

IBM WebSphere Business Integration Adapters



Adapter for DTS Protocol User Guide

IBM WebSphere Business Integration Adapters



Adapter for DTS Protocol User Guide

Note!

Before using this information and the product it supports, read the information in "Notices" on page 69.

28 January 2005

This edition of this document applies to IBM WebSphere Business Integration Adapter for DTS Protocol, version 1.0.x.

To send us your comments about IBM documentation, 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.

© **Copyright International Business Machines Corporation 2005. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About This Document	v
What this document includes	v
What this document does not include	v
Audience	v
Related documents	v
Typographic conventions	vi

New in this release	vii
--------------------------------------	------------

Chapter 1. Overview of the Adapter	1
Adapter for DTS Protocol environment	1
Terminology	2
Characteristics of the DTS protocol	2
Solution architecture	3

Chapter 2. Installation and startup	7
Overview of Installation Tasks	7
Installing the connector and related files	7
Installed file structure	7
Running multiple instances of the adapter	8
Starting the connector	9
Stopping the connector	10

Chapter 3. Standard configuration properties for connectors	13
Configuring standard connector properties	13
Summary of standard properties	15
Standard configuration properties	20

Chapter 4. Business object requirements	35
Business object naming convention	35
Business object structure	35
Business object attributes	35

Chapter 5. Adapter for DTS protocol processing	37
Adapter processing	37
Event handling and request processing overview	37
Event handling data flow	38
Request processing data flow	38
Handling transparent data	39

Configuring the connector	39
-------------------------------------	----

Appendix A. Connector Configurator	43
Overview of Connector Configurator	43
Starting Connector Configurator	44
Running Configurator from System Manager	44
Creating a connector-specific property template	45
Creating a new configuration file	47
Using an existing file	48
Completing a configuration file	49
Setting the configuration file properties	50
Saving your configuration file	55
Changing a configuration file	56
Completing the configuration	56
Using Connector Configurator in a globalized environment	56

Appendix B. Common Event Infrastructure	59
Required software	59
Enabling Common Event Infrastructure	59
Obtaining Common Event Infrastructure adapter events	59
For more information	60
Common Event Infrastructure event catalog definitions	60
XML format for "start adapter" metadata	60
XML format for "stop adapter" metadata	62
XML format for "timeout adapter" metadata	62
XML format for "request" or "delivery" metadata	63

Appendix C. Application Response Management	65
Application Response Measurement instrumentation support	65

Index	67
------------------------	-----------

Notices	69
Programming interface information	70
Trademarks and service marks	71

About This Document

The IBM[®] WebSphere[®] Business Integration Adapter portfolio supplies integration connectivity for leading e-business technologies, enterprise applications, legacy applications, 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 DTS Protocol.

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 each 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. You should be familiar with DTS Protocol and WebSphere Business Integration system adapter development.

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.

This document may contain references to two other documents: the *System Installation Guide for Windows[®]* or *for UNIX[®]* and the *Implementation Guide for WebSphere InterChange Server*. If you choose to print this document, you may want to print these documents as well.

You can install documentation from the following sites:

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

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

- For using adapters with InterChange Server:

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

<http://www.ibm.com/websphere/integration/wbicollaborations/infocenter>

- For more information about message brokers (WebSphere MQ Integrator Broker, WebSphere MQ Integrator, and WebSphere Business Integration Message Broker):
<http://www.ibm.com/software/integration/mqfamily/library/manualsa/>.
- For more information about WebSphere Application Server:
<http://www.ibm.com/software/webservers/appserv/library.html>

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

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/>.

Typographic conventions

This document uses the following conventions:

<code>courier font</code>	Indicates a literal value, such as a command name, file name, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
<i>italic, italic</i>	Indicates a variable name or a cross-reference.
<i>blue text</i>	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, <code>option[,...]</code> means that you can enter multiple, comma-separated options.
< >	Angle brackets surround individual elements of a name to distinguish them from each other, as in <code><server_name><connector_name>tmp.log</code> .

New in this release

The purpose of this section is to describe the changes made in this document from previous releases. Because this is the first release, no previous releases exist.

Chapter 1. Overview of the Adapter

The WebSphere Business Integration Adapter for DTS (Distributed Transaction System) protocol, a widely used application-to-application inter-working protocol, provides connectivity between a message broker and applications running on Fujitsu VME systems. It does so by interacting with Fujitsu's HostTalk V2.0, which then uses the DTS protocol to communicate with a Fujitsu VME system.

The adapter supports one business object, and it supports only the update verb that is used for both requests and events. The adapter is a thin connector, which means that the business object it supports relates to the form of the host interactions, rather than any particular VME transaction-processing service that is to be performed.

This chapter provides an overview of the adapter, the protocol it supports, and its requirements in the following sections:

- "Adapter for DTS Protocol environment"
- "Terminology" on page 2
- "Characteristics of the DTS protocol" on page 2
- "Solution architecture" on page 3

Adapter for DTS Protocol environment

Before installing, configuring, and using the adapter, you must understand its environmental requirements, which are discussed in the following sections:

- "Broker compatibility"
- "Adapter platforms"
- "Software dependencies" on page 2

Broker compatibility

This adapter runs with the WebSphere Business Integration Adapter Framework V2.6 and requires one of the following:

- WebSphere InterChange Server V4.2.2, V4.3
- WebSphere MQ Integrator Broker V2.1
- WebSphere Business Integration Message Broker V5.0.1
- WebSphere Application Server Enterprise V5.0.2, in conjunction with WebSphere Studio Application Developer Integration Edition V5.0.1
- WebSphere Business Integration Server Foundation V5.1

See the Release Notes for any exceptions.

Adapter platforms

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

- Windows 2000 (Professional, Server, or Advanced Server) with Service Pack 4
- Solaris 8, 9
- AIX^R 5.2

Software dependencies

The following software dependencies exist:

- HostTalk VME Connector for Java platform, V2.0. Any platform that supports Sun J2SE V1.4 will support this software.
- IBM JDK, V1.4.2

Note: You must install the WebSphere Business Integration Adapter Framework V2.6 before you install the HostTalk VME Connector for Java platform, because the HostTalk VME Connector and the Adapter for DTS Protocol need to run on the same JRE. After you have installed the Adapter Framework, update the PATH element so that the the Adapter Framework Installed JRE: "<WBI Install Dir>/AdapterJRE/bin " appears first.

Terminology

The following terms are used in this Guide:

- **DTS (Distributed Transaction System)** DTS is an established and widely used application-to-application inter-networking protocol. It is supported on the following platforms: OpenVME, UNIX, and Windows.
- **Business Object** A set of attributes that represent a business entity (such as Customer) and an action on the data (such as a create or update operation). Components of the IBM WebSphere system use business objects to exchange information and trigger actions.
- **OpenVME** An operating system used on Fujitsu mainframes.
- **HostTalk** A portfolio of products and tools provided by Fujitsu. The portfolio enables applications to integrate data across a wide range of host environments.
- **HostTalk VME Connector for Java platform (VJC)** A product within the HostTalk portfolio that provides a client API for Java applications. It provides the ability to access OpenVME transaction-processing applications and incorporate them in solutions built upon Java technology.
- **TPMS (Transaction Processing Management System)** A component of the OpenVME operating system that manages services associated to legacy applications.
- **DTS Link** A logical connection from the Java platform to the OpenVME transaction processing management service that sends messages to a remote service on OpenVME.
- **Initiator** A DTS client that sends one or more request messages to a VME service.
- **Responder** A request processing agent that returns one or more reply messages to an initiator.
- **DTS Conversation** A sequence of request and reply interactions that share the same conversation identifier.
- **Message Tag** Routing information that enables DTS messages to be routed to the appropriate application component within VME.
- **DTS Server** Responsible for establishing connections or links to the transaction processing services.

