

IBM WebSphere Business Integration Adapters



Adapter for Lotus Domino User Guide

Adapter Version 1.1.x

IBM WebSphere Business Integration Adapters



Adapter for Lotus Domino User Guide

Adapter Version 1.1.x

Note!

Before using this information and the product it supports, read the information in "Notices" on page 67.

19December2003

This edition of this document applies to IBM Business Integration Adapter for Lotus Domino version 1.1.x, and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

© Copyright International Business Machines Corporation 2003. 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
Audience	v
Related documents	v
Typographic conventions	v
Summary of Changes	vii
New for Adapter for Lotus Domino 1.1.0	vii
Chapter 1. Overview of the adapter	1
Usage scenarios	1
Adapter architecture	1
How the connector works	3
Adapter environment	5
Chapter 2. Installing and configuring the adapter	9
Overview of the installation process	9
Installing the adapter and related files	9
Installed file structure	9
Modifying the Domino server	10
Configuring the connector	15
Starting the connector	18
Stopping the connector	19
Creating multiple connector instances	19
Chapter 3. Lotus Domino adapter business objects	21
Defining metadata	21
Overview of business object structure	21
Lotus Domino business object definition	22
Generating business objects	24
Chapter 4. Troubleshooting and error handling	25
Event status	25
Error messages	25
Tracing	29
Appendix A. Standard configuration properties for connectors	31
New and deleted properties	31
Configuring standard connector properties	31
Summary of standard properties	32
Standard configuration properties	36
Appendix B. Connector Configurator	47
Overview of Connector Configurator	47
Starting Connector Configurator	48
Running Configurator from System Manager	49
Creating a connector-specific property template	49
Creating a new configuration file	51
Using an existing file	52
Completing a configuration file	53
Setting the configuration file properties	54
Saving your configuration file	59
Changing a configuration file	60
Completing the configuration	60
Using Connector Configurator in a globalized environment	60

Appendix C. Lotus Domino business object attributes	63
DominoDocument business object attributes	63
DominoItem business object attributes	64
DominoItemValue business object attributes	64
Index	65
Notices	67
Programming interface information	68
Trademarks and service marks	68

About this document

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

This document describes the installation, configuration, business object definitions, and troubleshooting for the IBM WebSphere Business Integration Adapter for Lotus Domino.

Audience

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

Related documents

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

You can install related documentation from the following sites:

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

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

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

- For using adapters with InterChange Server:

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

These sites contain simple directions for downloading, installing and viewing the documentation. To read the documentation, you need an HTML browser such as Netscape Navigator or Internet Explorer, or Adobe Acrobat Reader 4.0.5 or higher. For the latest version of Adobe Acrobat Reader for your platform, go to the Adobe website (<http://www.adobe.com>).

Typographic conventions

This document uses the following conventions:

`courier font`

Indicates a literal value, such as a command name, file name, information that you type, or information that the system prints on the screen.

italic, italic

Indicates a new term the first time that it appears, a variable name, or a cross-reference.

blue outline

A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference.

{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
	In a syntax line, a pipe separates a set of options from which you must choose one and only one.
[]	In a syntax line, square brackets surround an optional parameter.
...	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, <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> .
<i>ProductDir</i>	Represents the directory where the product is installed.
/, \	In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes. All product pathnames are relative to the directory where the connector for Lotus Domino is installed on your system.

Summary of Changes

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

New for Adapter for Lotus Domino 1.1.0

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

- Configuration for Domino server has been simplified.
- Explanation for error 4001 has changed.
- Contains updated standard configuration properties in Appendix A.
- Contains updated Connector Configurator section in Appendix B.
- Contains improved instructions for Starting and Stopping the adapter. Contains new information about starting multiple adapters.
- Adapter installation information has been moved from this guide. See Chapter 2 for the new location of that information.

Chapter 1. Overview of the adapter

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

The adapter for Lotus Domino consists of an adapter and the adapter framework. The adapter framework is the software that IBM provides to configure and run an adapter. Together, the adapter and adapter framework enable a Lotus Domino server to exchange information with an integration broker such as the WebSphere Application Server, the WebSphere Business Integration Message Broker (or any supported WebSphere message broker) or the WebSphere InterChange Server (ICS).

The adapter for Lotus Domino enables integration with Lotus Domino Server versions R5 (5.0.3 and above) and R6 (6.0.2 and below). It is not compatible with earlier versions of Lotus Domino. Each Domino server requires its own instance of the Lotus Domino adapter. However, a single adapter instance can support multiple databases on the same server.

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

This chapter contains the following sections:

- “Usage scenarios” on page 1
- “Adapter architecture” on page 1
- “How the connector works” on page 3
- “Adapter environment” on page 5

Usage scenarios

The Lotus Domino adapter can be used to provide communication to other enterprise applications about Lotus Domino documents that are created on a Domino server. For example, if a purchase order document is created in a Domino database, this is detected and relevant data is delivered via the adapter to the appropriate enterprise application for processing.

The adapter can also be used to communicate information to the Lotus Domino database about data that is generated or changed in other enterprise applications. For example, if some data about a company employee is updated on an internal employee database, the adapter could deliver a notification to the employee through a Lotus Note, which is a Domino document. Or, a report created in another application could be recreated as a Domino document.

Adapter architecture

The following adapter components provide communication from the integration broker to and from the Lotus Domino server:

- A connector, which enables bi-directional data exchange between a Lotus Domino server and an integration broker. The connector includes an application-specific component and the connector framework.
- An event table, which stores information about events that occur on the Domino server.

- An event listener, which detects functions (such as creation, deletion, and updating) that occur on the Domino server and records them in the event table.

For the Lotus Domino adapter, these and all other adapter components are provided for you. Installation and configuration are the only tasks necessary to set up the adapter.

Connector architecture

The connector consists of the following components that enable communication between Lotus Domino and the integration broker:

- An application-specific component, which polls the event table and if a desired event is detected, creates a business object to send to the integration broker. It also accepts incoming business objects from the integration broker.
- The connector framework. The connector framework acts as an intermediary between the integration broker and the application-specific component.

Application-specific connector component

The application-specific component of the connector performs the following tasks:

- Detects application events
- Archives application events
- Builds application-specific business objects
- Processes integration broker requests
- Logs error and trace messages

A business object handler is part of the application-specific component. It processes business objects sent to the adapter from the integration broker.

Connector framework

The connector framework manages interactions between a connector's application-specific component and the integration broker. It sends and receives business objects and manages the exchange of startup and administrative messages. It also retrieves the metadata that the connector requires from the repository.

Event table

The event table consists of three separate sets of data, or views: the configuration view, the event view, and the archive view. The **configuration view** stores the selections you make, during adapter configuration, about which types of Domino server events the listener should detect. The **event view** stores the information about actual server events. The **archive view** contains status information about events that have already been processed.

Event listener

The event listener is a Domino Extension Manager executable library that you will install as an extension to the Domino server. It detects events that occur on the Domino server and stores information about those that are of interest in the event table. As part of adapter setup, you will configure what event types are of interest; that is, those which you wish to pass to the integration broker (and eventually to another application). The listener will then detect any Domino server events of the designated type and record them in the event table. You must add the name of the event listener to the server's NOTES.INI file so that when the server is started, the event listener will also start. (Refer to Chapter 2, "Installing and configuring the adapter," on page 9 for more information about modifying the NOTES.INI file.)

How the connector works

The application-specific component of the connector polls the event table event view for events. If it finds an applicable event (sometimes called a subscribed-to event), it retrieves information about the event from the Domino server and builds an application-specific business object. The connector populates the business object with data obtained with an API call to the Domino server. It then sends this business object through the connector framework to the integration broker.

For incoming business objects (business objects sent to the adapter from the integration broker), the connector receives a business object from the integration broker, processes that business object based on its active verb, and then sends a request for operation to Lotus Domino.

The following sections explain how events that originate on the Lotus Domino server are passed to the broker, and how requests from another application are forwarded to the Lotus Domino server.

Event processing

Events that occur on the Lotus Domino server are passed to the integration broker as follows:

1. At startup, the event listener reads the event table configuration view to obtain the name of the database and determine which events should be monitored.
2. The event listener detects events from the database and filters them according to information in the event table configuration view. If the event is of a type listed in the event table configuration view, the information about the event is retained.
3. The event listener stores information (key data) about the event in the event table event view.
4. The connector (application-specific component) polls the event table event view to obtain key data about the event.
5. The application-specific component of the connector uses the event key data to retrieve all of the relevant event-related data from the Domino server.
6. The application-specific component of the connector creates a business object and sends it to the connector framework, which then sends it to the integration broker.
7. If there is another application requesting Lotus Domino event information from the broker, the broker passes the event information to the other application. (Details of this process vary depending upon which integration broker is being used. Refer to your integration broker documentation for more information.)

Figure 1 shows how an event is processed by the Lotus Domino adapter.

