrokerType is WMQI.
ResultsSetSize	Positive integer	0 (means the results set size is unlimited)	Component restart	Used only by connectors that support DB2II. This property is valid only if the value of ResultsSetEnabled is `true`.
RHF2MessageDomain	`mrm` or `xml`	`mrm`	Component restart	This property is valid only if the value of DeliveryTransport is JMS and the value of WireFormat is CwXML.
SourceQueue	Any valid WebSphere MQ queue name	`<CONNECTORNAME>` `/SOURCEQUEUE`	Agent restart	This property is valid only if the value of ContainerManagedEvents is JMS.
SynchronousRequest Queue	Any valid queue name.	`<CONNECTORNAME>` `/SYNCHRONOUSREQUEST` `QUEUE`	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
SynchronousRequest Timeout	0 to any number (milliseconds)	0	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
SynchronousResponse Queue	Any valid queue name	`<CONNECTORNAME>` `/SYNCHRONOUSRESPONSE` `QUEUE`	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
TivoliMonitorTransaction Performance	`true` or `false`	`false`	Component restart	

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
WireFormat	CwXML or CwBO	CwXML	Agent restart	The value of this property must be CwXML if the value of RepositoryDirectory is not set to <REMOTE>. The value must be CwBO if the value of RepositoryDirectory is set to <REMOTE>.
WsifSynchronousRequest Timeout	0 to any number (milliseconds)	0	Component restart	This property is valid only if the value of BrokerType is WAS.
XMLNameSpaceFormat	short or long	short	Agent restart	This property is valid only if the value of BrokerType is WMQI or WAS

Standard properties

This section describes the standard connector configuration properties.

AdapterHelpName

The AdapterHelpName property is the name of a directory in which connector-specific extended help files are located. The directory must be located in *<ProductDir>*\bin\Data\App\Help and must contain at least the language directory enu_usa. It may contain other directories according to locale.

The default value is the template name if it is valid, or it is blank.

AdminInQueue

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

The default value is *<CONNECTORNAME>*/ADMININQUEUE

AdminOutQueue

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

The default value is *<CONNECTORNAME>*/ADMINOUTQUEUE

AgentConnections

The AgentConnections property controls the number of ORB (Object Request Broker) connections opened when the ORB initializes.

It is valid only if the value of the RepositoryDirectory is set to <REMOTE> and the value of the DeliveryTransport property is MQ or IDL.

The default value of this property is 1.

AgentTraceLevel

The AgentTraceLevel property sets the level of trace messages for the application-specific component. The connector delivers all trace messages applicable at the tracing level set and lower.

The default value is 0.

ApplicationName

The ApplicationName property uniquely identifies the name of the connector application. This name is used by the system administrator to monitor the integration environment. This property must have a value before you can run the connector.

The default is the name of the connector.

BiDi.Application

The BiDi.Application property specifies the bidirectional format for data coming from an external application into the adapter in the form of any business object supported by this adapter. The property defines the bidirectional attributes of the application data. These attributes are:

- Type of text: implicit or visual (I or V)
- Text direction: left-to-right or right-to-left (L or R)
- Symmetric swapping: on or off (Y or N)
- Shaping (Arabic): on or off (S or N)
- Numerical shaping (Arabic): Hindi, contextual, or nominal (H, C, or N)

This property is valid only if the BiDi.Transformation property value is set to true.

The default value is ILYNN (implicit, left-to-right, on, off, nominal).

BiDi.Broker

The BiDi.Broker property specifies the bidirectional format for data sent from the adapter to the integration broker in the form of any supported business object. It defines the bidirectional attributes of the data, which are as listed under BiDi.Application above.

This property is valid only if the BiDi.Transformation property value is set to true. If the BrokerType property is ICS, the property value is read-only.

The default value is ILYNN (implicit, left-to-right, on, off, nominal).

BiDi.Metadata

The BiDi.Metadata property defines the bidirectional format or attributes for the metadata, which is used by the connector to establish and maintain a link to the external application. The attribute settings are specific to each adapter using the bidirectional capabilities. If your adapter supports bidirectional processing, refer to section on adapter-specific properties for more information.

This property is valid only if the BiDi.Transformation property value is set to true.

The default value is ILYNN (implicit, left-to-right, on, off, nominal).

BiDi.Transformation

The BiDi.Transformation property defines whether the system performs a bidirectional transformation at run time.

If the property value is set to `true`, the BiDi.Application, BiDi.Broker, and BiDi.Metadata properties are available. If the property value is set to `false`, they are hidden.

The default value is `false`.

BrokerType

The BrokerType property identifies the integration broker type that you are using. The possible values are ICS, WMQI (for WMQI, WMQIB or WBIMB), or WAS.

CharacterEncoding

The CharacterEncoding property 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. C++ connectors use the value `ascii7` for this property.

By default, only a subset of supported character encodings is displayed. To add other supported values to the list, you must manually modify the \Data\Std\stdConnProps.xml file in the product directory (<*ProductDir*>). For more information, see the Connector Configurator appendix in this guide.

CommonEventInfrastructure

The Common Event Infrastructure (CEI) is a simple event management function handling generated events. The CommonEventInfrastructure property specifies whether the CEI should be invoked at run time.

The default value is `false`.

CommonEventInfrastructureContextURL

The CommonEventInfrastructureContextURL is used to gain access to the WAS server that executes the Common Event Infrastructure (CEI) server application. This property specifies the URL to be used.

This property is valid only if the value of CommonEventInfrastructure is set to `true.`

The default value is a blank field.

ConcurrentEventTriggeredFlows

The ConcurrentEventTriggeredFlows property determines how many business objects can be concurrently processed by the connector for event delivery. You set the value of this attribute to the number of business objects that are mapped and delivered concurrently. For example, if you set the value of this property to 5, five business objects are processed concurrently.

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), the following properties must configured:

- The collaboration must be configured to use multiple threads by setting its `Maximum number of concurrent events` property high enough to use multiple threads.
- The destination application's application-specific component must be configured to process requests concurrently. That is, it must be multithreaded, or it must be able to use connector agent parallelism and be configured for multiple processes. The Parallel Process Degree configuration property must be set to a value larger than 1.