Characteristics of the DTS protocol

The adapter for the DTS protocol conforms to the following communication characteristics of the protocol:

- DTS messaging is bi-directional, so a request may be sent from either VME or the message broker.
- Each DTS link handles either text messages or binary messages, not a combination of both.
- Each configured VJC service uses a particular DTS link, and thus each service handles either text messages or binary messages, not a combination of both.
- Each VJC service is associated with a single DTS link.
- A Java client establishes a connection, or session, to a particular VJC service. Thus, each connection handles either text messages or binary messages, but not a combination of both.
- All messages of a particular Message Tag or Message Type on a particular service must be either normal or transparent, not a combination of both.
- Each VJC service may have multiple concurrent connections.
- Each connection established can be used either as an initiator or as a responder, and each connection can handle only a single conversation at a time.
- A DTS reply must be sent back on the same VJC connection, VJC service, and DTS link on which the request was received. Because of this, the data type (text or binary) of each reply message must match that of the request, and if a connection terminates, any in-progress conversation becomes invalid.
- Each DTS message must not exceed a maximum size of approximately 6K.
- DTS requests must be contained in a single DTS message.
- DTS replies may have 0 or more intermediate reply messages.
- DTS Conversation IDs are unique only to a VJC service.
- If a transaction-processing application times out waiting for a response, the VJC and the adapter are not notified.

Solution architecture

Figure 1 on page 4 illustrates the data flow between the components that comprise the solution. The Adapter for DTS Protocol exchanges information with a message broker, and with the VJC Java API and the HostTalk VJC. These components then use the DTS protocol over TCP/IP to exchange data with applications that reside in the VME domain.

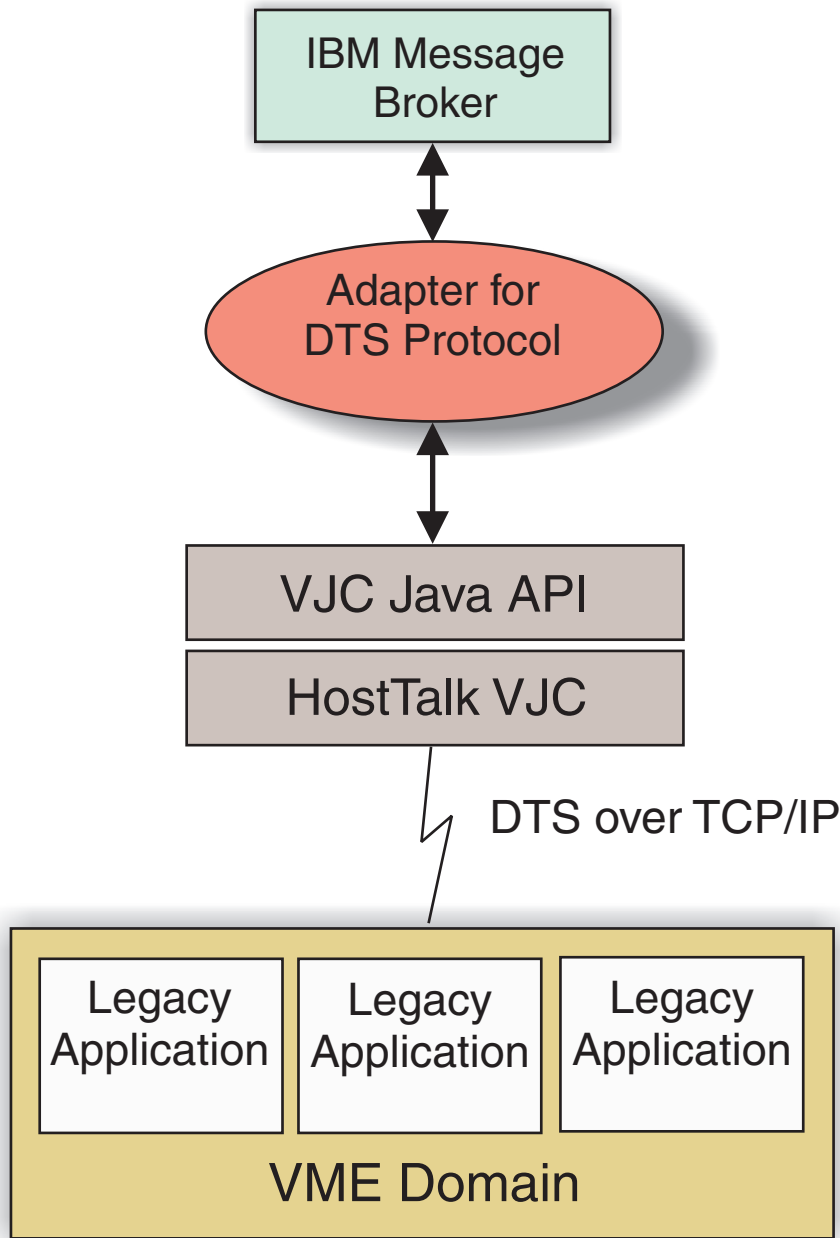


Figure 1. Adapter for DTS Protocol data flow

Installation, configuration, and design checklist

This section summarizes the tasks you must perform to install, configure, and design your DTS solution. Each section briefly describes the tasks and then provides links to sections in this document (and cross references to related documents) that describe how to perform the task or provide background information.

Installing the adapter

See Chapter 2, "Installation and startup," on page 7 for a description of what and where you must install.

Configuring connector properties

Connectors have two types of configuration properties: standard configuration properties and connector-specific configuration properties. Some of these properties have default values that you do not need to change. You may need to set the values of some of these properties before running the connector. For more information, see Chapter 5, “Adapter for DTS protocol processing,” on page 37.

Creating business objects

The Adapter for DTS Protocol supports only one business object type. The business object contains a DTS message as a data string within one of its parameters, but the composition and interpretation of the data is determined by endpoint applications. The business object definition, `BO_DTS_Call`, is delivered with the adapter product. Its structure is fixed and must not be changed, but you can create additional instances of it. See Chapter 4, “Business object requirements,” on page 35 for more information.

Chapter 2. Installation and startup

- “Overview of Installation Tasks”
- “Installing the connector and related files”
- “Running multiple instances of the adapter” on page 8
- “Starting the connector” on page 9
- “Stopping the connector” on page 10

This chapter describes how to install components for implementing the adapter for the DTS protocol. For information regarding installation of an InterChange Server system, see the *System Installation Guide* appropriate for your platform.