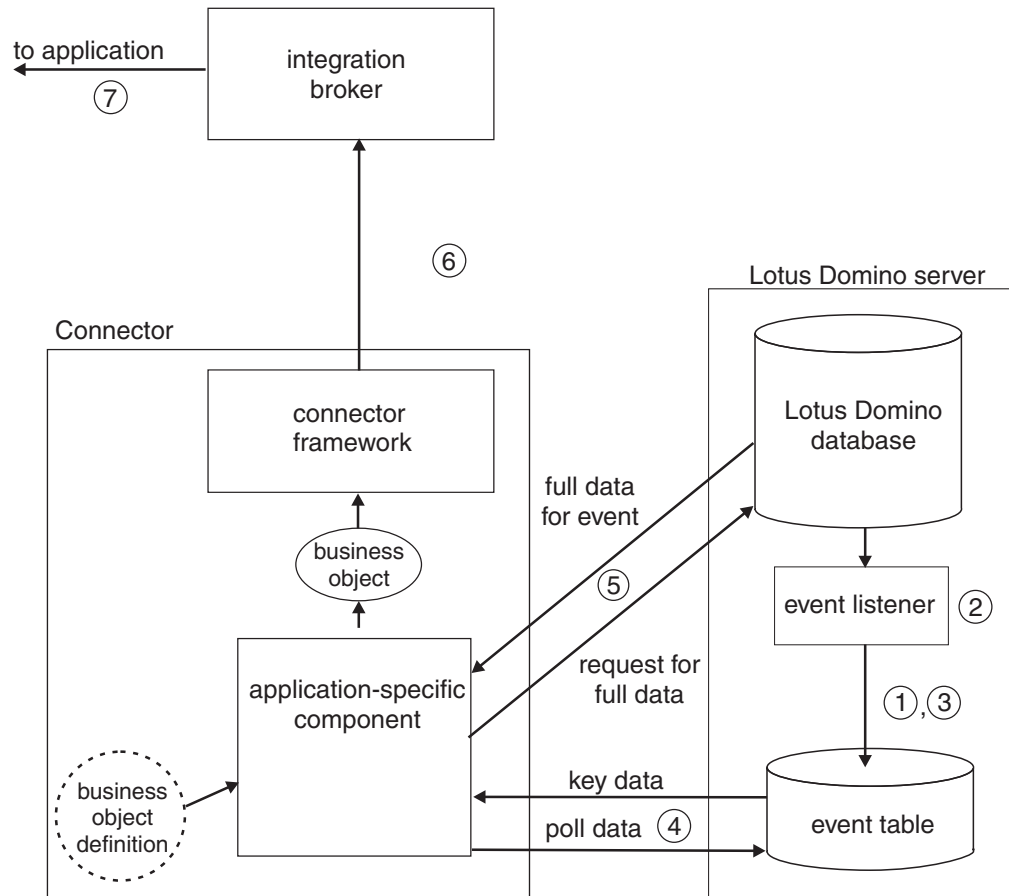


Figure 1. Event processing for Lotus Domino adapter

Request processing

For events originating from the integration broker:

1. The broker sends a business object to the connector.
2. The connector establishes a session with the Domino server.
3. An action occurs on the Lotus Domino server corresponding to the request passed to it by the broker. (For example, if the request is to update a field in a Notes document, the value is updated in the Domino database.)
4. The adapter sends the status of the requested action (whether successful or whether a problem occurred) to the integration broker, which sends it back to the requesting application.

Figure 2 shows how a request is processed by the Lotus Domino adapter.

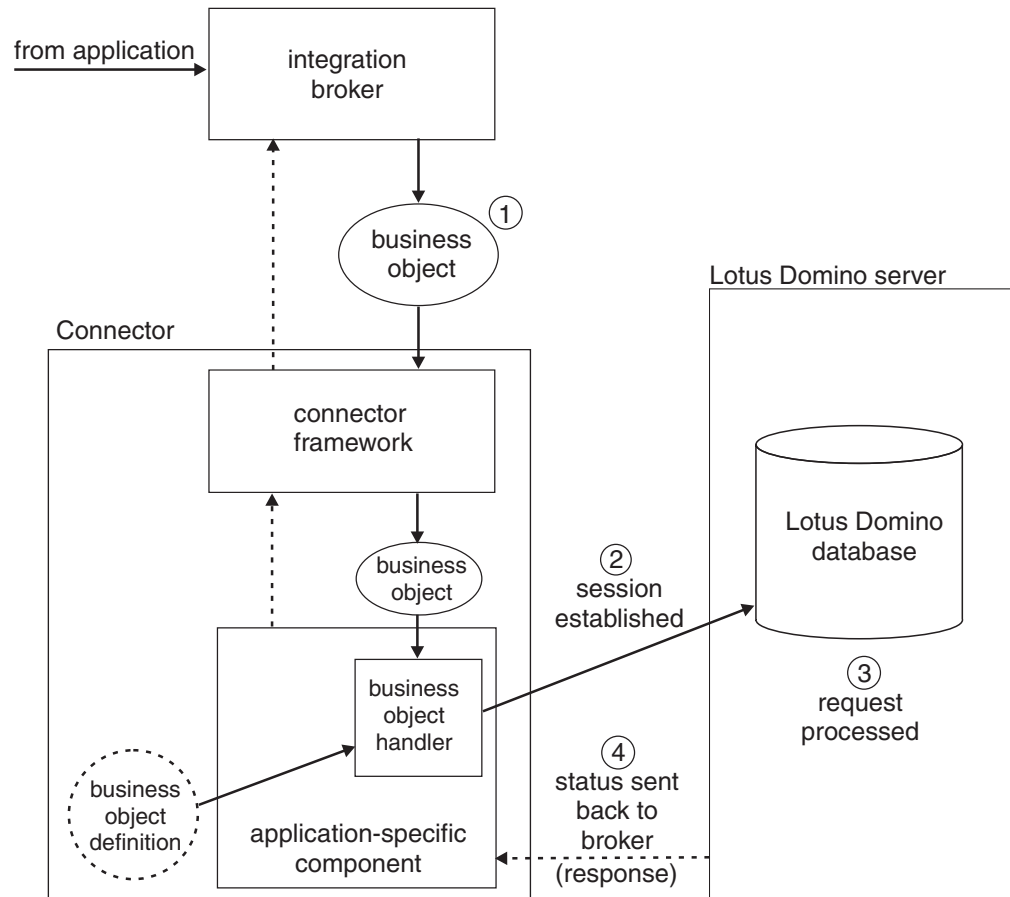


Figure 2. Request processing for Lotus Domino adapter

Adapter environment

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

- “Broker compatibility”
- “Software prerequisites” on page 6
- “Processing locale-dependent data” on page 7

Broker compatibility

The adapter framework that an adapter uses must be compatible with the version of the integration broker (or brokers) with which the adapter is communicating. The 1.1.0 version of the adapter for Lotus Domino is supported on the following adapter framework and integration brokers:

Adapter framework: WebSphere Business Integration Adapter Framework, versions 2.1, 2.2, 2.3.x, and 2.4.

Integration brokers:

- WebSphere InterChange Server, version 4.11, 4.2.x.
- WebSphere MQ Integrator Broker, version 2.1.0
- WebSphere MQ Integrator, version 2.1.0

- WebSphere Business Integration Message Broker version 5.0 with CSD02
- WebSphere Application Server Enterprise Edition, version 5.0.2, with WebSphere Studio Application Developer Integration Edition, version 5.0.1.

See the Release Notes for any exceptions.

Software prerequisites

In addition to the adapter framework and one of the integration brokers listed in “Broker compatibility” on page 5, the following software must be installed before you install the Lotus Domino adapter.

Operating system:

One of the following operating systems must be installed on the system on which the adapter will be installed:

- Windows: Windows 2000 with Service Pack 2 or 3 or later

Note: Beginning with the 1.1 version, the adapter for Lotus Domino is not supported on Microsoft Windows NT.

- UNIX: AIX 5.1 or later, Solaris 7.0 or later

Note: The adapter installer can be installed on Windows XP and will install Windows XP tools support; however, the adapter cannot run on a Windows XP host.

Application software

- Domino Server versions R5 (5.0.3 and above) and R6 (6.02 and below)

Additional software

- Java 2 Runtime Environment (JRE) Standard Edition version 1.3.1 or later.

Note: If the adapter will be installed on the same computer as the integration broker and your integration broker is WebSphere Interchange Server, the JRE already should have been installed as part of the broker installation. If your integration broker is a WebSphere message broker or WebSphere Application Server, you can install the JRE from the WebSphere Business Integration Adapters installer. Refer to the *WebSphere Business Integration Adapters Installation Guide* for more information.

For instructions on installing your integration broker and its prerequisites, see the following documentation:

- For WebSphere InterChange Server (ICS), see the *System Installation Guide for UNIX or for Windows*.
- For message brokers (WebSphere MQ Integrator Broker, WebSphere MQ Integrator, or WebSphere Business Integration Message Broker), see *Implementing Adapters with WebSphere Message Brokers* and the installation documentation for your particular broker. Some of this can be found at the following web site: <http://www-3.ibm.com/software/integration/mqfamily/library/manualsa/>
- For WebSphere Application Server (WAS), see *Implementing Adapters with WebSphere Application Server* and the documentation at <http://www.ibm.com/software/webservers/apserv/library.html>.

For information on installing Domino server, refer to the installation information provided with your Lotus Domino software.

Processing locale-dependent data

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

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

Chapter 2. Installing and configuring the adapter

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

- “Overview of the installation process” on page 9
- “Installing the adapter and related files”
- “Installed file structure” on page 9
- “Modifying the Domino server” on page 10
- “Configuring the connector” on page 15
- “Starting the connector” on page 18
- “Stopping the connector” on page 19
- “Creating multiple connector instances” on page 19

Note: Refer to “Adapter environment” on page 5 for pre-requisite software requirements.

Overview of the installation process

Follow these steps to enable the adapter for Lotus Domino. Details are provided in the remainder of this chapter.