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

This property is valid only if the value of the RepositoryDirectory property is set to <REMOTE>.

The default value is 1.

ContainerManagedEvents

The ContainerManagedEvents 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 one JMS transaction.

When this property is set to JMS, the following properties must also be set to enable guaranteed event delivery:

- PollQuantity = 1 to 500
- SourceQueue = /SOURCEQUEUE

You must also configure a data handler with the MimeType and DHClass (data handler class) properties. You can also add DataHandlerConfigMOName (the meta-object name, which is optional). To set those values, use the **Data Handler** tab in Connector Configurator.

Although these properties are adapter-specific, here are some example values:

- MimeType = text\xml
- DHClass = com.crossworlds.DataHandlers.text.xml
- DataHandlerConfigMOName = MO_DataHandler_Default

The fields for these values in the **Data Handler** tab are displayed only if you have set the ContainerManagedEvents property to the value JMS.

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

The ContainerManagedEvents property is valid only if the value of the DeliveryTransport property is set to JMS.

There is no default value.

ControllerEventSequencing

The ControllerEventSequencing property enables event sequencing in the connector controller.

This property is valid only if the value of the RepositoryDirectory property is set to set to <REMOTE> (`BrokerType` is ICS).

The default value is `true`.

ControllerStoreAndForwardMode

The ControllerStoreAndForwardMode property 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.

This property is valid only if the value of the RepositoryDirectory property is set to <REMOTE> (the value of the BrokerType property is ICS).

The default value is `true`.

ControllerTraceLevel

The ControllerTraceLevel property sets the level of trace messages for the connector controller.

This property is valid only if the value of the RepositoryDirectory property is set to set to <REMOTE>.

The default value is `0`.

DeliveryQueue

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

This property is valid only if the value of the DeliveryTransport property is set to `JMS`.

The default value is `<CONNECTORNAME>/DELIVERYQUEUE`.

DeliveryTransport

The DeliveryTransport property 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 the value of the RepositoryDirectory property is set to <REMOTE>, the value of the DeliveryTransport property can be MQ, IDL, or JMS, and the default is IDL.
- If the value of the RepositoryDirectory property is a local directory, the value can be only JMS.

The connector sends service-call requests and administrative messages over CORBA IIOP if the value of the RepositoryDirectory property is MQ or IDL.

The default value is JMS.

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 prevents writing 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 polling thread picks up an event, places it in the connector queue, then picks up the next event. This is faster than IDL, which requires the connector polling thread to pick up an event, go across the network into the server process, store the event persistently in the repository database, then pick up the next event.

JMS

The JMS transport mechanism 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 are listed in Connector Configurator. The properties jms.MessageBrokerName and jms.FactoryClassName are required for this transport.

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
- 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 768MB of process heap size, set the following variable and property:

- Set the LDR_CNTRL environment variable in the CWSharedEnv.sh script.

This script is located in the \bin directory below the product directory (*<ProductDir>*). Using 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 larger than this limit, page swapping can occur, which can adversely affect the performance of your system.

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

DuplicateEventElimination

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

Note: When the value of this property is `true`, the MonitorQueue property must be enabled to provide guaranteed event delivery.

The default value is `false`.

EnableOidForFlowMonitoring

When the value of this property is `true`, the adapter runtime will mark the incoming ObjectEventID as a foreign key for flow monitoring.

This property is only valid if the BrokerType property is set to ICS.

The default value is `false`.

FaultQueue

If the connector experiences an error while processing a message, it moves the message (and a status indicator and description of the problem) to the queue specified in the FaultQueue property.

The default value is `<CONNECTORNAME>/FAULTQUEUE`.

jms.FactoryClassName

The jms.FactoryClassName property specifies the class name to instantiate for a JMS provider. This property must be set if the value of the DeliveryTransport property is JMS.

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

jms.ListenerConcurrency

The jms.ListenerConcurrency property specifies the number of concurrent listeners for the JMS controller. It specifies the number of threads that fetch and process messages concurrently within a controller.

This property is valid only if the value of the jms.OptimizedTransport property is `true`.

The default value is 1.

jms.MessageBrokerName

The jms.MessageBrokerName specifies the broker name to use for the JMS provider. You must set this connector property if you specify JMS as the delivery transport mechanism (in the DeliveryTransport property).

When you connect to a remote message broker, this property requires the following values:
`QueueMgrName:Channel:HostName:PortNumber`
where:
`QueueMgrName` is the name of the queue manager.
`Channel` is the channel used by the client.
`HostName` is the name of the machine where the queue manager is to reside.
`PortNumber` is the port number used by the queue manager for listening

For example:
`jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456`

The default value is `crossworlds.queue.manager`. Use the default when connecting to a local message broker.

jms.NumConcurrentRequests

The jms.NumConcurrentRequests property 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 are blocked and must wait for another request to complete before proceeding.

The default value is 10.

jms.Password

The jms.Password property specifies the password for the JMS provider. A value for this property is optional.

There is no default value.

jms.TransportOptimized

The jms.TransportOptimized property determines if the WIP (work in progress) is optimized. You must have a WebSphere MQ provider to optimize the WIP. For optimized WIP to operate, the messaging provider must be able to:

1. Read a message without taking it off the queue
2. Delete a message with a specific ID without transferring the entire message to the receiver's memory space
3. Read a message by using a specific ID (needed for recovery purposes)
4. Track the point at which events that have not been read appear.

The JMS APIs cannot be used for optimized WIP because they do not meet conditions 2 and 4 above, but the MQ Java APIs meet all four conditions, and hence are required for optimized WIP.

This property is valid only if the value of DeliveryTransport is JMS and the value of BrokerType is ICS.

The default value is `false`.

jms.UserName

the jms.UserName property specifies the user name for the JMS provider. A value for this property is optional.

There is no default value.

JvmMaxHeapSize

The JvmMaxHeapSize property specifies the maximum heap size for the agent (in megabytes).

This property is valid only if the value for the RepositoryDirectory property is set to <REMOTE>.

The default value is 128m.

JvmMaxNativeStackSize

The JvmMaxNativeStackSize property specifies the maximum native stack size for the agent (in kilobytes).