Overview of Installation Tasks

To install the connector for DTS Protocol, you must perform the following tasks:

Confirm the adapter requirements

Confirm that the adapter’s prerequisites and dependencies are met before you attempt to install it. To do this, review the information on broker compatibility, adapter framework, and software dependencies provided in the section “Adapter for DTS Protocol environment” on page 1.

Install the connector and related files

This task includes installing the files for the connector (and related components) from the software package onto your system. See “Installing the connector and related files.”

Installing the connector and related files

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

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

Installed file structure

The tables in this section show the installed file structure. Use this information to confirm that all files have been properly installed.

Windows connector file structure

The Installer copies the standard files associated with the connector into your system. All adapter files begin with the prefix BIA_.

The utility installs the connector and adds a shortcut for the connector agent to the Start menu.

Table 1 describes the Windows file structure used by the connector.

Table 1. Installed Windows file structure for the adapter

Subdirectory of <i>ProductDir</i>	Description of Contents
Connector\DTSCConnector	DTS Adapter JAR file and start scripts
Connector\messages	Adapter for DTS Protocol message file
Connector\bin\Data\App	Adapter for DTS Protocol configuration template
Connector\repository\DTS	Business object definitions

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

UNIX connector file structure

The Installer copies the standard files associated with the connector into your system. All adapter files begin with the prefix BIA_.

Table 2 describes the UNIX file structure used by the connector and shows the files that are automatically installed by the Installer.

Table 2. Installed UNIX file structure for the adapter

Subdirectory of <i>ProductDir</i>	Description of Contents
Connector/DTSCConnector	DTS Adapter JAR file and start scripts
Connector/messages	Adapter for DTS Protocol message file
Connector/bin/Data/App	Adapter for DTS Protocol configuration template
Connector/repository/DTS	Business object definitions

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

Running multiple instances of the adapter

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 these steps below:

- 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 have the following structure, where *connectorInstance* uniquely identifies the connector instance:

`ProductDir\connectors\connectorInstance`

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 the following directory and store the file in it:

`ProductDir\repository\connectorInstance`

Create additional business objects, if needed

The Adapter for DTS Protocol supports only one business object definition, whose structure is fixed and cannot be modified. The business object contains a DTS message within one of its parameters.

If your implementation will run multiple instances of the connector, you can create additional business objects with unique names so that each connector instance can subscribe to a single business object. However, you must make sure that any additional business objects conform to the structure of the supported business object definition. For more information, see “Business object structure” on page 35.

Files for the initial connector should be located 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 8.
3. Create a startup script shortcut (Windows only).
4. Copy the initial connector’s shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

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

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

Starting the connector

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

`ProductDir\connectors\connName`

where *connName* identifies the connector. The name of the startup script depends on the operating-system platform, as Table 3 shows.

Table 3. Startup scripts for a connector

Operating system	Startup script
UNIX-based systems	<code>connector_manager_connName</code>
Windows	<code>start_connName.bat</code>

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

- On Windows systems, from the **Start** menu
Select **Programs>IBM WebSphere Business Integration Adapters>Adapters>Connectors**. By default, the program name is “IBM WebSphere Business Integration Adapters”. However, it can be customized. Alternatively, you can create a desktop shortcut to your connector.
 - From the command line
 - On Windows systems:
`start_connName connName brokerName [-cconfigFile]`
 - On UNIX-based systems:
`connector_manager_connName -start`where *connName* is the name of the connector and *brokerName* identifies your integration broker, as follows:
 - For WebSphere InterChange Server, specify for *brokerName* the name of the ICS instance.
 - For WebSphere message brokers (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, or WebSphere Business Integration Message Broker) or WebSphere Application Server, specify for *brokerName* a string that identifies the broker.
- Note:** For a WebSphere message broker or WebSphere Application Server on a Windows system, you *must* include the `-c` option followed by the name of the connector configuration file. For ICS, the `-c` is optional.
- From Adapter Monitor (WebSphere Business Integration Adapters product only), which is launched when you start System Manager
You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
 - From System Monitor (WebSphere InterChange Server product only)
You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
 - On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector starts when the Windows system boots (for an Auto service) or when you start the service through the Windows Services window (for a Manual service).

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

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

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. Standard configuration properties for connectors

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

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

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

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

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

Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties, used by the framework
- Application, or connector-specific, configuration properties, used by the agent

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

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

Using Connector Configurator

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

Setting and updating property values

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

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

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 - how and when a change to the connector properties will take effect - depends on the individual property and on also its property type.

There are four different 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 working in stand-alone mode (independently of System Manager), for example with one of the WebSphere message brokers, you can only change properties through the configuration file. In this case, a dynamic update is not possible.
- **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 4 on page 15 below.

There are three types of location in which a standard property may reside. Some properties may reside in more than one place.

- **ReposController**
The property resides in the connector controller and is only effective there. If you change the value on the agent side, it will not affect the controller.
- **ReposAgent**
The property resides in the agent and is only effective there. A local configuration may be able to override this value, depending on the property.
- **LocalConfig**
The property resides in the configuration file for a particular connector and can only act through the configuration file. The controller cannot change the value of the property, and will not be aware of any changes made to the configuration file unless the system is redeployed to explicitly update the controller.

Any changes to a connector-specific property require an agent restart.

Summary of standard properties

Table 4 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker, as standard property dependencies are based on RepositoryDirectory.

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

Note: In the “Notes” column in Table 4, the phrase “Repository directory is REMOTE” indicates that the broker is the InterChange Server. When the broker is WMQI or WAS, the repository directory is set to LOCAL

Table 4. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdapterHelpName	Droplist consisting of valid subdirectories in <i>HomeDirectory\bin\Data\App\Help\RegionalSetting\</i>	Template name if valid or None	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	<i>CONNECTORNAME /ADMININQUEUE</i>	Component restart	Delivery Transport is JMS
AdminOutQueue	Valid JMS queue name	<i>CONNECTORNAME /ADMINOUTQUEUE</i>	Component restart	Delivery Transport is JMS
AgentConnections	1-4	1	Component restart	Delivery Transport is MQ or IDL: Repository directory is <REMOTE> (broker is ICS)
AgentTraceLevel	0-5	0	Dynamic	
ApplicationName	Application name	Value specified for the connector application name	Component restart	
BiDi.Application	Any valid combination of the BiDi attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (5 letters)	Component restart	BiDi.Transformation is true:

Table 4. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
BiDi.Broker	Any valid combination of the BiDi attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (5 letters)	Component restart	Applicable only if: BiDi.Transformation is true: BrokerType is ICS
BiDi.Metadata	Any valid combination of the BiDi attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (5 letters)	Component restart	BiDi.Transformation is true
BiDi.Transformation	true or false	false	Component restart	Applicable only if BrokerType is not WAS
BrokerType	ICS, WMQI, WAS		Component restart	
CharacterEncoding	Any supported code. The droplist shows this subset: ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437	ascii7	Component restart	
CommonEventInfrastructure	true or false	false	Component restart	
CommonEventInfrastructureContextURL	URL string. For example, corbaloc:iiop:host:2809	None	Component restart	CommonEventInfrastructure is true
ConcurrentEventTriggeredFlows	1 to 32,767	1	Component restart	Repository directory is <REMOTE> (broker is ICS)
ContainerManagedEvents	No value or JMS	No value	Component restart	Delivery Transport is JMS
ControllerEventSequencing	true or false	true	Dynamic	Repository directory is <REMOTE> (BrokerType is ICS)
ControllerStoreAndForwardMode	true or false	true	Dynamic	Repository directory is <REMOTE> (BrokerType is ICS)