1. Install and configure the integration broker. The broker can be installed on the same computer or a different computer than the one on which you will install Lotus Domino and the adapter.
2. Install and configure Lotus Domino, if not already installed.
3. Install Java 2 Runtime Environment on the same system on which you will install the adapter.
4. Install the adapter for Lotus Domino.
5. Modify the Domino server. This involves copying files, making some configuration changes, changing the NOTES.INI file, and configuring the event table.
6. Make the NCSO.jar file accessible to the adapter.
7. Configure the adapter.
8. Start the adapter.

Installing the adapter and related files

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

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

Installed file structure

The adapter files will be copied to the directories shown in Table 1 on page 10. In Table 7, *ProductDir* represents the name of a higher level-directory path that you selected or verified during installation. The default path will vary depending upon your operating system and integration broker.

Table 1. Adapter directory structure and files

Directory	File (s)	Description
<i>ProductDir</i> /connectors/Domino	BIADominoConnector.jar	Needed by the adapter for service call handling and event processing.
	start_Domino.sh	Startup script for UNIX systems
	start_Domino.bat	Startup script for Windows systems
<i>ProductDir</i> /connectors/Domino/dependencies	BIALD_EventTable.nsf	Event table
	BIALD_emptytrash.gif BIALD_movetotrash.gif BIALD_newconfig.gif	Event table bitmaps (appear if event table is viewed through Lotus Notes client).
	libbiald.a	Event listener for AIX systems.
	libbiald.so	Event listener for Solaris systems.
	nbiald.dll	Event listener for Windows systems.
<i>ProductDir</i> /connectors/messages	BIADominoConnector.txt	Contains logging and tracing messages.
<i>ProductDir</i> /respository/Domino	BIACN_Domino.txt	Connector property configuration file.
	BIADominoDocument.txt	Business object definition.

Note: On Windows systems, the installer adds an icon for the connector file to the IBM WebSphere Business Integration Adapters menu. For convenience you can also create a shortcut to this file on the desktop.

Once they are loaded onto your system, you must copy some files to the Domino server as described in “Copying files to the Domino server” on page 12.

For more information on installing the adapter, refer to the *Installation Guide for WebSphere Business Integration Adapters*.

Modifying the Domino server

You must make the following modifications to the Domino server before you can run the adapter:

1. Configure the server to run the Domino Internet Interoperability Protocol (IIOP) and HTTP server (for Web Applications), and configure the server to allow remote calls from an application. These changes are made with the Domino Administrator.
2. Modify the NOTES.INI file on the Domino server.
3. Copy the event table and associated files and the event listener to the required directories on the Domino server.
4. Configure the event table.
5. Make the NCSO.jar file accessible to the adapter.

Configuring changes with Domino Administrator

The Domino server must be configured to accept remote calls from an application for the adapter to operate. Configuration is done with the Domino Administrator (refer to your Lotus Domino administration documentation for details). The following sections describe what must be configured. Make sure that the server is restarted after these changes are made so that they will take effect.

HTTP

The HTTP server (for Web Applications) must be enabled on the Domino server. Refer to your Lotus Domino administration documentation for more information on how to set this up.

IIOp

The Domino Internet Interoperability Protocol (IIOp, sometimes called DIIOP) must be enabled. Make sure that a TCP/IP port number is configured and enabled for IIOp.

Java/COM

You must configure the username that is configured for the adapter, as well as the username(s) that will be specified in any incoming business objects, (if applicable) to run restricted and unrestricted Java agents (and methods, for R6).

Note: The username configured for the adapter is the value for `ApplicationUserName` as configured with Connector Configurator. Refer to “Application-specific configuration properties” on page 16 for more information.

Configure these as follows:

In R5:

1. Select the Security tab.
2. In the Java/COM Restrictions section, in the “Who can -” column under both “Run unrestricted Java/Javascript/COM” and “Run restricted Java/Javascript/COM,” enter the `ApplicationUserName` that is configured in the Connector Configurator file, as well as the username(s) that will be specified in any incoming business objects, if applicable.

In R6:

1. Select the Security tab.
2. In the Programmability Restrictions section, in the “Who can -” column, enter the `ApplicationUserName` that is configured in the Connector Configurator file, as well as the username(s) that will be specified in any incoming business objects, if applicable, for the following item:
 - Run restricted Java/Javascript/COM

Modifying NOTES.INI

Add one of the following lines to the NOTES.INI file on the Domino server, depending on the operating system:

On Windows: `EXTMGR_ADDINS=nbiald.dll`

On AIX: `EXTMGR_ADDINS=libbiald.a`

On Solaris: `EXTMGR_ADDINS=libbiald.so`

This enables the Domino server to call the event listener at startup.

Note: The event listener is a Domino Extension Manager library; therefore, the line should be entered on the same line in the NOTES.INI file as any other Domino Extension Manager libraries, separated from other entries by commas.

Copying files to the Domino server

You must copy the event table files and the event listener from the *ProductDir/connectors/Domino/dependencies* directory on the adapter host computer to the required directories on the Domino server. The following table lists the files and where they must be copied:

Table 2. Files to copy into Domino server directories

Directory to copy to	File	Description
Domino data directory	BIALD_EventTable.nsf	Event table
Domino data directory	BIALD_emptytrash.gif BIALD_movetotrash.gif BIALD_newconfig.gif	Event table bitmaps (appear if event table is viewed through Lotus Notes Client).
Copy one of the following:		
Domino server root	libbiald.a	Event listener for AIX systems. Copy this only if your Domino server runs on AIX.
Domino server root	libbiald.so	Event listener for Solaris systems. Copy this only if your Domino server runs on Solaris.
Domino server root	nbiald.dll	Event listener for Windows systems. Copy this only if your Domino server runs on Windows.

Copying event table files

Copy the following files from their installation directory to the Domino data directory. (On Windows, the default for this data directory is c:\Lotus\Domino\Data; on UNIX there is no default so the directory name will vary.)

BIALD_EventTable.nsf

BIALD_emptytrash.gif

BIALD_movetotrash.gif

BIALD_newconfig.gif

Copying the event listener

Copy the event listener from the installation directory (*ProductDir/connectors/Domino/dependencies*) to the Domino server root directory.

This will be one of the following files, depending on the operating system on which the Domino server will be run:

For AIX systems: libbiald.a

For Solaris systems: libbiald.so

For Windows systems: nbiald.dll

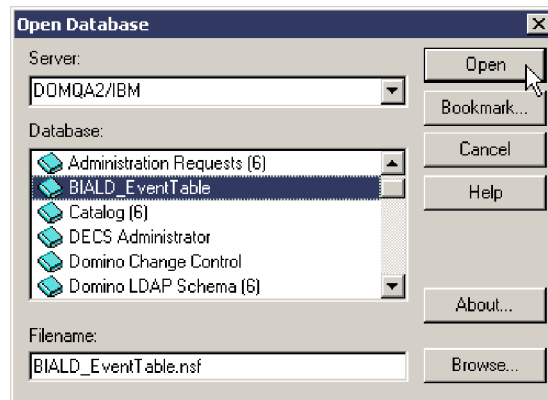
Typically, the root directory to which you will install is as follows, depending on your operating system: /opt/lotus/notes/latest/ibmpow for AIX, /opt/lotus/notes/latest/sunspa for Solaris, and c:\lotus\domino for Windows.

Configuring the event table

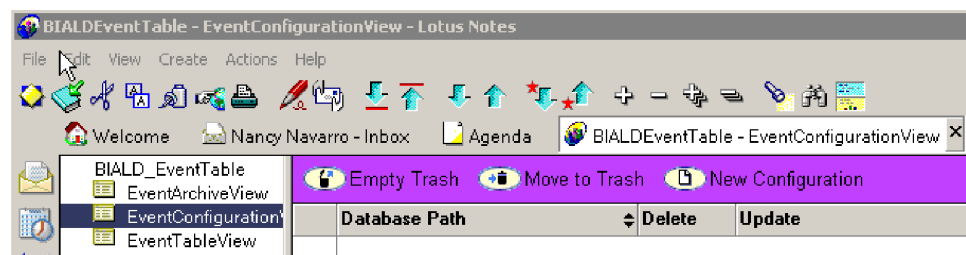
To allow the adapter's event listener to monitor events that occur on the Domino server, you must configure the adapter's event table. The event table is a Lotus Domino database file which you can open from a Lotus Notes client.

To configure the event table:

1. From a Lotus Notes client, select File > Database > Open and select the Domino server name to be supported by this adapter.
2. Select the BIALD_EventTable database. If it does not appear in the list, enter or browse for BIALD_EventTable.nsf in the Filename field. The window should appear similar to the following:



3. Click Open. The following window is displayed:



4. Select EventConfigurationView from the navigation view in the left panel.

- Select New Configuration from the menu bar. The following table will appear:

WBI Adapter for Lotus Domino ConfigurationTable:

Event Info	Value
Database Path:	''
Event Type:	<input type="checkbox"/> Create <input type="checkbox"/> Delete <input type="checkbox"/> Update

- In the Database Path field, enter the path of the database that you want the adapter to monitor. The path should be relative to your *Domino server root directory/Data*. For example, if the database is called Buyer and the path is *Domino server root/Data/BusApp/Buyer.nsf*, enter BusApp/Buyer in the Database Path field.
- In the Event Type row, select the verb types or verb type combination that will enable the adapter to monitor the desired event types, as follows:

For the adapter to monitor:	Select:
Create events	Create and Update
Delete events	Delete
Update events	Update

For example, the following selections enable the adapter to monitor create, update and delete events:

WBI Adapter for Lotus Domino ConfigurationTable:

Event Info	Value
Database Path:	'' BusApp/Buyer
Event Type:	<input checked="" type="checkbox"/> Create <input checked="" type="checkbox"/> Delete <input checked="" type="checkbox"/> Update

Note: It is not possible for the adapter to monitor Create events without also monitoring Update events. For monitoring Create events you must select Create *and* Update as shown in the following example:

WBI Adapter for Lotus Domino ConfigurationTable:

Event Info	Value
Database Path:	'' BusApp/Buyer
Event Type:	<input checked="" type="checkbox"/> Create <input type="checkbox"/> Delete <input checked="" type="checkbox"/> Update

- Select File > Save to save the configuration, then close the event table database.