This property is valid only if the value for the RepositoryDirectory property is set to <REMOTE>.

The default value is 128k.

JvmMinHeapSize

The JvmMinHeapSize property specifies the minimum heap size for the agent (in megabytes).

This property is valid only if the value for the RepositoryDirectory property is set to <REMOTE>.

The default value is 1m.

ListenerConcurrency

The ListenerConcurrency property supports multithreading in WebSphere MQ Listener when ICS is the integration broker. It enables batch writing of multiple events to the database, thereby improving system performance.

This property valid only with connectors that use MQ transport. The value of the DeliveryTransport property must be MQ.

The default value is 1.

Locale

The Locale property specifies the language code, country or territory, and, optionally, the associated character code set. The value of this property determines cultural conventions such 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 is a two-character language code (in lowercase letters)
TT is a two-letter country or territory code (in uppercase letters)
codeset is the name of the associated character code set (may be optional).

By default, only a subset of supported locales are listed. To add other supported values to the list, you modify the \Data\Std\stdConnProps.xml file in the *<ProductDir>*\bin directory. For more information, refer to the Connector Configurator appendix in this guide.

If the connector has not been internationalized, the only valid value for this property is en_US. To determine whether a specific connector has been globalized, refer to the user guide for that adapter.

The default value is en_US.

LogAtInterchangeEnd

The LogAtInterchangeEnd property specifies whether to log errors to the log destination of the integration broker.

Logging to the log destination also turns on e-mail notification, which generates e-mail messages for the recipient specified as the value of MESSAGE_RECIPIENT in the InterchangeSystem.cfg file when errors or fatal errors occur. For example, when a connector loses its connection to the application, if the value of LogAtInterChangeEnd is true, an e-mail message is sent to the specified message recipient.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

The default value is false.

MaxEventCapacity

The MaxEventCapacity property specifies maximum number of events in the controller buffer. This property is used by the flow control feature.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

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

The default value is 2147483647.

MessageFileName

The MessageFileName property specifies the name of the connector message file. The standard location for the message file is \connectors\messages in the product directory. Specify the message file name 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 connector has its own message file, see the individual adapter user guide.

The default value is `InterchangeSystem.txt`.

MonitorQueue

The MonitorQueue property specifies the logical queue that the connector uses to monitor duplicate events.

It is valid only if the value of the DeliveryTransport property is JMS and the value of the DuplicateEventElimination is `true`.

The default value is `<CONNECTORNAME>/MONITORQUEUE`

OADAutoRestartAgent

the OADAutoRestartAgent property specifies whether the connector uses the automatic and remote restart feature. This feature uses the WebSphere 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 WebSphere MQ-triggered OAD feature. see the *Installation Guide for Windows* or *for UNIX*.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

The default value is `false`.

OADMaxNumRetry

The OADMaxNumRetry property specifies the maximum number of times that the WebSphere MQ-triggered Object Activation Daemon (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.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

The default value is `1000`.

OADRetryTimeInterval

The OADRetryTimeInterval property specifies the number of minutes in the retry-time interval for the WebSphere MQ-triggered Object Activation Daemon (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.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

The default value is `10`.

PollEndTime

The PollEndTime property specifies the time to stop polling the event queue. The format is *HH:MM*, where *HH* is 0 through 23 hours, and *MM* represents 0 through 59 minutes.

You must provide a valid value for this property. The default value is `HH:MM` without a value, and it must be changed.

If the adapter runtime detects:
* PollStartTime set and PollEndTime not set, or
* PollEndTime set and PollStartTime not set

it will poll using the value configured for the PollFrequency property.

PollFrequency

The PollFrequency property specifies the amount of time (in milliseconds) between the end of one polling action and the start of the next polling action. This is not the interval between polling actions. Rather, the logic is as follows:
* Poll to obtain the number of objects specified by the value of the PollQuantity property.
* Process these objects. For some connectors, this may be partly done on separate threads, which execute asynchronously to the next polling action.
* Delay for the interval specified by the PollFrequency property.
* Repeat the cycle.

The following values are valid for this property:
* The number of milliseconds between polling actions (a positive integer).
* The word `no`, which causes the connector not to poll. Enter the word in lowercase.
* The word `key`, which causes the connector to poll only when you type the letter `p` in the connector Command Prompt window. Enter the word in lowercase.

The default is `10000`.

Important: Some connectors have restrictions on the use of this property. Where they exist, these restrictions are documented in the chapter on installing and configuring the adapter.

PollQuantity

The PollQuantity property designates the number of items from the application that the connector polls for. If the adapter has a connector-specific property for setting the poll quantity, the value set in the connector-specific property overrides the standard property value.

This property is valid only if the value of the `DeliveryTransport` property is `JMS`, and the `ContainerManagedEvents` property has a value.

An e-mail message is also considered an event. The connector actions are as follows when it is polled for e-mail.
* When it is polled once, the connector detects the body of the message, which it reads as an attachment. Since no data handler was specified for this mime type, it will then ignore the message.

- The connector processes the first BO attachment. The data handler is available for this MIME type, so it sends the business object to Visual Test Connector.
- When it is polled for the second time, the connector processes the second BO attachment. The data handler is available for this MIME type, so it sends the business object to Visual Test Connector.
- Once it is accepted, the third BO attachment should be transmitted.

PollStartTime

The PollStartTime property specifies the time to start polling the event queue. The format is *HH:MM*, where *HH* is 0 through 23 hours, and *MM* represents 0 through 59 minutes.

You must provide a valid value for this property. The default value is HH:MM without a value, and it must be changed.

If the adapter runtime detects:
- PollStartTime set and PollEndTime not set, or
- PollEndTime set and PollStartTime not set

it will poll using the value configured for the PollFrequency property.

RepositoryDirectory

The RepositoryDirectory property is the location of the repository from which the connector reads the XML schema documents that store the metadata for business object definitions.

If the integration broker is ICS, this value must be set to 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 is set to *<ProductDir>*\repository by default. However, it may be set to any valid directory name.

RequestQueue

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

This property is valid only if the value of the DeliveryTransport property is JMS.