Table 4. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
ControllerTraceLevel	0-5	0	Dynamic	Repository directory is <REMOTE> (broker is ICS)
DeliveryQueue		CONNECTORNAME/DELIVERYQUEUE	Component restart	JMS transport only
DeliveryTransport	MQ, IDL, or JMS	JMS	Component restart	If Repository directory is local, then value is JMS only
DuplicateEventElimination	true or false	false	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		CONNECTORNAME/FAULTQUEUE	Component restart	JMS transport only
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory or CxCommon.Messaging.jms.SonicMQFactory or any Java class name	CxCommon.Messaging.jms.IBMMQSeriesFactory	Component restart	JMS transport only
jms.ListenerConcurrency	1-32767	1	Component restart	jms.TransportOptimized is true
jms.MessageBrokerName	If FactoryClassName is IBM, use crossworlds.queue.manager. If FactoryClassName is Sonic, use localhost:2506.	crossworlds.queue.manager	Component restart	JMS transport only
jms.NumConcurrentRequests	Positive integer	10	Component restart	JMS transport only
jms.Password	Any valid password		Component restart	JMS transport only
jms.TransportOptimized	true or false	false	Component restart	Delivery Transport is JMS: BrokerType is ICS
jms.UserName	Any valid name		Component restart	JMS transport only
JvmMaxHeapSize	Heap size in megabytes	128m	Component restart	Repository directory is <REMOTE> (BrokerType is ICS)

Table 4. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
JvmMaxNativeStackSize	Size of stack in kilobytes	128k	Component restart	Repository directory is <REMOTE> (BrokerType is ICS)
JvmMinHeapSize	Heap size in megabytes	1m	Component restart	Repository directory is <REMOTE> (BrokerType is ICS)
ListenerConcurrency	1- 100	1	Component restart	Delivery Transport must be MQ
Locale	en_US, ja_JP, ko_KR, zh_CN, zh_TW, fr_FR, de_DE, it_IT, es_ES, pt_BR Note: This is a subset of the supported locales.	en_US	Component restart	
LogAtInterchangeEnd	true or false	false	Component restart	Repository Directory must be <REMOTE> (BrokerType is ICS)
MaxEventCapacity	1-2147483647	2147483647	Dynamic	Repository Directory must be <REMOTE> (BrokerType is ICS)
MessageFileName	Path or filename	CONNECTORNAMEConnector.txt	Component restart	
MonitorQueue	Any valid queue name	CONNECTORNAME/MONITORQUEUE	Component restart	JMS transport only: DuplicateEvent Elimination must be true
OADAutoRestartAgent	true or false	false	Dynamic	Repository Directory must be <REMOTE> (BrokerType is ICS)
OADMaxNumRetry	A positive number	1000	Dynamic	Repository Directory must be <REMOTE> (BrokerType is ICS)
OADRetryTimeInterval	A positive number in minutes	10	Dynamic	Repository Directory must be <REMOTE> (BrokerType is ICS)
PollEndTime	HH:MM	HH:MM	Component restart	

Table 4. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
PollFrequency	A positive integer in milliseconds no (to disable polling) key (to poll only when the letter p is entered in the connector's Command Prompt window)	10000	Dynamic	
PollQuantity	1-500	1	Agent restart	JMS transport only: Container Managed Events is specified
PollStartTime	HH:MM(HH is 0-23, MM is 0-59)	HH:MM	Component restart	
RepositoryDirectory	Location of metadata repository		Agent restart	For ICS: set to <REMOTE> For WebSphere MQ message brokers and WAS: set to C:\crossworlds\repository
RequestQueue	Valid JMS queue name	CONNECTORNAME/REQUESTQUEUE	Component restart	Delivery Transport is JMS
ResponseQueue	Valid JMS queue name	CONNECTORNAME/RESPONSEQUEUE	Component restart	Delivery Transport is JMS: required only if Repository directory is <REMOTE>
RestartRetryCount	0-99	3	Dynamic	
RestartRetryInterval	A sensible positive value in minutes: 1 - 2147483547	1	Dynamic	
ResultsSetEnabled	true or false	false	Component restart	Only if Delivery Transport is JMS and BrokerType is WMQI
ResultsSetSize	Positive integer	0	Component restart	Only if Results SetEnabled is true. A value of zero means the results set size is unbounded.
RHF2MessageDomain	mrm, xml	mrm	Component restart	Only if Delivery Transport is JMS and WireFormat is CwXML

Table 4. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
SourceQueue	Valid WebSphere MQ name	CONNECTORNAME/SOURCEQUEUE	Agent restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		CONNECTORNAME/ SYNCHRONOUSREQUESTQUEUE	Component restart	Delivery Transport is JMS
SynchronousRequestTimeout	0 - any number (milliseconds)	0	Component restart	Delivery Transport is JMS
SynchronousResponseQueue		CONNECTORNAME/ SYNCHRONOUSRESPONSEQUEUE	Component restart	Delivery Transport is JMS
TivoliMonitorTransaction Performance	true or false	false	Component restart	
WireFormat	CwXML, CwBO	CwXML	Agent restart	CwXML if Repository Directory is not <REMOTE>; CwBO if Repository Directory is <REMOTE>
WsifSynchronousRequestTimeout	0 - any number (milliseconds)	0	Component restart	WAS only
XMLNamespaceFormat	short, long	short	Agent restart	WebSphere MQ message brokers and WAS only

Standard configuration properties

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

AdapterHelpName

Used to find an extended help file for a connector-specific property, if one exists.

The default value is the template name if valid, or else *none*.

AdminInQueue

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

The default value is CONNECTORNAME/ADMININQUEUE.

AdminOutQueue

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

The default value is CONNECTORNAME/ADMINOUTQUEUE.

AgentConnections

Applicable only if RepositoryDirectory is <REMOTE>.

The AgentConnections property controls the number of ORB (Object Request Broker) connections opened by orb.init[].

The default value of this property is set to 1. You can change it as required.

AgentTraceLevel

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

ApplicationName

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

BiDi.Application

Applicable only if BiDi.Transformation is true.

This property specifies whether the bi-directional format for data is coming from an external application into the adapter in the form of a business object supported by this adapter. It defines the bi-directional 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)

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

BiDi.Broker

Applicable only if BiDi.Transformation is true.

This property specifies whether the bi-directional format for data is sent to or received from the integration broker. It defines the bi-directional attributes of the broker data. These attributes are as listed in BiDi.application above.

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

BiDi.Metadata

Applicable only if BiDi.Transformation is true.

This property defines the bi-directional format or attributes for the metadata. These attributes are as listed in BiDi.application above.