Making NCSO.jar accessible

The file NCSO.jar is part of the Lotus Domino Toolkit for Java/CORBA, which should have come with your Lotus Domino server. (The file should be available on the Domino server and on Notes clients.) NCSO.jar contains the Lotus Domino API (known as Domino Objects for Java). You must update the adapter startup script to point to this file.

Location of NCSO.jar

Typically, NCSO.jar resides in the following directory on the Domino server:

```
DirectoryName\Data\domino\java
```

where *DirectoryName* is the name of the Domino server directory or the Notes client directory.

Note: For UNIX, substitute slashes (/) for backslashes in the pathname shown above.

Modifying the startup script

Change the following line in the adapter startup script to point to NCSO.jar.

On UNIX: Change

```
DOMINOJAVACLASSES=/server1/domino/java  
CON_SPEC_JAR_ONE=${DOMINOJAVACLASSES}/NCSO.jar
```

to

```
DOMINOJAVACLASSES=directoryname  
CON_SPEC_JAR_ONE=${DOMINOJAVACLASSES}/NCSO.jar
```

where

directoryname is the name of the directory containing NCSO.jar.

On Windows: Change

```
set DOMINOCLASSES=C:\Notes\Data\domino\java\NCSO.jar
```

to

```
set DOMINOCLASSES=directoryname\NCSO.jar
```

where

directoryname is the name of the directory containing NCSO.jar.

Configuring the connector

Connectors have two types of configuration properties: standard configuration properties and adapter-specific configuration properties. You must set the values of these properties before running the adapter with the Connector Configurator, a tool provided with the WebSphere Business Integration Adapters. When you have finished specifying values for the connector's configuration properties, Connector Configurator saves the values in a repository where it will be available to the adapter upon startup.

You use Connector Configurator to configure connector properties:

- For a description of Connector Configurator and step-by-step procedures, see Appendix B, "Connector Configurator," on page 47.
- For a description of standard connector properties, see "Standard configuration properties" on page 16 and then see Appendix A, "Standard configuration properties for connectors," on page 31.

- For a description of connector-specific properties, see “Application-specific configuration properties.”

Standard configuration properties

Standard configuration properties provide information that all adapters use. See Appendix A, “Standard configuration properties for connectors,” on page 31 for documentation of these properties.

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

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

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

Note that logging and tracing parameters are standard configuration properties. For more information about setting up logging and tracing, refer to Appendix A, “Standard configuration properties for connectors,” on page 31.

Application-specific configuration properties

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

To view and configure these properties, select the Connector Configurator **Application Config Properties** tab to add or modify configuration properties. For more information, see Appendix B, “Connector Configurator,” on page 47..

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

Table 3. Application-specific configuration properties for Lotus Domino

Property	Description	Type	Default value
DominoServerName	machine name of Domino server	string	none
ApplicationUserName	Domino user name	string	none
ApplicationPassword	Domino user password	string	none
EventDBName	Domino event database name	string	BIALD_EventTable.nsf
DocumentBOName	Name of the business object for Lotus Domino	string	DominoDocument
ForceSave	When the adapter is processing a Create or Update request, if ForceSave is true, the document is saved even if someone else edits and saves the document while the adapter is processing the request. The last version of the document is the one that is saved; the earlier version is discarded. If false, and someone else edits the document while the adapter is processing the request, the MakeResponse property determines the outcome.	Boolean	false

Table 3. Application-specific configuration properties for Lotus Domino (continued)

Property	Description	Type	Default value
MakeResponse	When the adapter is processing a Create or Update request, if ForceSave is false and MakeResponse is true, the current document becomes a response to the original document. If MakeResponse is false, the save is cancelled. If ForceSave is true, the MakeResponse property has no effect.	Boolean	true
ForceDelete	When the adapter is processing a Delete request, if ForceDelete is true the document is removed even if another user modifies the document after the adapter opens it. If false, the document is not removed if another user modifies it.	Boolean	false
InDoubtEvents	Recovery options for InProgress events. Possible values: Reprocess, FailOnStartup, LogError, or Ignore. Refer to Table 4 for descriptions of these options.	string	Reprocess
DateFormats	Possible formats of Created and LastModified dates for incoming business objects. When a business object is received from the integration broker, the Domino adapter must convert its Created and LastModified dates to the Java format understood by the Lotus Domino server. Multiple formats can be specified, separated by a semicolon (;). The adapter will compare the incoming date with the specified formats in the order listed. If no format is specified, the date format on the adapter's host system is used. The date format must conform to that required by the java.text.SimpleDateFormat API in the Java 2 Platform, Standard Edition (J2SE) v1.3.1.	string	MM/dd/yy hh:mm aaa; MM/dd/yy

You must configure DominoServerName, ApplicationUserName, and ApplicationPassword. If a server name, user name, and password are defined in the business object received from the broker, then these are used to establish the adapter's session with the Domino server. If the business object does not contain any of these items, the adapter uses the values defined for them in the configuration file.

Table 4 defines the possible values for InDoubtEvents. An in-doubt event is one that was being processed when the adapter terminates, and before the adapter could update the event status to indicate whether the event was either successfully sent or failed. When the adapter is restarted, it checks the event table for events that were in progress and take action depending upon which option has been configured for the InDoubtEvents property.

Table 4. Possible values for InDoubtEvents property

Value of InDoubtEvents	Description
Reprocess	Causes the event to be resubmitted for a subsequent poll call. Note: This could cause an event to be duplicated. To avoid this, use another option.
FailOnStartup	Logs a fatal error, shutting down the adapter.

Table 4. Possible values for *InDoubtEvents* property (continued)

Value of <i>InDoubtEvents</i>	Description
LogError	Logs an error without shutting down the adapter.
Ignore	Ignores the in-progress event records in the event table.

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 5 shows.

Table 5. Startup scripts for a connector

Operating system	Startup script
UNIX-based systems	connector_manager_ <i>connName</i>
Windows	start_ <i>connName</i> .bat

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

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


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


```
connector_manager_connName -start
```

where *connName* is the name of the connector and *brokerName* identifies your integration broker, as follows:

- For WebSphere InterChange Server, specify for *brokerName* the name of the ICS instance.
- For WebSphere message brokers (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, or WebSphere Business Integration Message Broker) or WebSphere Application Server, specify for *brokerName* a string that identifies the broker.

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

- From Adapter Monitor (WebSphere Business Integration Adapters product only), which is launched when you start System Manager
 - You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- From System Monitor (WebSphere InterChange Server product only)

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

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

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

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

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.

Creating multiple connector instances

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

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

Create a new directory

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

```
ProductDir\connectors\connectorInstance
```

where `connectorInstance` uniquely identifies the connector instance.

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

```
ProductDir\repository\connectorInstance
```

Create business object definitions

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

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

```
ProductDir\repository\initialConnectorInstance
```

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

Create a connector definition

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

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

Create a start-up script

To create a startup script:

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

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

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

Chapter 3. Lotus Domino adapter business objects

This chapter describes Lotus Domino business objects. As part of the Lotus Domino adapter, IBM provides you with a preconfigured business object that can be used as-is; no modification is necessary, and the required business object definition is installed as part of the installation process described in Chapter 2, “Installing and configuring the adapter,” on page 9.

For your background information, this chapter explains the structure of business objects, describes Lotus Domino business objects, and how the adapter processes them.

The chapter contains the following sections:

- “Defining metadata” on page 21
- “Overview of business object structure” on page 21
- “Lotus Domino business object definition” on page 22
- “Generating business objects” on page 24

Defining metadata

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

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

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

Overview of business object structure

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

- A type name
- Supported verbs
- Attributes

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

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

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

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

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

Lotus Domino business object definition

The Lotus Domino adapter uses one main business object, the DominoDocument business object. DominoDocument contains one child business object that is in use for this release: the DominoItem business object. The DominoItem business object, in turn has one child, the DominoItemValue business object.

Note: An additional child business object, the DominoEmbeddedObject business object, has been defined for DominoDocument but it is not used for this release of the adapter.

The DominoDocument business object represents a Lotus Domino document. A document can be any kind of Lotus Domino data note, such as a mail message, calendar entry, or “to do” item. The DominoItem business object represents items, which are components of Domino documents.

Note: The adapter supports only the following text, date/times and numbers simple Domino Item types in this release: AUTHORS, DATETIMES, NAMES, NUMBERS, READERS, TEXT. The adapter does not support the RICHTEXT Domino Item type.

Through the DominoDocument business object, the adapter supports the verbs Create, Update, Delete, Retrieve and Exists for inbound requests. That is, another application can request that on the Domino server, a new Domino document be created, an existing one be updated, or an existing one be deleted. It can also request that an existing Domino document be sent to the application, and check that a particular document already exists on the Domino server.