The default value is *<CONNECTORNAME>*/REQUESTQUEUE.

ResponseQueue

The ResponseQueue property specifies 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.

This property is valid only if the value of the DeliveryTransport property is JMS.

The default value is *<CONNECTORNAME>*/RESPONSEQUEUE.

RestartRetryCount

The RestartRetryCount property specifies the number of times the connector attempts to restart itself. When this property is used for a connector that is connected in parallel, it specifies the number of times the master connector application-specific component attempts to restart the client connector application-specific component.

The default value is 3.

RestartRetryInterval

The RestartRetryInterval property specifies the interval in minutes at which the connector attempts to restart itself. When this property is used for a connector that is linked in parallel, it specifies the interval at which the master connector application-specific component attempts to restart the client connector application-specific component.

Possible values for the property range from 1 through 2147483647.

The default value is 1.

ResultsSetEnabled

The ResultsSetEnabled property enables or disables results set support when Information Integrator is active. This property can be used only if the adapter supports DB2 Information Integrator.

This property is valid only if the value of the DeliveryTransport property is JMS, and the value of BrokerType is WMQI.

The default value is `false`.

ResultsSetSize

The ResultsSetSize property defines the maximum number of business objects that can be returned to Information Integrator. This property can be used only if the adapter supports DB2 Information Integrator.

This property is valid only if the value of the ResultsSetEnabled property is `true`.

The default value is `0`. This means that the size of the results set is unlimited.

RHF2MessageDomain

The RHF2MessageDomain property allows you to configure the value of the field domain name in the JMS header. When data is sent to a WebSphere message broker over JMS transport, the adapter framework writes JMS header information, with a domain name and a fixed value of `mrm`. A configurable domain name lets you track how the WebSphere message broker processes the message data.