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

BiDi.Transformation

This property defines whether the system should perform a bi-directional transformation at runtime or not.

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

The default value is `false`.

BrokerType

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

CharacterEncoding

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

Note: Java-based connectors do not use this property. A C++ connector currently uses the value `asci i7` for this property.

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

CommonEventInfrastructure

Defines whether the Common Event Infrastructure (CEI) should be invoked at runtime.

The default value is `false`.

CommonEventInfrastructureContextURL

Applicable only if `CommonEventInfrastructure` is set to `true`. It defines the URL used to gain access to the Common Event Infrastructure (CEI) server application.

The default value is `none`.

ConcurrentEventTriggeredFlows

Applicable only if `RepositoryDirectory` is `<REMOTE>`.

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

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

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

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

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

ContainerManagedEvents

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

There is no default value.

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

- PollQuantity = 1 to 500
- SourceQueue = /SOURCEQUEUE

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

These properties are adapter-specific, but **example** values are:

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

The fields for these values in the Data Handler tab will be displayed only if you have set ContainerManagedEvents to JMS.

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

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

ControllerEventSequencing

Applicable only if RepositoryDirectory is <REMOTE> (BrokerType is ICS).

This property enables event sequencing in the connector controller. The default is true.

ControllerStoreAndForwardMode

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

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

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

The default is true.

ControllerTraceLevel

Applicable only if RepositoryDirectory is <REMOTE>.

This property sets the level of trace messages for the connector controller. The default is 0.

DeliveryQueue

Applicable only if DeliveryTransport is JMS.

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

The default value is CONNECTORNAME/DELIVERYQUEUE.

DeliveryTransport

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

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

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

WebSphere MQ and IDL

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

- Asynchronous communication:
WebSphere MQ allows the application-specific component to poll and persistently store events even when the server is not available.
- Server side performance:
WebSphere MQ provides faster performance on the server side. In optimized mode, WebSphere MQ stores only the pointer to an event in the repository database, while the actual event remains in the WebSphere MQ queue. This saves having to write potentially large events to the repository database.

- Agent side performance:
WebSphere MQ provides faster performance on the application-specific component side. Using WebSphere MQ, the connector's polling thread picks up an event, places it in the connector's queue, then picks up the next event. This is faster than IDL, which requires the connector's polling thread to pick up an event, go over the network into the server process, store the event persistently in the repository database, then pick up the next event.

JMS

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

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

This property can also be set to `false`.

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

FaultQueue

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

The default value is `CONNECTORNAME/FAULTQUEUE`.

JvmMaxHeapSize

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

The default value is 128m.

JvmMaxNativeStackSize

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

The default value is 128k.

JvmMinHeapSize

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

The default value is 1m.

jms.FactoryClassName

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

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

jms.ListenerConcurrency

Applicable only if `jms.OptimizedTransport` is true.

The property specifies the number of concurrent listeners for the JMS controller.

The default is 1.

jms.MessageBrokerName

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

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

When you connect to a remote message broker, this property takes the following (mandatory) values:

`QueueMgrName:<Channel>:<HostName>:<PortNumber>`,

where the variables are:

`QueueMgrName`: The name of the queue manager.

`Channel`: The channel used by the client.

`HostName`: The name of the machine where the queue manager is to reside.

`PortNumber`: The port number to be used by the queue manager for listening.

For example:

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

jms.NumConcurrentRequests

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

The default value is 10.

jms.Password

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

There is no default.

jms.TransportOptimized

Applicable only if `DeliveryTransport` is JMS and `BrokerType` is ICS.

The property determines whether the WIP will be optimized. You need an MQ provider if you want to optimize the WIP.

The default value is false.

jms.UserName

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

There is no default.

ListenerConcurrency

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

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

Locale

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

A locale name has the following format:

ll_TT.codeset

where:

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

By default, only a subset of supported locales appears in the drop-down list. To add other supported values to the drop-down list, you must manually modify the `\Data\Std\stdConnProps.xml` file in the product directory. For more information, refer to the sections on Connector Configurator in this guide.

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

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

LogAtInterchangeEnd

Applicable only if `RepositoryDirectory` is `<REMOTE>`.

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

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

MaxEventCapacity

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

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

MessageFileName

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

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

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

MonitorQueue

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

The default value is `CONNECTORNAME/MONITORQUEUE`

OADAutoRestartAgent

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

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

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

The default value is false.

OADMaxNumRetry

Valid only when the RepositoryDirectory is <REMOTE>.

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

The default value is 1000.

OADRetryTimeInterval

Valid only when the RepositoryDirectory is <REMOTE>.

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

The default is 10.

PollEndTime

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

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

PollFrequency

This is the interval between the end of the last poll and the start of the next poll. PollFrequency 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 PollQuantity.
- Process these objects. For some adapters, this may be partly done on separate threads, which execute asynchronously to the next polling action.
- Delay for the interval specified by PollFrequency.
- Repeat the cycle.

Set PollFrequency to one of the following values:

- The number of milliseconds between polling actions (an integer).

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

The default is 10000.

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

PollQuantity

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

An email message is also considered an event. The connector behaves as follows when it is polled for email.

- When it is polled once, the connector goes to pick the body of the message, as it is also considered an attachment also. Since no data handler was specified for this mime type, it will ignore the body.
- The connector processes the first PO attachment. The DH is available for this mime type so it sends the business object to the Visual Test Connector.
- When it is polled for the second time, the connector processes the second PO attachment. The DH is available for this mime type so it sends the business object to the Visual Test Connector.
- Once it is accepted, the third PO attachment should come through.

PollStartTime

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

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

RequestQueue

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

The default value is `CONNECTOR/REQUESTQUEUE`.

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

The default is 3.

RestartRetryInterval

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

The default is 1.

ResultsSetEnabled

Applicable only if `DeliveryTransport` is set to JMS and `BrokerType` is WMQI.

The property provides a means of enabling or disabling results set support when the Information Integrator is active.

The default is `false`.

ResultsSetSize

Applicable only if `ResultsSetEnabled` is set to `true`.

The property defines the maximum number of business objects that can be returned to the Information Integrator.

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

RHF2MessageDomain

WebSphere message brokers and WAS only.

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

A sample header would look like this:

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>  
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

The default value is `mrm`, but it may also be set to `xml`. This property only appears when `DeliveryTransport` is set to `JMS` and `WireFormat` is set to `CwXML`.

SourceQueue

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

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see “`ContainerManagedEvents`” on page 23.

The default value is `CONNECTOR/SOURCEQUEUE`.

SynchronousRequestQueue

Applicable only if `DeliveryTransport` is `JMS`.

Delivers request messages that require a synchronous response from the connector framework to the broker. This queue is necessary only if the connector uses synchronous execution. With synchronous execution, the connector framework sends a message to the `SynchronousRequestQueue` and waits for a response back from the broker on the `SynchronousResponseQueue`. The response message sent to the connector bears a correlation ID that matches the ID of the original message.

The default is `CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE`

SynchronousResponseQueue

Applicable only if `DeliveryTransport` is `JMS`.

Delivers response messages sent in reply to a synchronous request from the broker to the connector framework. This queue is necessary only if the connector uses synchronous execution.