The adapter also supports the verbs Create, Update, and Delete for outbound events. That is, whenever a document on the Domino server is Created, Updated, or Deleted, notification of the event will be sent to the integration broker.

Figure 3 shows how the DominoDocument business object appears in the Business Object Designer.

	Pos	Name	Type	Key	Foreign	Required	Card	Maximum Length	Default	App Spec Info
1	1	NoteID	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
2	2	DatabaseName	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
3	3	UNID	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
4	4	ServerName	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
5	5	UserName	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
6	6	Password	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
7	7	Created	Date	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
8	8	FTSearchScore	Integer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
9	9	Key	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
10	10	LastModified	Date	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
11	11	NameOfProfile	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
12	12	ParentDocumentUNID	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
13	13	Signer	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
14	14	Size	Integer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
15	15	URL	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
16	16	Verifier	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
17	17	HasEmbedded	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
18	18	IsDeleted	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
19	19	IsProfile	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
20	20	IsResponse	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
21	21	IsSentByAgent	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
22	22	IsSigned	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
23	23	IsValid	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
24	24	EmbeddedObjects	DominoEmbeddedObject	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	N			
25	25	Items	DominoItem	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	N			
26	26	ObjectEventId	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
27	27			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		

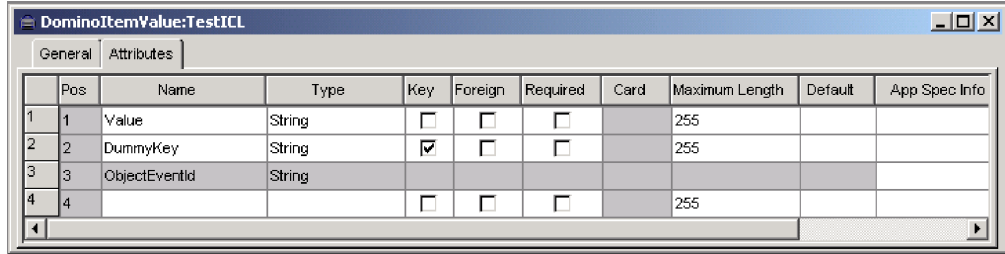
Figure 3. DominoDocument business object

Figure 4 shows the child business object DominoItem as it appears in Business Object designer.

	Pos	Name	Type	Key	Foreign	Required	Card	Maximum Length	Default
1	1	LastModified	Date	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
2	2	Name	String	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255	
3	3	Type	String	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255	
4	4	Values	DominoItemValue	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	N		
5	5	IsAuthors	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
6	6	ValueLength	Integer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
7	7	IsEncrypted	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
8	8	IsNames	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
9	9	IsProtected	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
10	10	IsReaders	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
11	11	IsSigned	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
12	12	IsSummary	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
13	13	DummyKey	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	
14	14	ObjectEventId	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
15	15			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	

Figure 4. DominoItem child business object

Figure 5 shows the DominoItemValue business object as it appears in Business Object Designer.



	Pos	Name	Type	Key	Foreign	Required	Card	Maximum Length	Default	App Spec Info
1	1	Value	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
2	2	DummyKey	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
3	3	ObjectEventId	String							
4	4			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		

Figure 5. DominoItemValue child business object

Except for the child business objects, all DominoDocument business object attributes have a cardinality 1 relationship. The child business object DominoItem has n cardinality; that is, the adapter will handle a Domino document with any number of subordinate document items. And within the DominoItem business object, the child business object DominoItemValue also has n cardinality; that is, there can be multiple values associated with an item. (For example, an e-mail message might have multiple recipients.)

Refer to Appendix C, “Lotus Domino business object attributes” of this manual for definitions of Lotus Domino business object attributes.

Generating business objects

During run-time, each time an event of interest occurs on the Lotus Domino server (that is, when a Domino document is created, updated or deleted) the connector obtains the event data from Lotus Domino. (“How the connector works” on page 3 describes this process in more detail.) The connector then maps the data to the business object definition to create a specific business object instance (commonly known as simply a “business object.”) It then sends this business object to the integration broker for processing, and the event information is in this way available to another application. The connector also receives business objects from the integration broker. Business objects received from the integration broker that originate from another application might request that Lotus Domino create, update or delete a document on the Lotus Domino server or that a document’s existence be checked, or that an entire Domino document be sent back to the broker, and from there to the requesting application.

Chapter 4. Troubleshooting and error handling

This chapter describes how the adapter for Lotus Domino handles errors. The adapter reports on the status of events, and generates error and tracing messages. This chapter describes these messages and provides troubleshooting tips. The chapter contains the following sections:

- “Event status”
- “Error messages” on page 25
- “Tracing” on page 29

Event status

You can review the status of an event in the event view and the archive view of the event table. This might help you determine the problem that has occurred in processing an event. In the event view, event status can be “Ready for Poll” or “In-progress.” In the archive view, event status can be “Success,” “Unsubscribed,” “Error Processing Event,” “Error Posting Event” or “Error Object Not Found.”

You can change the status of an event after resolving an error or for troubleshooting purposes, as follows:

1. Navigate to the event table archive view (EventArchiveView) or event view (EventTableView) from the Lotus Notes client.
2. Select the event whose status you wish to change.
3. Right click and select Edit.
4. Select the status you wish to change to from the drop-down box.
5. Select File > Save.

The event’s status will change and will be displayed in the applicable event table view.

Error messages

Error messages are logged to STDOUT or a file that you specify. The method for specifying a log file differs depending on your integration broker. For instructions on how to change where messages are logged, refer to the documentation for your particular integration broker, as follows:

- *Implementation Guide for WebSphere InterChange Server* (when ICS is used as the integration broker)
- *Implementing Adapters with WebSphere MQ Integrator Broker* (when WebSphere MQ Integrator Broker is used as the integration broker)
- *Implementing Adapters with WebSphere Application Server* (when WebSphere Application Server is used as the integration broker).

Table 6 on page 26 lists common errors generated by the adapter and recommended solutions. In the table, “xxx” and “yyy” represent strings in actual generated messages.

Table 6. Common adapter errors and recommended solutions

Error number	Error message	Explanation/Recommended solution
4001	Exception caught in xxx: yyy	An exception was caught in method xxx. The exception text is described in yyy, which may contain the message ID and text extracted from the NotesException class.
4006	No value for configuration property xxx. Using default value: yyy	If needed, set the property as appropriate using the connector configurator.
4008	No value for configuration property xxx.	If needed, set the property as appropriate using the connector configurator.
4034	Failed to deliver the Document BO to the broker, event # xxx, status yyy.	The status will indicate why the failure occurred. You can change the status to "ready for poll" once the problem has been corrected.
4036	No Subscription found for the attempted BO delivery, event # xxx, status yyy.	Verify collaborations and connectors are started.
4049	Failed to Retrieve Document with NoteID: xxx, event # yyy	A problem occurred retrieving the document specified in the event specified. Document may have been deleted subsequent to the specified event.
4050	Failed to Retrieve Event Record #: xxx	Verify that the BIALD_EventTable.nsf database is available and that the user specified in connector configuration has appropriate authority to read documents in the database.
4051	No more events to poll from the EventTable.	There are no events listed in the event table.No action is required.
4052	Failed to retrieve the Event View: xxx	There was a problem reading the event table view of the BIALD_EventTable.nsf database. Verify that the BIALD_EventTable.nsf database is available and that the user specified in connector configurator has appropriate authority to read documents in the database.
4059	Item Type is not recognized.	The adapter is processing a document that has an Item type that is not supported. The item will not appear in the DominoDocument Business Object. Refer to "Lotus Domino business object definition" for the supported Item types.
4064	Failed to open Database: xxx for Domino event, Document NoteID = yyy, event # zzz	Verify that the database is available and that the user specified in connector configuration has appropriate authority to read documents in the database.
4072	Encountered an IN_PROGRESS Event, event number # xxx. Connector has been configured to Log Error.	Based on connector configuration, In Progress events are logged as errors. If configured action is incorrect, update using connector configurator.
4073	Encountered an IN_PROGRESS Event, event number # xxx. Connector has been configured to Log Fatal Error.	Based on connector configuration, In Progress events are logged as fatal errors. If configured action is incorrect, update using connector configurator.
4074	Encountered an IN_PROGRESS Event. Connector has been configured to Ignore Event.	Based on connector configuration, In-Progress events are ignored. If configured action is incorrect, update using connector configurator.
4075	Encountered an IN_PROGRESS Event. Connector has been configured to Reprocess Event.	Based on connector configuration, In Progress events are reprocessed. If configured action is incorrect, update using connector configurator.