This is an example header:
```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

This property is valid only if the value of BrokerType is WMQI or WAS. Also, it is valid only if the value of the DeliveryTransport property is JMS, and the value of the WireFormat property is `CwXML`.

Possible values are `mrm` and `xml`. The default value is `mrm`.

SourceQueue

The SourceQueue property 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 52.

This property is valid only if the value of DeliveryTransport is JMS, and a value for ContainerManagedEvents is specified.

The default value is `<CONNECTORNAME>/SOURCEQUEUE`.

SynchronousRequestQueue

The SynchronousRequestQueue property 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 synchronous request queue and waits for a response from the broker on the synchronous response queue. The response message sent to the connector has a correlation ID that matches the ID of the original message.

This property is valid only if the value of DeliveryTransport is JMS.

The default value is `<CONNECTORNAME>/SYNCHRONOUSREQUESTQUEUE`

SynchronousRequestTimeout

The SynchronousRequestTimeout property specifies the time in milliseconds that the connector waits for a response to a synchronous request. If the response is not received within the specified time, the connector moves the original synchronous request message (and error message) to the fault queue.

This property is valid only if the value of DeliveryTransport is JMS.

The default value is `0`.

SynchronousResponseQueue

The SynchronousResponseQueue property delivers response messages in reply to a synchronous request from the broker to the connector framework. This queue is necessary only if the connector uses synchronous execution.

This property is valid only if the value of DeliveryTransport is JMS.

The default is `<CONNECTORNAME>/SYNCHRONOUSRESPONSEQUEUE`

TivoliMonitorTransactionPerformance

The TivoliMonitorTransactionPerformance property specifies whether IBM Tivoli Monitoring for Transaction Performance (ITMTP) is invoked at run time.

The default value is `false`.

WireFormat

The WireFormat property specifies the message format on the transport:

- If the value of the RepositoryDirectory property is a local directory, the value is `CwXML.`
- If the value of the RepositoryDirectory property is a remote directory, the value is `CwBO`.

WsifSynchronousRequestTimeout

The WsifSynchronousRequestTimeout property specifies the time in milliseconds that the connector waits for a response to a synchronous request. If the response is not received within the specified time, the connector moves the original synchronous request message (and an error message) to the fault queue.

This property is valid only if the value of BrokerType is `WAS`.

The default value is 0.

XMLNameSpaceFormat

The XMLNameSpaceFormat property specifies short or long namespaces in the XML format of business object definitions.

This property is valid only if the value of BrokerType is set to WMQI or WAS.

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

The topics covered in this appendix are:

Overview of Connector Configurator

Connector Configurator allows you to configure the connector component of your adapter for use with these integration brokers:

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

If your adapter supports DB2 Information Integrator, use the WMQI options and the DB2 II standard properties (see the Notes column in the Standard Properties appendix.)

You use Connector Configurator to:

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

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see "Running Configurator in stand-alone mode" on page 66).

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

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

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

For **connector-specific properties**, however, you need first to define the properties and then set their values. You do this by creating a connector-specific property template for your particular adapter. There may already be a template set up in your system, in which case, you simply use that. If not, follow the steps in "Creating a new template" on page 67 to set up a new one.

Running connectors on UNIX

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.

Some properties in the Connector Configurator use directory paths, which default to the Windows convention for directory paths. If you use the configuration file in a UNIX environment, revise the directory paths to match the UNIX convention for these paths. Select the target operating system in the toolbar drop-list so that the correct operating system rules are used for extended validation.

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 without running System Manager and work with connector configuration files, irrespective of your broker.

To do so:
- From **Start>Programs**, click **IBM WebSphere Business Integration Adapters>IBM WebSphere Business Integration Toolset>Connector Configurator**.
- Select **File>New>Connector Configuration**.
- 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 72.)

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:

- 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.
- From Connector Configurator, select **File>Open**. Select the name of the connector configuration file from a project or from the directory in which it is stored.
- 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 connector definition as the template.

- To create a new template, see "Creating a new template" on page 67.
- To use an existing file, simply modify an existing template and save it under the new name. You can find existing templates in your \WebSphereAdapters\bin\Data\App directory.

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 in Connector Configurator:

1. Click **File>New>Connector-Specific Property Template**.
2. The **Connector-Specific Property Template** dialog box appears.
   - Enter a name for the new template in the **Name** field below **Input a New Template Name.** You will see this name again when you open the dialog box for creating a new configuration file from a template.

- 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 appears in the **Template Preview** display.

3. You can use an existing template whose property definitions are similar to those required by your connector as a starting point for your template. If you do not see any template that displays the connector-specific properties used by your connector, you will need to create one.

  - If you are planning to modify an existing template, select the name of the template from the list in the **Template Name** table below **Select the Existing Template to Modify: Find Template.**

  - This table displays the names of all currently available templates. You can also search for a template.

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
  Property Subtype
  Updated Method
  Description

- **Flags**
  Standard flags

- **Custom Flag**
  Flag

The **Property Subtype** can be selected when **Property Type** is a String. It is an optional value which provides syntax checking when you save the configuration file. The default is a blank space, and means that the property has not been subtyped.

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**, enter your values.

To create a new property value:

1. Right-click on the square to the left of the Value column heading.
2. From the pop-up menu, select **Add** to display the Property Value dialog box. Depending on the property type, the dialog box allows you to enter either a value, or both a value and a range.
3. Enter the new property value and click OK. The value appears in the **Value** panel on the right.

The **Value** panel 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 **Dependencies - 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.

Setting pathnames

Some general rules for setting pathnames are:
- The maximum length of a filename in Windows and UNIX is 255 characters.
- In Windows, the absolute pathname must follow the format [Drive:][Directory]\filename: for example, C:\WebSphereAdapters\bin\Data\Std\StdConnProps.xml In UNIX the first character should be /.

• Queue names may not have leading or embedded spaces.

Creating a new configuration file

When you create a new configuration file, you must name it and select an integration broker.

You also select an operating system for extended validation on the file. The toolbar has a droplist called **Target System** that allows you to select the target operating system for extended validation of the properties. The available options are: Windows, UNIX, Other (if not Windows or UNIX), and None-no extended validation (switches off extended validation). The default on startup is Windows.

To start Connector Configurator:
• In the System Manager window, select **Connector Configurator** from the **Tools** menu. Connector Configurator opens.
• In stand-alone mode, launch Connector Configurator.

To set the operating system for extended validation of the configuration file:
• Pull down the **Target System:** droplist on the menu bar.
• Select the operating system you are running on.

Then select **File>New>Connector Configuration**. In the New Connector window, enter the name of the new connector.

You also need 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 **Integration 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.

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. Set the operating system for extended validation of the configuration file using the **Target System:** droplist on the menu bar (see "Creating a new configuration file" above).
2. Click **File>New>Connector Configuration**.
3. 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**.

4. A configuration screen appears for the connector that you are configuring. The title bar shows the integration broker and connector name. You can fill in all the field values to complete the definition now, or you can save the file and complete the fields later.

5. 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.

6. 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 connector properties associated with the selected broker. The table shows **Property name**, **Value**, **Type**, **Subtype** (if the Type is a string), **Description**, and **Update Method**.
3. You can save the file now or complete the remaining configuration fields, as described in "Specifying supported business object definitions" on page 75..
4. 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 you created the configuration file, you used the **Target System** droplist that allows you to select the target operating system for extended validation of the properties.

   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.

   If you have elected to use the extended validation feature by selecting a value of Windows, UNIX or Other from the **Target System** droplist, the system will validate the property subtype s well as the type, and it displays a warning message if the validation fails.

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)
- Security

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 displayed for each property. It indicates whether a component or agent restart is necessary to activate changed values. You cannot configure this setting.

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.

   Note: If the property has a Type of String, it may have a subtype value in the Subtype column. This subtype is used for extended validation of the property.

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.

To get more information on a particular standard property, left-click the entry in the Description column for that property in the Standard Properties tabbed sheet. If you have Extended Help installed, an arrow button will appear on the right. When you click on the button, a Help window will open and display details of the standard property.

Note: If the hot button does not appear, no Extended Help was found for that property.

If installed, the Extended Help files are located in
<ProductDir>\bin\Data\Std\Help*<RegionalSetting>*\.

Setting connector-specific configuration properties

For connector-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.

   Note: If the property has a Type of String, you can select a subtype from the Subtype droplist. This subtype is used for extended validation of the property.

3. To encrypt a property, select the **Encrypt** box.

4. To get more information on a particular property, left-click the entry in the Description column for that property. If you have Extended Help installed, a hot button will appear. When you click on the hot button, a Help window will open and display details of the standard property.

   Note: If the hot button does not appear, no Extended Help was found for that property.

5. Choose to save or discard changes, as described for "Setting standard connector properties" on page 74.

If the Extended Help files are installed and the AdapterHelpName property is blank, Connector Configurator will point to the adapter-specific Extended Help files located in *<ProductDir>*\bin\Data\App\Help*<RegionalSetting>*\. Otherwise, Connector Configurator will point to the adapter-specific Extended Help files located in *<ProductDir>*\bin\Data\App\Help*<AdapterHelpName>**<RegionalSetting>*\. See the AdapterHelpName property described in the Standard Properties appendix.

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 Connector-specific Properties 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 Properties appendix, under "Configuration property values overview" 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 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 an ICL (Integration Component Library) 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 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 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)

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 Binding**

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** tab enables you to configure messaging properties. 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.

Validating messaging queues

Before you can validate a messaging queue, you must:

- Make sure that WebSphere MQ Series is installed.
- Create a messaging queue with channel and port on the host machine.
- Set up a connection to the host machine.

To validate the queue, use the Validate button to the right of the Messaging Type and Host Name fields on the Messaging tab.

Security (ICS)

You can use the **Security** tab in Connector Configurator to set various privacy levels for a message. You can only use this feature when the DeliveryTransport property is set to JMS.

By default, Privacy is turned off. Check the **Privacy** box to enable it.

The **Keystore Target System Absolute Pathname** is:
- For Windows:
  `<ProductDir>\connectors\security\<connectorname>.jks`
- For UNIX:
  `opt/IBM/WebSphereAdapters/connectors/security/<connectorname>.jks`

This path and file should be on the system where you plan to start the connector, that is, the target system.

You can use the Browse button at the right only if the target system is the one currently running. It is greyed out unless **Privacy** is enabled and the **Target System** in the menu bar is set to Windows.

The **Message Privacy Level** may be set as follows for the three messages categories (All Messages, All Administrative Messages, and All Business Object Messages):
- "" is the default; used when no privacy levels for a message category have been set.
- none
  Not the same as the default: use this to deliberately set a privacy level of none for a message category.
- integrity
- privacy
- integrity_plus_privacy

The **Key Maintenance** feature lets you generate, import and export public keys for the server and adapter.
- When you select **Generate Keys**, the Generate Keys dialog box appears with the defaults for the keytool that will generate the keys.
- The keystore value defaults to the value you entered in **Keystore Target System Absolute Pathname** on the Security tab.
- When you select OK, the entries are validated, the key certificate is generated and the output is sent to the Connector Configurator log window.

Before you can import a certificate into the adapter keystore, you must export it from the server keystore. When you select **Export Adapter Public Key**, the Export Adapter Public Key dialog box appears.
- The export certificate defaults to the same value as the keystore, except that the file extension is <filename>.cer.

When you select **Import Server Public Key**, the Import Server Public Key dialog box appears.
- The import certificate defaults to *<ProductDir>*\bin\ics.cer (if the file exists on the system).
- The import Certificate Association should be the server name. If a server is registered, you can select it from the droplist.

The **Adapter Access Control** feature is enabled only when the value of DeliveryTransport is IDL. By default, the adapter logs in with the guest identity. If the **Use guest identity** box is not checked, the **Adapter Identity** and **Adapter Password** fields are enabled.

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. By default, the file is saved to \WebSphereAdapters\bin\Data\App.
- You can also save it to a WebSphere Application Server project if you have set one up.

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 in the properties window will immediately change, to show only those tabs and fields that pertain to the new broker you have selected.

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

                <DefaultValue>en_US</DefaultValue>
        </ValidValues>
</Property>
```