The default is `CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE`

SynchronousRequestTimeout

Applicable only if `DeliveryTransport` is `JMS`.

Specifies the time in minutes that the connector waits for a response to a synchronous request. If the response is not received within the specified time, then the connector moves the original synchronous request message into the fault queue along with an error message.

The default value is `0`.

TivoliMonitorTransactionPerformance

This property specifies whether IBM Tivoli Monitoring for Transaction Performance (ITMTP) is invoked at runtime.

The default value is `false`.

WireFormat

Message format on the transport.

- If the `RepositoryDirectory` is a local directory, the setting is `CwXML`.

- If the value of RepositoryDirectory is <REMOTE>, the setting is CwBO.

WsifSynchronousRequestTimeout

WAS integration broker only.

Specifies the time in minutes that the connector waits for a response to a synchronous request. If the response is not received within the specified, time then the connector moves the original synchronous request message into the fault queue along with an error message.

The default value is 0.

XMLNameSpaceFormat

WebSphere message brokers and WAS integration broker only.

A strong property that allows the user to specify short and long name spaces in the XML format of business object definitions.

The default value is short.

Chapter 4. Business object requirements

- “Business object naming convention”
- “Business object structure”
- “Business object attributes”

The Adapter for DTS Protocol supports only one business object definition. The business object contains a DTS message as a data string within one of its parameters, but the composition and interpretation of the data is determined by endpoint applications. The same business object is used for both requests and events. This chapter describes the structure, requirements, and attributes of connector business objects.

Business object naming convention

The business object definition BO_DTS_Call is delivered with the adapter product. The business object structure is fixed and must not be changed, but the business object name can be changed. This allows you to run multiple instances of the adapter with the same broker. For example, one adapter instance can subscribe to a business object named "example_one" while another adapter instance subscribes to a business object named "example_two," as long as both business objects have the structure of BO_DTS_Call.

Business object structure

The following table describes the structure and attributes of the business object.

Table 5. Attributes of BO_DTS_Call business object

Attribute Name	Type	Key	Required	Max Length	Default
ServiceName	String	Yes	Yes	16	None
MessageDataFormat	String	No	Yes	4	Text
MessageType	String	No	If using Template: No	12	None
MessageData	Long Text	No	No		None
ConversationID	String	No	On Events: Yes	4	None
ReplyMessageTag	String	No	No	12	None
Template	String	No	If using Message Type: No	12	None

Business object attributes

This section provides more information about the attributes of BO_DTS_Call.

ServiceName

The service name. This name must be between one and 16 characters in length, and it must be the name of a VJC service that is configured to connect to an OpenVME TPMS system.

MessageDataFormat

The data format of the message, either plain text or HEX. If the message is formatted as HEX data, this data must be converted to a string before it can be sent via VJC. Valid values are either Text or HEX.

MessageType

The type of the message, equivalent to the MessageTag property of the DTS message.

MessageData

The actual message that is being sent.

ConversationID

An identifier that specifies an ongoing conversation. This attribute is used to correlate a business object that represents a reply to an event with an existing DTS connection.

ReplyMessageTag

This is the ReplyMessageTag of the DTS message.

Template

This is the Template of the DTS message.

Chapter 5. Adapter for DTS protocol processing

- “Adapter processing” on page 37
- “Event handling and request processing overview” on page 37
- “Event handling data flow” on page 38
- “Request processing data flow” on page 38
- “Handling transparent data” on page 39
- “Configuring the connector” on page 39

This chapter describes how the adapter acts as both a DTS initiator and responder, and it describes how to configure the connector.

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

For more information about the relationship of the integration broker to the connector, see the *System Administration Guide*.

Adapter processing

The Adapter for DTS Protocol interacts with the following external interfaces:

- A message broker, via the WebSphere Business Integration Adapter Framework
- The VJC, which enables connections to VME systems using the DTS protocol

The adapter processes requests from the broker. The adapter acts as a DTS initiator and sends requests from the broker to VJC. The reply from VJC is put into the request business object before the broker resumes control of the transaction. The adapter also processes requests that are sent to the adapter from VJC. The adapter acts as a DTS responder when it processes requests from VJC. Request messages are put into a business object and sent to the broker. Replies to these requests are received by the adapter from the broker in the form of system request calls. Business objects that represent reply messages are distinguished from those that represent a new request by value of the **conversationID** field, in the business object. The conversationID correlates requests and responses for a single DTS conversation. The adapter supports concurrent responder conversations.

Conversations that are in progress when a connection terminates, or when the adapter is restarted, become invalid. Replies received by the adapter after a connection is terminated will not be processed, and an exception message will be generated to indicate this to the broker.

Event handling and request processing overview

The adapter supports event handling (notification) from a VJC service as well as request processing from the message broker and associated responses. These supported functions are represented by:

- Request/response pairs initiated from the integration broker
- Event/response pairs initiated from the VJC service

During system call request processing, a business object received from the broker can represent either a DTS request or a DTS response, depending on the value of the **ConversationID** field in the business object. If it has a ConversationID, it is a response to a DTS request. During event processing, the adapter receives messages that represent DTS requests from the VJC. The maximum size of data that can be sent in a single DTS message is approximately 6KB. This maximum is defined by the DTS protocol, and it is set in the `MaxMessageDataSize` connector configuration property and can be changed to reflect DTS requirements. The request part of a message-pair must be sent in a single message, but replies may consist of more than one message.

Event handling data flow

When the adapter acts as a DTS responder, data flow occurs as follows:

1. The request message is received by the adapter.
2. A business object is constructed by the adapter and sent to the broker.
3. A reply is sent back to the adapter.
4. The reply is sent to VME via VJC as one or more messages.

On initialization, the adapter checks for the value set for the configuration property `ResponderConnectionsPerService`, and it opens that number of responder connections per configured service. When event polling occurs, the adapter requests a message on each connection where a message may be waiting.

The DTS protocol requires that the size of the request from the VJC is only one message. If the request consists of more than one message, the request is discarded. Reply messages, received from the broker, must be contained in a single business object. If the reply message in the reply business object is greater than the size specified in the `MaxMessageDataSize` configuration property, the adapter will break it into appropriate size messages and each message will be sent to the VJC. The connector configuration property `ReplyTimeout` determines how long to keep a VJC connection for a particular conversation open while waiting for a response. When this time has elapsed, the conversation is terminated and the connection is made available. If a response is received after termination, it is discarded and an exception is thrown. Additionally, the adapter will take the value specified in the `ReplyMessageTag` attribute of the business object, and set the `MessageTag` of the DTS message to that value before sending the message to the VJC.

Request processing data flow

When the adapter acts as a DTS initiator, data flow occurs as follows:

1. The Request business object is received by the adapter.
2. A request message is constructed by the adapter and sent to the VME via the VJC.
3. Zero or more intermediate replies may be sent back to the adapter.
4. The final reply is sent to the adapter.
5. The reply is saved to a business object and returned to the integration broker.