Table 6. Common adapter errors and recommended solutions (continued)

Error number	Error message	Explanation/Recommended solution
4076	Failed to validate session in method xxx.	Verify Domino server is started. Verify server, username and password are configured correctly in connector configuration.
4077	Failed to establish connection to event table database/view in method xxx.	Verify Domino server is started. Verify server, username and password are configured correctly in connector configuration.
4081	Invalid document database path for event # xxx	This message indicates that the database does not exist at the configured path. The database might have been deleted after the event was sent to the event table and before the adapter attempted to process the event.
4084	Domino Session is no longer valid.	Verify Domino server is started. Verify that server, username and password are configured correctly in connector configuration or Business Object.
5001	Error accessing Domino server Server: xxx Username: yyy Error text: zzz	An error occurred accessing the Domino server. Verify server is started.
5002	Error accessing database: xxx. Error text: yyy	Verify configured user has appropriate authority to access the database.
5003	Database xxx not found.	Database specified in the business object was not found. Verify database path/name specified in business object.
5005	Error opening database: xxx Error text: yyy	See the Error text for more information. Verify configured user has appropriate authority to access database.
5006	Unsupported verb: xxx	The specified verb in the incoming business object is not supported by the connector. Supported verbs are as follows: Create, Retrieve, Update, Delete, Exist.
5007	Error opening document. NoteID=xxx. Error text: yyy	The business object handler received a business object with a NoteID that is not in the specified database. The document with this NoteID might have been deleted. If the NoteID is incorrect, correct it and resend the business object.
5008	Error opening document. NoteID=xxx. Document not found in database.	A document with the specified NoteID was not found in the database.
5009	Error creating document. Error text: xxx	The user might not have the required authority to create a document in the specified database. Change the authority level or use different user login information.
5010	Error retrieving NoteID from Document.	During processing of Create verb, an error occurred retrieving the NoteID from the document after it was successfully created in the database. The NoteID will not be added to the Business Object. If the NoteID is required, access the document using the Notes Client to view.
5011	Error deleting document. NoteID=xxx. Error text: yyy	User might not have authority to delete the document. Change authority or use different user login information.
5012	Error validating document. NoteID=xxx. Error text: yyy	An error occurred validating the specified document. Refer to the error text for more information.

Table 6. Common adapter errors and recommended solutions (continued)

Error number	Error message	Explanation/Recommended solution
5013	Error saving document. Error text: xxx	Explanation: An error occurred saving the specified document. Refer to the error text for more information.
5014	The Document was not saved.	Possible causes: The document was unchanged by the request, or because of settings of ForceSave and MakeResponse properties. The document was not saved either because it was not changed, or because of connector properties. Set connector application-specific properties; ForceSave, MakeResponse, as appropriate See Table 3 on page 16 or your Domino documentation for more information.
5015	Attribute xxx not found in Business Object yyy.	The specified required attribute was not found in the Business Object. Verify that correct version of the business object is being used.
5016	Attribute xxx is not String type in Business Object yyy	The specified attribute is not a String type in the specified business object as expected by the adapter. Verify that correct version of the business object is being used.
5017	Attribute xxx is not boolean type in Business Object yyy	The specified attribute is not a boolean type in the specified Business Object as expected by the adapter. Verify that correct version of the business object is being used.
5018	Attribute xxx is not Business Object type in Business Object yyy	The specified attribute is not a Business Object type in the specified business object as expected by the adapter. Verify that correct version of the business object is being used.
5019	Error adding Item xxx to document. Error text: yyy	User might not have authority. Change authority or use different user login information.
5020	Error retrieving Document parent Database. Error text: yyy	An error occurred retrieving the parent database of the document. See the Error text for more information.
5021	Error retrieving current Session from the Database. Error text: xxx	An error occurred retrieving the current Session from the Database. See the Error text for more information.
5022	Error creating DateTime object from the String xxx. Error text: yyy	Explanation: An error occurred creating a DateTime object from the specified String. Date Strings need to conform to the patterns specified in the java.text.SimpleDateFormat API. Refer to "Application-specific configuration properties" for more information.
5023	No ItemHandler defined for type: xxx. Valid types include "Double", "String," and "Date."	No ItemHandler is defined for the type specified in the DominoItem.type attribute. Verify a supported type is set in the DominoItem.type attribute.
5024	Error setting flag xxx in Item yyy. Exception text: zzz	An error occurred setting the specified flag in the specified item. See the error text for more information.
5025	Error converting String xxx to a Double.	An error occurred converting the specified String to a Double object. Verify that DominoItemValue.value is a valid Double value in the incoming business object.
5026	Error converting Domino Document to a Business Object. NoteID=xxx Error text: yyy	An error occurred converting a document to a business object. See the error text for more information.

Table 6. Common adapter errors and recommended solutions (continued)

Error number	Error message	Explanation/Recommended solution
5029	Error removing Item xxx from Document. NoteID=yyy Error text: zzz	An error occurred removing the specified Item from the Document. See the error text for more information.
5030	Error initializing and validating Business Object attributes. BO Name: xxx Error text: yyy	Make sure that a value is specified in the incoming business object for any attributes with IsRequired set to Yes.
5031	Unsupported subverb: xxx for DominoItem Name=yyy	The specified subverb is not supported by the connector. Verify that a supported sub-verb is specified for the DominoItem business object. One of: Create, Update, Delete.
5032	Error parsing Date String using default format for locale. Date: xxx Locale: yyy	An error occurred parsing the specified date string using the system default format for the specified locale. Verify that the Date string matches the system default format for the Locale of the business object. If not, specify required date format in the DateFormats connector application-specific property. Refer to "Application-specific configuration properties" for more information.
5033	Error parsing Date String using configured format. Date: xxx Locale: yyy Format: zzz.	An error occurred parsing the specified date String using the specified format and Locale. Verify that the Date string matches one of the formats specified in the DateFormats connector application-specific property. Refer to "Application-specific configuration properties" for more information.
5034	Unable to generate Date object using configured Date Format strings. Date: xxx Locale: yyy.	All date formats will be attempted before an error is logged. Verify that all attempts to generate the Date have failed. If so, specify required date format in the DateFormats connector application-specific property. Refer to "Application-specific configuration properties" for more information.
5035	The NoteID was not specified in the Business Object.	Verify that NoteID is always specified in the incoming business object.
5038	Invalid value specified for Name attribute. Name=xxx, BO=yyy.	Verify that the attribute does not have a CxIgnore or CxBlank value.

Tracing

Tracing is an optional debugging feature you can turn on to closely follow connector behavior. Trace messages, by default, are written to STDOUT. Tracing properties are set with the standard configuration properties AgentTraceLevel, TraceFileName, and ControllerTraceLevel. For more on configuring trace messages, refer to Appendix A, "Standard configuration properties for connectors," on page 31.

Table 7 on page 30 shows the type of information you will receive for different tracing levels.

Table 7. Tracing messages content

Level	Description
Level 0	Identifies the connector version. No other tracing is performed at this level.
Level 1	<ul style="list-style-type: none"> • Provides status information. • Provides key information on each business object processed. • Records each time polling occurs.
Level 2	<ul style="list-style-type: none"> • Identifies the business object handler used for each object that the connector processes. • Logs each time a business object is posted to the integration broker. • Indicates each time a request business object is received.
Level 3	<ul style="list-style-type: none"> • Identifies the foreign keys being processed, if applicable. These messages appear when the connector has encountered a foreign key in a business object or when the connector sets a foreign key in a business object. • Business object processing. Examples of this include finding a match between business objects, or finding a business object in an array of child business objects.
Level 4	<ul style="list-style-type: none"> • Identifies application-specific information. Examples of this include the values returned by the methods that process the application-specific information fields in business objects. • Identifies when the connector enters or exits a function. These messages help trace the process flow of the connector. • Records any thread-specific processing. For example, if the connector spawns multiple threads, a message logs the creation of each new thread.
Level 5	<ul style="list-style-type: none"> • Indicates connector initialization. This type of message can include, for example, the value of each connector configurator property that has been retrieved from the broker. • Details the status of each thread that the connector spawns while it is running. • Represents statements executed in the application. The connector log file contains all statements executed in the target application and the value of any variables that are substituted, where applicable. • Records business object dumps. The connector provides a text representation of a business object before it begins processing (showing the object that the connector receives from the collaboration) as well as after it finishes processing the object (showing the object that the connector returns to the collaboration).

Appendix A. Standard configuration properties for connectors

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

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

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

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

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

New and deleted properties

These standard properties have been added in this release.

New properties

- XMLNamespaceFormat

Deleted properties

- RestartCount
- RHF2MessageDomain

Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties
- Connector-specific configuration properties

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

Using Connector Configurator

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

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

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

Setting and updating property values

The default length of a property field is 255 characters.

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

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

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

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

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

Summary of standard properties

Table 8 on page 33 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker, as standard property dependencies are based on RepositoryDirectory.