Appendix C. Application Response Measurement

This adapter is compatible with the Application Response Measurement
application programming interface (API), an API that allows applications to be
managed for availability, service level agreements, and capacity planning. An
ARM-instrumented application can participate in IBM Tivoli Monitoring for
Transaction Performance, allowing collection and review of data concerning
transaction metrics.

Application Response Measurement instrumentation support

This adapter is compatible with the Application Response Measurement
application programming interface (API), an API that allows applications to be
managed for availability, service level agreements, and capacity planning. An
ARM-instrumented application can participate in IBM Tivoli Monitoring for
Transaction Performance, allowing collection and review of data concerning
transaction metrics.

Required software

In addition to the software prerequisites required for the adapter, you must have
the following installed for ARM to operate:

- WebSphere Application Server 5.0.1 (contains the IBM Tivoli Monitoring for
  Transaction Performance server). This does not have to be installed on the same
  system as the adapter.
- IBM Tivoli Monitoring for Transaction Performance v. 5.2 Fixpack 1. This must
  be installed on the same system on which the adapter is installed and
  configured to point to the system on which the IBM Tivoli Monitoring for
  Transaction Performance server resides.

Application Response Measurement support is available using any integration
broker supported with this release.

Note: Application Response Measurement instrumentation is supported on all
operating systems supported with this IBM WebSphere Business Integration
Adapters release *except* HP-UX (any version) and Red Hat Linux 3.0.

Enabling Application Response Measurement

ARM instrumentation is enabled via by setting the standard property
`TivoliMonitorTransactionPerformance` in Connector Configurator to "True." By
default ARM support is not enabled. (Refer to the "Standard Properties" appendix
of this document for more information.)

Transaction monitoring

When ARM is enabled, the transactions that are monitored are service events and
event deliveries. The transaction is measured from the start of a service request or
event delivery to the end of the service request or event delivery. The name of the
transaction displayed on the Tivoli Monitoring for Transaction Performance console
will start with either `SERVICE REQUEST` or `EVENT DELIVERY`. The next part of the
name will be the business object verb (such as `CREATE`, `RETRIEVE`, `UPDATE` or `DELETE`).
The final part of the name will be the business object name such as "EMPLOYEE."

For example, the name of a transaction for an event delivery for creation of an employee might be EVENT DELIVERY CREATE EMPLOYEE. Another might be SERVICE REQUEST UPDATE ORDER.

The following metrics are collected by default for each type of service request or event delivery:

- Minimum transaction time
- Maximum transaction time
- Average transaction time
- Total transaction runs

You (or the system administrator of the WebSphere Application Server) can select which of these metrics to display, for which adapter events, by configuring Discovery Policies and Listener Policies for particular transactions from within the Tivoli Monitoring for Transaction Performance console. (Refer to "For more information.")

For more information

Refer to the IBM Tivoli Monitoring for Transaction Performance documentation for more information. In particular, refer to the *IBM Tivoli Monitoring for Transaction Performance User's Guide* for information about monitoring and managing the metrics generated by the adapter.

Appendix D. Common Event Infrastructure

WebSphere Business Integration Server Foundation includes the Common Event Infrastructure Server Application, which is required for Common Event Infrastructure to operate. The WebSphere Application Server Foundation can be installed on any system (it does not have to be the same machine on which the adapter is installed.)

The WebSphere Application Server Application Client includes the libraries required for interaction between the adapter and the Common Event Infrastructure Server Application. You must install WebSphere Application Server Application Client on the same system on which you install the adapter. The adapter connects to the WebSphere Application Server (within the WebSphere Business Integration Server Foundation) by means of a configurable URL.

Common Event Infrastructure support is available using any integration broker supported with this release.

Required software

In addition to the software prerequisites required for the adapter, you must have the following installed for Common Event Infrastructure to operate:

*   WebSphere Business Integration Server Foundation 5.1.1
*   WebSphere Application Server Application Client 5.0.2, 5.1, or 5.1.1.

    (WebSphere Application Server Application Client 5.1.1 is provided with WebSphere Business Integration Server Foundation 5.1.1.)

Note: Common Event Infrastructure is not supported on any HP-UX or Linux platform.