If the message in the business object exceeds the size specified in the connector configuration property `MaxMessageDataSize`, it will not be processed and the adapter will throw an exception. The adapter accepts replies from the VJC that consist of multiple messages, or intermediate reply messages followed by the final reply message. If the application-specific configuration property `DiscardIntermediateReplies` is set to `false`, intermediate reply messages in the

conversation are concatenated to be sent back in the final reply. The default value of `DiscardIntermediateReplies` is `true`, which means that intermediate reply messages are discarded, and only the final reply message is sent to the broker in the business object.

Handling transparent data

The `MessageData` attribute in the business object is a string value. When the adapter receives business objects from the broker that represent either DTS requests or responses, the `MessageData` attribute may contain transparent data (binary data) or text data. If it contains binary data, the string must be converted to a byte array before sending it to VJC. The business object attribute, `MessageDataFormat` determines whether the data is binary or text. During event handling, when the adapter receives a message from the VJC representing a DTS request, the data may also be binary or text. If it is binary, the message data must be converted from a byte array to a string. For event handling the connector configuration properties define whether a particular DTS service will send and receive binary or text data. When converting a byte array containing the hexadecimal `0x07AF` (2 bytes) the result would be a string containing the characters `"07AF"`. Conversely, when converting a string containing the characters `"07AF"`, the result is a byte array containing the hexadecimal `0x07AF` (2 bytes).

Note: Byte data contained in the business object may be in either lowercase or uppercase form when it is sent to the adapter, but is only in lowercase form when it is sent from the adapter.

Configuring the connector

After using the Installer to install the connector files to your system, you must set the standard and application-specific connector configuration properties.

Setting configuration properties

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

Connector-specific configuration properties

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

Table 6 lists the connector-specific configuration properties. See the sections that follow for explanations of the properties. Note that some of the properties contain other properties. The + character indicates the entry's position in the property hierarchy.

Table 6. Connector-specific configuration properties

Name	Possible values	Default value	Required
Version	Version number of the adapter	1	Yes
PoolSize	The maximum number of simultaneous connections held open by the Connection Pool	5	Yes
MaxMessageDataSize	The maximum allowable size, in bytes, of message data	6000	Yes

Table 6. Connector-specific configuration properties (continued)

Name	Possible values	Default value	Required
ReaperSleepTime	Time between checks for connections	100	Yes
MaxConnectionAge	Passage of time before an unused connection is terminated	300000	Yes
DiscardIntermediateReplies	Determines intermediate reply handling	True	Yes
+ Services	Hierarchical property.	N/A	No
- ServiceX	Child property	None; entries only exist for configured services.	No
-- <EventTypeX>	Child property	N/A	Yes, at least 1
-- <TransparentData>	Child property; indicates data format of messages	False	Yes
ReplyTimeout	Indicates maximum DTS connection wait time	0	No
ResponderConnectionsPerService	Specifies the maximum number of concurrent responder connections that are handled	1	Yes

Version: Version number of the adapter. This version is reported by the adapter as part of its initialization messages. Do not change this value.

PoolSize: The maximum number of simultaneous connections held open by the Connection Pool for use in System Call Request Processing. Set this value to a multiple of the number of services to connect to and the number of concurrent conversations expected.

MaxMessageDataSize: The maximum allowable size, in bytes, of message data. Set this value to be less than or equal to the maximum message size allowed by the DTS protocol, which is 6000 bytes. Values less than 1 are invalid. For string messages, the minimum is 2 bytes, since the Java character is 2 bytes.

ReaperSleepTime: The time, in milliseconds, that the reaper sleeps between checking the connections in the connection pool.

MaxConnectionAge: The maximum age, in milliseconds, an unused connection can reach before it is terminated by the connection reaper.

DiscardIntermediateReplies: If set to true, intermediate VME replies in the conversation will be discarded. Otherwise, if false, intermediate VME replies in the conversation will be concatenated and returned in the final reply.

+ **Services:** This is a hierarchical property.

- **ServiceX:** This is a child of the Services property, and it holds the name of the service to subscribe to as a responder. X indicates an incremental number, and there is no limit to the number of services that can exist under the Services property.

-- **<EventTypeX>:** This is a child of the ServiceX property, and it holds a message type value of messages that the adapter will receive during polling from the particular service. A message type consists of 1-12 alphanumeric characters, and it is normally allocated by the transaction-processing service manager.

X indicates an incremental number, and there is no limit to the number of EventType elements that can exist.

-- **<TransparentData>**: This is a child of the ServiceX property. If this is set to true, all messages to and from a given service will be handled transparently (using binary data). If this property is set to false, message data is in text format.

ReplyTimeout: The maximum time in milliseconds that a DTS connection will be held open while waiting for a reply business object from the broker. If the value is zero, or if the property is not set, the connection will be held indefinitely.

ResponderConnectionsPerService: Specifies the maximum number of concurrent responder connections that the adapter handles. If all connections are in use, additional messages are rejected. Because additional connections claim processing resources, set the value of this parameter to the fewest number of concurrent connections sufficient for your implementation of the adapter.

Appendix A. Connector Configurator

This appendix describes how to use Connector Configurator to set configuration property values for your adapter.

You use Connector Configurator to:

- Create a connector-specific property template for configuring your connector
- Create a configuration file
- Set properties in a configuration file

Note:

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

The topics covered in this appendix are:

- “Overview of Connector Configurator” on page 43
- “Starting Connector Configurator” on page 44
- “Creating a connector-specific property template” on page 45
- “Creating a new configuration file” on page 47
- “Setting the configuration file properties” on page 50
- “Using Connector Configurator in a globalized environment” on page 56

Overview of Connector Configurator

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

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

You use Connector Configurator to:

- Create a **connector-specific property template** for configuring your connector.
- Create a **connector configuration file**; you must create one configuration file for each connector you install.
- Set properties in a configuration file.

You may need to modify the default values that are set for properties in the connector templates. You must also designate supported business object definitions and, with ICS, maps for use with collaborations as well as specify messaging, logging and tracing, and data handler parameters, as required.

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see “Running Configurator in stand-alone mode” on page 44).

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 45 to set up a new one.

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

Starting Connector Configurator

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

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

Running Configurator in stand-alone mode

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

To do so:

- From **Start>Programs**, click **IBM WebSphere InterChange Server>IBM WebSphere Business Integration Tools>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 49.)

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 45.
- 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
Updated Method
Description
- **Flags**
Standard flags
- **Custom Flag**
Flag

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

Specifying values

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

1. Click the **Value** tab. The display panel for Value replaces the display panel for General.
2. Select the name of the property in the **Edit properties** display.
3. In the fields for **Max Length** and **Max Multiple Values**, enter your values.

To create a new property value:

1. Select the property in the **Edit properties** list and right-click on it.
2. From the dialog box, select **Add**.
3. Enter the name of 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.

Creating a new configuration file

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

- In the System Manager window, right-click on the **Connectors** folder and select **Create New Connector**. Connector Configurator opens and displays the **New Connector** dialog box.
- In stand-alone mode: from Connector Configurator, 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. Click **File>New>Connector Configuration**.
2. The **New Connector** dialog box appears, with the following fields:
 - **Name**

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

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

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

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

Select the template you want to use and click **OK**.
3. A configuration screen appears for the connector that you are configuring. The title bar shows the integration broker and connector 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.
4. To save the file, click **File>Save>To File** or **File>Save>To Project**. To save to a project, System Manager must be running.