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

Table 8. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdminInQueue	Valid JMS queue name	CONNECTORNAME /ADMININQUEUE	Component restart	Delivery Transport is JMS
AdminOutQueue	Valid JMS queue name	CONNECTORNAME/ADMINOUTQUEUE	Component restart	Delivery Transport is JMS
AgentConnections	1-4	1	Component restart	Delivery Transport is MQ or IDL: Repository directory is <REMOTE>
AgentTraceLevel	0-5	0	Dynamic	
ApplicationName	Application name	Value specified for the connector application name	Component restart	
BrokerType	ICS, WMQI, WAS			
CharacterEncoding	ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437 Note: This is a subset of supported values.	ascii7	Component restart	
ConcurrentEventTriggeredFlows	1 to 32,767	1	Component restart	Repository directory is <REMOTE>
ContainerManagedEvents	No value or JMS	No value	Component restart	Delivery Transport is JMS
ControllerStoreAndForwardMode	true or false	True	Dynamic	Repository directory is <REMOTE>
ControllerTraceLevel	0-5	0	Dynamic	Repository directory is <REMOTE>
DeliveryQueue		CONNECTORNAME/DELIVERYQUEUE	Component restart	JMS transport only
DeliveryTransport	MQ, IDL, or JMS	JMS	Component restart	If Repository directory is local, then value is JMS only
DuplicateEventElimination	True or False	False	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		CONNECTORNAME/FAULTQUEUE	Component restart	JMS transport only

Table 8. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory or CxCommon.Messaging.jms.SonicMQFactory or any Java class name	CxCommon.Messaging.jms.IBMMQSeriesFactory	Component restart	JMS transport only
jms.MessageBrokerName	If FactoryClassName is IBM, use crossworlds.queue.manager. If FactoryClassName is Sonic, use localhost:2506.	crossworlds.queue.manager	Component restart	JMS transport only
jms.NumConcurrentRequests	Positive integer	10	Component restart	JMS transport only
jms.Password	Any valid password		Component restart	JMS transport only
jms.UserName	Any valid name		Component restart	JMS transport only
JvmMaxHeapSize	Heap size in megabytes	128m	Component restart	Repository directory is <REMOTE>
JvmMaxNativeStackSize	Size of stack in kilobytes	128k	Component restart	Repository directory is <REMOTE>
JvmMinHeapSize	Heap size in megabytes	1m	Component restart	Repository directory is <REMOTE>
ListenerConcurrency	1- 100	1	Component restart	Delivery Transport must be MQ
Locale	en_US, ja_JP, ko_KR, zh_CN, zh_TW, fr_FR, de_DE, it_IT, es_ES, pt_BR Note: This is a subset of the supported locales.	en_US	Component restart	
LogAtInterchangeEnd	True or False	False	Component restart	Repository Directory must be <REMOTE>
MaxEventCapacity	1-2147483647	2147483647	Dynamic	Repository Directory must be <REMOTE>
MessageFileName	Path or filename	InterchangeSystem.txt	Component restart	
MonitorQueue	Any valid queue name	CONNECTORNAME/MONITORQUEUE	Component restart	JMS transport only: DuplicateEvent Elimination must be True
OADAutoRestartAgent	True or False	False	Dynamic	Repository Directory must be <REMOTE>

Table 8. Summary of standard configuration properties (continued)

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

Table 8. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
SynchronousRequestTimeout	0 - any number (milliseconds)	0	Component restart	Delivery Transport is JMS
SynchronousResponseQueue		CONNECTORNAME/ SYNCHRONOUSRESPONSEQUEUE	Component restart	Delivery Transport is JMS
WireFormat	CwXML, CwBO	CwXML	Agent restart	CwXML if Repository Directory is not <REMOTE>: CwBO if Repository Directory is <REMOTE>
WsifSynchronousRequest Timeout	0 - any number (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.

AdminInQueue

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

The default value is CONNECTORNAME/ADMININQUEUE.

AdminOutQueue

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

The default value is CONNECTORNAME/ADMINOUTQUEUE.

AgentConnections

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

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

ConcurrentEventTriggeredFlows

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

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

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

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

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

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

ContainerManagedEvents

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

The default value is No value.

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

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

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

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

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

ControllerStoreAndForwardMode

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

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

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

The default is true.

ControllerTraceLevel

Applicable only if RepositoryDirectory is <REMOTE>.

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

DeliveryQueue

Applicable only if DeliveryTransport is JMS.

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

The default value is CONNECTORNAME/DELIVERYQUEUE.

DeliveryTransport

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

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

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

WebSphere MQ and IDL

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

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

JMS

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

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

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

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

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

- The LDR_CNTRL environment variable in the CWSHaredEnv.sh script.

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

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

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

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

DuplicateEventElimination

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

This property can also be set to false.

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

FaultQueue

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

The default value is `CONNECTORNAME/FAULTQUEUE`.

JvmMaxHeapSize

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

The default value is 128m.

JvmMaxNativeStackSize

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

The default value is 128k.

JvmMinHeapSize

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

The default value is 1m.

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

The default is `crossworlds.queue.manager`.

jms.NumConcurrentRequests

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

The default value is 10.

jms.Password

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

There is no default.

jms.UserName

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

There is no default.

ListenerConcurrency

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

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

Locale

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

A locale name has the following format:

ll_TT.codeset

where:

<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 list. To add other supported values to the drop list, you must manually modify the

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

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

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

LogAtInterchangeEnd

Applicable only if 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. Specify the message filename in an absolute path if the message file is not located in the standard location.

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

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

MonitorQueue

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

The default value is CONNECTORNAME/MONITORQUEUE

OADAutoRestartAgent

Valid only when the RepositoryDirectory is <REMOTE>.

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

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

The default value is false.

OADMaxNumRetry

Valid only when the RepositoryDirectory is <REMOTE>.

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

The default value is 1000.

OADRetryTimeInterval

Valid only when the RepositoryDirectory is <REMOTE>.

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

The default is 10.

PollEndTime

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

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

PollFrequency

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

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

The default is 10000.

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

PollQuantity

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

PollStartTime

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

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

RequestQueue

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

The default value is `CONNECTOR/REQUESTQUEUE`.

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

The default is 3.

RestartRetryInterval

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

The default is 1.

SourceQueue

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

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see “`ContainerManagedEvents`” on page 37.

The default value is `CONNECTOR/SOURCEQUEUE`.

SynchronousRequestQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default is `CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE`

SynchronousResponseQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default is `CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE`

SynchronousRequestTimeout

Applicable only if `DeliveryTransport` is JMS.

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

The default value is 0.

WireFormat

Message format on the transport.

- If the `RepositoryDirectory` is a local directory, the setting is `CwXML`.
- If the value of `RepositoryDirectory` is `<REMOTE>`, the setting is `CwB0`.

WsifSynchronousRequest Timeout

WAS integration broker only.

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

The default value is 0.

XMLNameSpaceFormat

WebSphere message brokers and WAS integration broker only.

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

The default value is short.

Appendix B. Connector Configurator

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

You use Connector Configurator to:

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

Note:

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

The topics covered in this appendix are:

- “Overview of Connector Configurator” on page 47
- “Starting Connector Configurator” on page 48
- “Creating a connector-specific property template” on page 49
- “Creating a new configuration file” on page 51
- “Setting the configuration file properties” on page 54
- “Using Connector Configurator in a globalized environment” on page 60

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 48).

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 49 to set up a new one.

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

Starting Connector Configurator

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

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

Running Configurator in stand-alone mode

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

To do so:

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

You may choose to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see “Completing a configuration file” on page 53.)

Running Configurator from System Manager

You can run Connector Configurator from System Manager.

To run Connector Configurator:

1. Open the System Manager.
2. In the System Manager window, expand the **Integration Component Libraries** icon and highlight **Connectors**.
3. From the System Manager menu bar, click **Tools>Connector Configurator**. The Connector Configurator window opens and displays a **New Connector** dialog box.
4. When you click the pull-down menu next to **System Connectivity Integration Broker**, you can select ICS, WebSphere Message Brokers or WAS, depending on your broker.

To edit an existing configuration file:

1. In the System Manager window, select any of the configuration files listed in the Connector folder and right-click on it. Connector Configurator opens and displays the configuration file with the integration broker type and file name at the top.
2. Click the Standard Properties tab to see which properties are included in this configuration file.

Creating a connector-specific property template

To create a configuration file for your connector, you need a connector-specific property template as well as the system-supplied standard properties.

You can create a brand-new template for the connector-specific properties of your connector, or you can use an existing file as the template.

- To create a new template, see “Creating a new template” on page 49.
- To use an existing file, simply modify an existing template and save it under the new name.

Creating a new template

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

To create a template:

1. Click **File>New>Connector-Specific Property Template**.
2. The **Connector-Specific Property Template** dialog box appears, with the following fields:
 - **Template, and Name**
Enter a unique name that identifies the connector, or type of connector, for which this template will be used. You will see this name again when you open the dialog box for creating a new configuration file from a template.
 - **Old Template, and Select the Existing Template to Modify**
The names of all currently available templates are displayed in the Template Name display.

- To see the connector-specific property definitions in any template, select that template's name in the **Template Name** display. A list of the property definitions contained in that template will appear in the **Template Preview** display. You can use an existing template whose property definitions are similar to those required by your connector as a starting point for your template.
3. Select a template from the **Template Name** display, enter that template name in the **Find Name** field (or highlight your selection in **Template Name**), and click **Next**.

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

Specifying general characteristics

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

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

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

Specifying values

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

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

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

Setting dependencies

When you have made your changes to the **General** and **Value** tabs, click **Next**. The **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, your first step is to select an integration broker. The broker you select determines the properties that will appear in the configuration file.

To select a broker:

- In the Connector Configurator home menu, click **File>New>Connector Configuration**. The **New Connector** dialog box appears.
- In the **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.

You can also do this:

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

Creating a configuration file from a connector-specific template

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

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

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

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

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

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

Select the template you want to use and click **OK**.
3. A configuration screen appears for the connector that you are configuring. The title bar shows the integration broker and connector names. You can fill in all the field values to complete the definition now, or you can save the file and complete the fields later.
4. To save the file, click **File>Save>To File** or **File>Save>To Project**. To save to a project, System Manager must be running.

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

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

Using an existing file

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

- A connector definition file.

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

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

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

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

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

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

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

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

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

Completing a configuration file

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

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

1. Under the **Standard Properties** tab, select the value field for the BrokerType property. In the drop-down menu, select the value ICS, WMQI, or WAS.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 56..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

Setting the configuration file properties

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

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

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

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

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

- Associated Maps
- Resources
- Messaging (where applicable)

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

Standard properties differ from connector-specific properties as follows:

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

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

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

Setting standard connector properties

To change the value of a standard property:

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

Setting application-specific configuration properties

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

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

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

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

Encryption for connector properties

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

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

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

Update method

Refer to the descriptions of update methods found in the *Standard configuration properties for connectors* appendix, under “Setting and updating property values” on page 32.