Enabling Common Event Infrastructure

Common Event Infrastructure functionality is enabled with the standard properties `CommonEventInfrastructure` and `CommonEventInfrastructureContextURL`, configured with Connector Configurator. By default, Common Event Infrastructure is not enabled. The `CommonEventInfrastructureContextURL` property enables you to configure the URL of the Common Event Infrastructure server.(Refer to the "Standard Properties" appendix of this document for more information.)

Obtaining Common Event Infrastructure adapter events

If Common Event Infrastructure is enabled, the adapter generates Common Event Infrastructure events that map to the following adapter events:

*   Starting the adapter
*   Stopping the adapter
*   An application response to a timeout from the adapter agent
*   Any `doVerbFor` call issued from the adapter agent
*   A `gotApplEvent` call from the adapter agent

For another application (the "consumer application") to receive the Common Event Infrastructure events generated by the adapter, the application must use the

Common Event Infrastructure event catalog to determine the definitions of appropriate events and their properties. The events must be defined in the event catalog for the consumer application to be able to consume the sending application's events.

The "Common Event Infrastructure event catalog definitions" appendix of this document contains XML format metadata showing, for WebSphere Business Information adapters, the event descriptors and properties the consumer application should search for.

For more information

For more information about Common Event Infrastructure, refer to the Common Event Infrastructure information in the WebSphere Business Integration Server Foundation documentation, available at the following URL:

http://publib.boulder.ibm.com/infocenter/ws51help

For sample XML metadata showing the adapter-generated event descriptors and properties a consumer application should search for, refer to"Common Event Infrastructure event catalog definitions."

Common Event Infrastructure event catalog definitions

The Common Event Infrastructure event catalog contains event definitions that can be queried by other applications. The following are event definition samples, using XML metadata, for typical adapter events. If you are writing another application, your application can use event catalog interfaces to query against the event definition. For more information about event definitions and how to query them, refer to the Common Event Infrastructure documentation that is available from the online IBM WebSphere Server Foundation Information Center.

For WebSphere Business Integration adapters, the extended data elements that need to be defined in the event catalog are the keys of the business object. Each business object key requires an event definition. So for any given adapter, various events such as start adapter, stop adapter, timeout adapter, and any `doVerbFor` event (create, update, or delete, for example) must have a corresponding event definition in the event catalog.

The following sections contain examples of the XML metadata for start adapter, stop adapter, and event request or delivery.

XML format for "start adapter" metadata

```
<eventDefinition name="startADAPTER"
      parent="event">
   <property name ="creationTime" //Comment: example value would be
 "2004-05-13T17:00:16.319Z"
         required="true" />
   <property name="globalInstanceId" //Comment: Automatically generated
 by Common Event Infrastructure
         required="true"/>
   <property name="sequenceNumber"    //Comment: Source defined number
 for messages to be sent/sorted logically
         required="false"/>
   <property name="version"    //Comment: Version of the event
         required="false"
         defaultValue="1.0.1"/>
```

```
    <property name="sourceComponentId"
        path="sourceComponentId"
        required="true"/>
    <property name="application"    //Comment: The name#version of the
source application generating the event. Example is "SampleConnector#3.0.0"
        path="sourceComponentId/application"            required="false"/>
    <property name="component"    //Comment: This will be the name#version
 of the source component.
        path="sourceComponentId/component"
        required="true"
        defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"    //Comment: specifies the format
and meaning of the component
        path="sourceComponentId/componentIdType"
        required="true"
        defaultValue="Application"/>
    <property name="executionEnvironment"
 //Comment: Identifies the environment the application is running
 in...example is "Windows 2000#5.0"
        path="sourceComponentId/executionEnvironment"
        required="false" />
     <property name="location"     //Comment: The value of this is the
 server name...example is "WQMI"
        path="sourceComponentId/location"
        required="true"/>
    <property name="locationType"    //Comment specifies the format and
    meaning of the location
        path="sourceComponentId/locationType"
        required="true"
        defaultValue="Hostname"/>
     <property name="subComponent"       //Comment:further distinction
of the logical component
        path="sourceComponentId/subComponent"
        required="true"
        defaultValue="AppSide_Connector.AgentBusinessObjectManager"/>
     <property name="componentType"        //Comment: well-defined name
used to characterize all instances of this component
        path="sourceComponentId/componentType"
        required="true"
        defaultValue="ADAPTER"/>
     <property name="situation"    //Comment: Defines the type of
 situation that caused the event to be reported
        path="situation"
        required="true"/>
     <property name="categoryName="     //Comment: Specifies the type
of situation for the event
        path="situation/categoryName"
        required="true"
        defaultValue="StartSituation"/>
    <property name="situationType"     //Comment: Specifies the type
of situation and disposition of the event
        path="situation/situationType"
        required="true"
    <property name="reasoningScope" //Comment: Specifies the scope
 of the impact of the event
        path="situation/situationType/reasoningScope"
        required="true"
        permittedValue="INTERNAL"
        permittedValue="EXTERNAL"/>
    <property name="successDisposition" //Comment: Specifies the
 success of event
        path="situation/situationType/successDisposition"
        required="true"
        permittedValue="SUCCESSFUL"
        permittedValue="UNSUCCESSFUL" />
    <property name="situationQualifier"  //Comment: Specifies the
 situation qualifiers for this event
```

```
                              path="situation/situationType/situationQualifier"
                              required="true"
                              permittedValue="START_INITIATED"
                              permittedValue="RESTART_INITIATED"
                              permittedValue="START_COMPLETED" />
                </eventDefinition>
```

XML format for ″stop adapter″ metadata

The metadata for "stop adapter" is the same as that for "start adapter" with the
following exceptions:

* The default value for the categoryName property is StopSituation:

```
<property name="categoryName="
 //Comment: Specifies the type
 of situation for the event
                    path="situation/categoryName"
                    required="true"
                    defaultValue="StopSituation"/>
```

* The permitted values for the situationQualifier property differ and are as
  follows for "stop adapter":

```
<property name="situationQualifier"
 //Comment: Specifies the situation qualifiers for this event
            path="situation/situationType/situationQualifier"
            required="true"
            permittedValue="STOP_INITIATED"
            permittedValue="ABORT_INITIATED"
            permittedValue="PAUSE_INITIATED"
            permittedValue="STOP_COMPLETED"
    />
```