If you save as a file, the **Save File Connector** dialog box appears. Choose *.cfg as the file type, verify in the File Name field that the name is spelled correctly and has the correct case, navigate to the directory where you want to locate the file, and click **Save**. The status display in the message panel of Connector Configurator indicates that the configuration file was successfully created.

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

Using an existing file

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

- A connector definition file.

This is a text file that lists properties and applicable default values for a specific connector. Some connectors include such a file in a `\repository` directory in their delivery package (the file typically has the extension `.txt`; for example, `CN_XML.txt` for the XML connector).
- An ICS repository file.

Definitions used in a previous ICS implementation of the connector may be available to you in a repository file that was used in the configuration of that connector. Such a file typically has the extension `.in` or `.out`.
- A previous configuration file for the connector.

Such a file typically has the extension `*.cfg`.

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

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

Follow these steps to open a *.txt, *.cfg, or *.in file from a directory:

1. In Connector Configurator, click **File>Open>From File**.
2. In the **Open File Connector** dialog box, select one of the following file types to see the available files:
 - Configuration (*.cfg)
 - ICS Repository (*.in, *.out)
Choose this option if a repository file was used to configure the connector in an ICS environment. A repository file may include multiple connector definitions, all of which will appear when you open the file.
 - All files (*.*)
Choose this option if a *.txt file was delivered in the adapter package for the connector, or if a definition file is available under another extension.
3. In the directory display, navigate to the appropriate connector definition file, select it, and click **Open**.

Follow these steps to open a connector configuration from a System Manager project:

1. Start System Manager. A configuration can be opened from or saved to System Manager only if System Manager has been started.
2. Start Connector Configurator.
3. Click **File>Open>From Project**.

Completing a configuration file

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

The title of the configuration screen displays the integration broker and connector name as specified in the file. Make sure you have the correct broker. If not, change the broker value before you configure the connector. To do so:

1. Under the **Standard Properties** tab, select the value field for the BrokerType property. In the drop-down menu, select the value ICS, WMQI, or WAS.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 52..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

Setting the configuration file properties

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

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

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

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

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

- Associated Maps
- Resources
- Messaging (where applicable)

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

Standard properties differ from connector-specific properties as follows:

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

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

- A field with a grey background indicates a standard property. You can change the value but cannot change the name or remove the property.

- A field with a white background indicates an application-specific property. These properties vary according to the specific needs of the application or connector. You can change the value and delete these properties.
- Value fields are configurable.
- The **Update Method** field is 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.
3. After entering all the values for the standard properties, you can do one of the following:
 - To discard the changes, preserve the original values, and exit Connector Configurator, click **File>Exit** (or close the window), and click **No** when prompted to save changes.
 - To enter values for other categories in Connector Configurator, select the tab for the category. The values you enter for **Standard Properties** (or any other category) are retained when you move to the next category. When you close the window, you are prompted to either save or discard the values that you entered in all the categories as a whole.
 - To save the revised values, click **File>Exit** (or close the window) and click **Yes** when prompted to save changes. Alternatively, click **Save>To File** from either the File menu or the toolbar.

Setting application-specific configuration properties

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

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

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

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

Encryption for connector properties

Application-specific properties can be encrypted by selecting the **Encrypt** check box in the 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 configuration properties for connectors* appendix, under “Setting and updating property values” on page 14.

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 only)

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

The list of business objects contains the application-specific business object which the agent supports and the corresponding generic object that the controller sends

to the subscribing collaboration. The association of a map determines which map will be used to transform the application-specific business object to the generic business object or the generic business object to the application-specific business object.

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

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

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

- **Business Object Name**

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

- **Associated Maps**

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

- **Explicit**

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

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

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

To explicitly bind a map:

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

Resources (ICS)

The **Resource** tab allows you to set a value that determines whether and to what extent the connector agent will handle multiple processes concurrently, using connector agent parallelism.

Not all connectors support this feature. If you are running a connector agent that was designed in Java to be multi-threaded, you are advised not to use this feature, since it is usually more efficient to use multiple threads than multiple processes.

Messaging (ICS)

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

Setting trace/log file values

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

To change the logging and tracing values:

1. Click the **Trace/Log Files** tab.
2. For either logging or tracing, you can choose to write messages to one or both of the following:

- To console (STDOUT):
Writes logging or tracing messages to the STDOUT display.

Note: You can only use the STDOUT option from the **Trace/Log Files** tab for connectors running on the Windows platform.

- To File:
Writes logging or tracing messages to a file that you specify. To specify the file, click the directory button (ellipsis), navigate to the preferred location, provide a file name, and click **Save**. Logging or tracing message are written to the file and location that you specify.

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

Data handlers

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

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

Saving your configuration file

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

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

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

- In stand-alone mode, as a file with a *.cfg extension in a directory folder. 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 on the properties screen will immediately change, to show only those tabs and fields that pertain to the new broker you have selected.

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

Appendix B. 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 `gotAppEvent` 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>

```

Appendix C. Application Response Management

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.

Index

Special characters

-- 40
- ServiceX 40
+ Services 40

A

Adapter environment 1
adapter platforms 1
AIX 1
Application Response Measurement
instrumentation, support for 65

B

BO_DTS_Call 35
Broker compatibility 1
business object 2
business object, definition 35
business object, structure 35

C

Common Event Infrastructure
event catalog 60
metadata 60
ConversationID 36

D

DiscardIntermediateReplies 40
Distributed Transaction System 2
DTS 2
DTS Conversation 2
DTS Link 2
DTS protocol characteristics 2
DTS Server 2

E

environment, for adapter 1
event catalog, for Common Event
Infrastructure 60
Event handling 37

H

HostTalk 2
HostTalk VME Connector for Java
platform 2

I

IBM Tivoli Monitoring for Transaction
Performance 65
Initiator 2

M

MaxConnectionAge 40
MaxMessageDataSize 40
Message Tag 2
MessageData 36
MessageDataFormat 36
MessageType 36
monitoring, of transactions 65

O

OpenVME 2

P

PoolSize 40

R

ReaperSleepTime 40
ReplyMessageTag 36
ReplyTimeout 41
Request processing 37
required operating systems 1
Responder 2
ResponderConnectionsPerService 41

S

ServiceName 36
Solaris 1

T

Template 36
Tivoli Monitoring for Transaction
Performance 65
TPMS 2
transaction monitoring 65
Transaction Processing Management
System 2
transparent data 39

V

Version 40

W

WebSphere Application Server
Enterprise 1
WebSphere Business Integration Message
Broker 1
WebSphere InterChange Server 1
WebSphere MQ Integrator Broker 1
Windows 2000 1

Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

COPYRIGHT LICENSE

This information may contain sample application programs in source language, which illustrates 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.

Programming interface information

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

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

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

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

Trademarks and service marks

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

IBM
the IBM logo
AIX
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.



IBM WebSphere Business Integration Adapter Framework, V2.6.0



Printed in USA