Specifying supported business object definitions

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

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

If ICS is your broker

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

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

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

To delete a business object from the supported list:

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

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

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

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

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

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

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

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

If a WebSphere Message Broker is your broker

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

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

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

If WAS is your broker

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

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

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

Associated maps (ICS only)

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

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

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

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

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

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

- **Business Object Name**

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

- **Associated Maps**

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

- **Explicit**

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

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

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

To explicitly bind a map:

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

Resources (ICS)

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

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

Messaging (ICS)

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

Setting trace/log file values

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

To change the logging and tracing values:

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

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

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

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

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

Data handlers

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

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

Saving your configuration file

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

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

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

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

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

Changing a configuration file

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

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

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

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

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

Appendix C. Lotus Domino business object attributes

This appendix defines the attributes of the DominoDocument business object, the DominoItem child business object, and the DominoItemValue child business object.

DominoDocument business object attributes

Attribute Name	Type	Cardinality	Description
NoteID	String	1	The NoteID of the document.
DatabaseName	String	1	The name of the database that contains the document.
UNID	String	1	The universal ID of the document.
ServerName	String	1	The name of the Domino server. If blank, the value of the ServerName configuration property is used.
UserName	String	1	The user name that enables the adapter to establish a session with the Domino server. If blank, the value of the UserName configuration property is used.
Password	String	1	The user password enabling the adapter to establish a session with the Domino server. If blank, the value of the Password configuration property is used.
Created	Date	1	The date and time the document was created. This date must be in one of the formats specified in the DateFormats configuration property.
FTSearchScore	Integer	1	The relevance value if the document was retrieved from a full-text search.
Key	String	1	The profile key of a profile document.
LastModified	Date	1	The date and time the document was last modified. This date must be in one of the formats specified in the DateFormats configuration property.
NameOfProfile	String	1	The profile name of a profile document.
ParentDocumentUNID	String	1	The universal ID of a document's parent, if the document is a response.
Signer	String	1	The signer of the document, if the document is signed.
Size	Integer	1	The size of the document in bytes, including any attachments.
URL	String	1	The Domino URL for the document's parent object.
Verifier	String	1	The name of the certificate verifying a signature if the document is signed.
HasEmbedded	Boolean	1	True if the document has embedded objects.
IsDeleted	Boolean	1	True if the document has been deleted.
IsProfile	Boolean	1	True if the document is a profile document.
IsResponse	Boolean	1	True if the document is a response document.
IsSentByAgent	Boolean	1	True if the document was mailed by a program.
IsSigned	Boolean	1	True if the document is signed.
IsValid	Boolean	1	True if the document exists (is not a deletion stub).

Attribute Name	Type	Cardinality	Description
EmbeddedObjects	DominoEmbeddedObject	1	Not used in this release.
Items	DominoItem	n	Child business object representing a document item.
ObjectEventID	String	1	A value generated by the Websphere Business integration system to track business objects, requests, and responses.

DominolItem business object attributes

DominolItem is a child of the DominoDocument business object.

Attribute Name	Type	Cardinality	Description
LastModified	Date	1	The date and time the document was last modified.
Name	String	1	The name of the item. Required.
Type	String	1	One of the following item types: Item.AUTHORS, Item.DATETIMES, Item.NAMES, Item.NUMBERS, Item.READERS, Item.TEXT. Required.
Values	DominolItemValue	n	Child business object.
ValueLength	Integer	1	The item's length in bytes.
isAuthors	Boolean	1	True if item is of type AUTHORS.
isEncrypted	Boolean	1	True if item is encrypted.
isNames	Boolean	1	True if item is of type NAMES.
isProtected	Boolean	1	True if editor access is required to modify the item.
isReaders	Boolean	1	True if the item is of type READERS.
isSigned	Boolean	1	True if the document is signed.
isSummary	Boolean	1	True if the item value can appear in a view.
ObjectEventID	String	1	A value generated by the Websphere Business integration system to track business objects, requests, and responses.

DominolItemValue business object attributes

DominolItemValue is a child of the DominolItem business object.

Attribute Name	Type	Cardinality	Description
Value	String	1	The value of the item.
ObjectEventID	String	1	A value generated by the Websphere Business integration system to track business objects, requests, and responses.

Index

A

- adapter architecture 1
- adapter framework, definition of 1
- adapter framework, version number 5
- adapter, configuration 15
- AIX, versions supported 6
- application-specific component 2
- application-specific configuration properties 16
- ApplicationPassword 16
- ApplicationUserName, configuring in Domino server 11
- architecture, adapter 1
- archive view 2
- attributes, in business objects 21
- attributes, of business object 63

B

- broker, integration
 - installing 6
- brokers, integration
 - version numbers 5
- business object handler
 - handler, business object 2
- business object structure 21
- business object, child 21
- business object, processing 2
- business objects
 - attributes 63
 - DominoDocument 22
 - DominoItem 22
 - DominoItemValue 22
- business objects, description 21
- business objects, for Lotus Domino adapter 21
- business objects, generating 24

C

- cardinality, in business objects 22
- component, application-specific 2
- configuration 14
 - adapter 15
 - ApplicationUserName in Domino server 11
 - connector 15
 - Create event 14
 - database to monitor 14
 - IIOP 11
 - Domino Internet Interoperability Protocol 11
 - event table 13
 - Event Type 14
 - HTTP 11
 - IIOP 11
 - Java/Javascript/COM on Domino server 11
 - NOTES.INI 11
 - Update event 14

- configuration properties 16
- configuration properties, application-specific 16
- configuration properties, standard 16
- configuration property
 - ApplicationPassword 16
 - DocumentBOName 16
 - DominoServerName 16
 - EventDBName 16
 - ForceDelete 17
 - ForceSave 16
 - InDoubtEvents 17
 - MakeResponse 17
- configuration view 2
- configuration, of the adapter 9
- configuring 14
- configuring the adapter 9
- configuring the event table 13
- connector architecture 2
- connector component 2
- connector framework 2
 - framework, connector 2
- connector, configuration 15
- connector, how it works
 - how the connector works 3
- conventions, typographic v
- copying files to Domino server 12
- copying, event table 12
- Create event 14

D

- database, selecting 14
- Delete event 14
- Delete event, configuration 14
- directories, on Domino server 12
- documentation, related v
- DocumentBOName 16
- documents, related v
- Domino Administrator, making changes with 11
- Domino API 14
- Domino Item types, supported 22
- Domino Objects for Java 14
- Domino server, configuration for adapter 10
- Domino server, files to copy 12
- Domino Server, installing 7
- Domino server, modifying 10
- Domino Server, versions supported 6
- DominoDocument business object 22
- DominoItem 22
- DominoItemValue 22
- DominoServerName 16

E

- error handling 25
- error messages 25
- error messages, logging 25

- errors, reporting 25
- event listener 2
- event listener, copying to Domino server 12
- event processing, steps 3
- event status, in event table 25
- event status, reviewing 25
- event table 1, 2
 - event status 25
- event table files 12
- event table, configuring 13
- event table, copying 12
- event table, copying to Domino server 12
- Event Type 14
- event view 2
- EventDBName 16
- events, selecting to monitor 14
- example uses 1

F

- files, copied to Domino server 12
- files, copying for event table 12
- files, for event table 12
- files, to copy to Domino server 12
- ForceDelete 17
- ForceSave 16

H

- how the adapter would be used 1
- HTTP, configuring 11

I

- IIOP, configuring 11
- InDoubtEvents 17
- InDoubtEvents, values 17, 18
- installation steps 9
- installation, of the adapter 9
- installation, overview of 9
- installing the adapter 9
- integration broker, installing 6
- integration brokers, version numbers 5
- Internet Interoperability Protocol, configuring 11

J

- Java runtime version required 6
- Java/Javascript/COM configuration 11
- JRE version required 6

L

- Lotus Domino API 14
- Lotus Domino, versions of 1

M

- MakeResponse 17
- messages, error 25
- messages, tracing 29
- meta-data, definition 21
- Microsoft Windows, versions supported 6

N

- NCSO.jar 14
- NCSO.jar, and startup script 15
- NOTES.INI configuration 11

O

- operating systems, supported 6
- overview, of the adapter
 - introduction, to the adapter 1

P

- processing, events 3
- processing, requests 4
- ProductDir, definition of vi
- properties, configuration 16
- properties, configuration, application-specific 16

R

- related documents v
- request processing, steps 4

S

- selecting database 14
- Solaris, versions supported 6
- startup script, modifying 15
- status information
 - status, event
 - event status 2
- supported verbs 22

T

- trace levels, definitions 29
- trace messages 29
- tracing 29
- troubleshooting 25
- typographic conventions v

U

- UNIX, versions supported 6
- Update event 14
- usage example 1
- usage scenario 1

V

- verbs, supported 22

- view

- archive, in event table 2
 - configuration, in event table 2
 - event, in event table 2
- views, event table 2
- views, of data in event table 2

W

- WBIA, installation 9
- WebSphere Business Integration Adapters, installation 9
- Windows, versions supported 6

Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

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

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

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

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



WebSphere Business Integration Adapter Framework V2.4.0



Printed in USA