XML format for "timeout adapter" metadata

The metadata for "timeout adapter" is the same as that for "start adapter" and
"stop adapter" with the following exceptions:

* The default value for the categoryName property is ConnectSituation:

```
<property name="categoryName="
 //Comment: Specifies the type
 of situation for the event
                path="situation/categoryName"
                required="true"
                defaultValue="ConnectSituation"/>
```

* The permitted values for the situationQualifier property differ and are as
  follows for "timeout adapter":

```
<property name="situationQualifier"  //Comment: Specifies
 the situation qualifiers for this event
            path="situation/situationType/situationQualifier"
            required="true"
            permittedValue="IN_USE"
            permittedValue="FREED"
            permittedValue="CLOSED"
            permittedValue="AVAILABLE"
    />
```

XML format for ″request″ or ″delivery″ metadata

At the end of this XML format are the extended data elements. The extended data
elements for adapter request and delivery events represent data from the business
object being processed. This data includes the name of the business object, the key
(foreign or local) for the business object, and business objects that are children of
parent business objects. The children business objects are then broken down into
the same data as the parent (name, key, and any children business objects). This
data is represented in an extended data element of the event definition. This data
will change depending on which business object, which keys, and which child
business objects are being processed. The extended data in this event definition is
just an example and represents a business object named Employee with a key
EmployeeId and a child business object EmployeeAddress with a key EmployeeId.
This pattern could continue for as much data as exists for the particular business
object.

```
<eventDefinition name="createEmployee"      //Comment: This
 extension name is always the business object verb followed by the business
 object name
         parent="event">
    <property name ="creationTime"  //Comment: example value would be
"2004-05-13T17:00:16.319Z"
         required="true" />
    <property name="globalInstanceId" //Comment: Automatically generated
 by Common Event Infrastructure
         required="true"/>
    <property name="localInstanceId"     //Comment: Value is business
 object verb+business object name+#+app name+ business object identifier
         required="false"/>
    <property name="sequenceNumber"     //Comment: Source defined number
for messages to be sent/sorted logically
         required="false"/>
    <property name="version"  //Comment: Version of the event...value is
 set to 1.0.1
         required="false"
         defaultValue="1.0.1"/>
    <property name="sourceComponentId"
         path="sourceComponentId"
         required="true"/>
    <property name="application"     //Comment: The name#version of the
 source application generating the event...example is
"SampleConnector#3.0.0"
         path="sourceComponentId/application"
         required="false"/>
    <property name="component"   //Comment: This will be the name#version
of the source component.
         path="sourceComponentId/component"
         required="true"
         defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"      //Comment: specifies the format
 and meaning of the component
         path="sourceComponentId/componentIdType"
         required="true"
         defaultValue="Application"/>
    <property name="executionEnvironment" //Comment: Identifies the
 environment#version the app is running in...example is "Windows 2000#5.0"
         path="sourceComponentId/executionEnvironment"
         required="false" />
    <property name="instanceId" //Comment: Value is business object
  verb+business object name+#+app name+ business object identifier
         path="sourceComponentId/instanceId"
         required="false"
    <property name="location"    //Comment: The value of this is the
server name...example is "WQMI"
         path="sourceComponentId/location"
```

```
            required="true"/>
    <property name="locationType" //Comment specifies the format and
 meaning of the location
            path="sourceComponentId/locationType"
            required="true"
            defaultValue="Hostname"/>
    <property name="subComponent"  //Comment:further distinction of the
 logical component-in this case the value is the name of the business
 object
            path="sourceComponentId/subComponent"
            required="true"/>
    <property name="componentType"      //Comment: well-defined name used
 to characterize all instances of this component
            path="sourceComponentId/componentType"
            required="true"
            defaultValue="ADAPTER"/>
    <property name="situation" //Comment: Defines the type of
situation that caused the event to be reported
            path="situation"
            required="true"/>
   <property name="categoryName"     //Comment: Specifies the type
 of situation for the event
            path="situation/categoryName"
            required="true"
            permittedValue="CreateSituation"
            permittedValue="DestroySituation"
            permittedValue="OtherSituation" />
    <property name="situationType"     //Comment: Specifies the type
of situation and disposition of the event
            path="situation/situationType"
            required="true"
    <property name="reasoningScope" //Comment: Specifies the scope
of the impact of the event
            path="situation/situationType/reasoningScope"
            required="true"
            permittedValue="INTERNAL"
            permittedValue="EXTERNAL"/>
    <property name="successDisposition" //Comment: Specifies the
 success of event
            path="situation/situationType/successDisposition"
            required="true"
            permittedValue="SUCCESSFUL"
            permittedValue="UNSUCCESSFUL" />
   <extendedDataElements name="Employee" //Comment: name of business
 object itself
              type="noValue"
              <children name="EmployeeId"
                   type="string"/>  //Comment: type is one of the
 permitted values within Common Event Infrastructure documentation
              <children name="EmployeeAddress"
                   type="noValue"/>
                    <children name="EmployeeId"
                        type="string"/>
                  -
                  -
                  -
   </extendedDataElements>
</eventDefinition>
```

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other 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 grant 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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

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 the program(s) described in this publication at any time without notice. Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
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 measurements 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 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. All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples 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. COPYRIGHT LICENSE: This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. If you are viewing this information softcopy, the photographs and color illustrations may not appear.

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:

i5/OS
IBM
the IBM logo
AIX
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
IMS
Informix
iSeries
Lotus
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both. Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Linux is a trademark of Linus Torvalds in the United States, other countries, or both. Other company, product or service names may be trademarks or service marks of others.

WebSphere Business Integration Adapter Framework V2.6.0

Index

meta-objects *(continued)*
    child attributes *(continued)*
        UserName   35
    example   36
    top-level attributes   34
monitoring, of transactions   11, 83

P

poll location   4

R

recovery mechanism   7

S

standard configuration properties   19
Standard configuration properties for
  connectors   85

T

Tivoli Monitoring for Transaction
  Performance   11, 83
transaction monitoring   11, 83

U

Unix
    installed file structure   18

W

Windows
    installed file structure   17

IBM ®

Printed in USA