IBM WebSphere Business Integration Adapters

# Adapter for eMatrix User Guide

*V 2.1.0*

**Note!**

Before using this information and the product it supports, read the information in "Notices" on page 85

**19December2003**

This edition of this document applies to the adapter for eMatrix version 2.1.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

# Contents

# About this document

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

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

## Audience

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

## Related documents

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

You can install related documentation from the following sites:

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

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

## Typographic conventions

This document uses the following conventions:

| | |
|---|---|
| `courier font` | Indicates a literal value, such as a command name, file name, information that you type, or information that the system prints on the screen. |
| *italic* | Indicates a new term the first time that it appears, a variable name, or a cross-reference. |

| | |
|---|---|
| *blue outline* | A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference. |
| `{ }` | In a syntax line, curly braces surround a set of options from which you must choose one and only one. |
| `\|` | In a syntax line, a pipe separates a set of options from which you must choose one and only one. |
| `[ ]` | In a syntax line, square brackets surround an optional parameter. |
| `...` | In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options. |
| `< >` | Angle brackets surround individual elements of a name to distinguish them from each other, as in `<server_name><`*`connector_name`*`>tmp.log`. |
| `/, \` | In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes. All product pathnames are relative to the directory where the connector for eMatrix is installed on your system. |
| *`%text%`* and *`$text`* | Text within percent (%) signs indicates the value of the Windows `text` system variable or user variable. The equivalent notation in a UNIX environment is *`$text`*, indicating the value of the *`text`* UNIX environment variable. |
| *ProductDir* | Represents the directory where the product is installed. For IBM WebSphere InterChange Server, the default product directory is IBM\WebSphereICS. For IBM WebSphere Business Integration Adapters, the default product directory is WebSphereAdapters. |

# New in this release

## New in release 2.1.0

The adapter for eMatrix version 2.1.0 supports the eMatrix Server and Collaboration Server version 10.0.0.0.

Beginning with version 2.1.0, the adapter for eMatrix is no longer supported on Microsoft Windows NT.

## New in release 2.0.x

Updated in July, 2003. The adapter for eMatrix can now use WebSphere Application Server as an integration broker.

The adapter now runs on the following platforms:
- Solaris 7, 8
- AIX 5.1
- HP-UX 11.i

## New in release 1.0.x

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

The eMatrix adapter is now globalized but the messages are not translated.

# Chapter 1. Overview of the adapter

This chapter describes the IBM(R) WebSphere(R) Business Integration adapter for eMatrix(R) and the associated system architecture.

MatrixOne eMatrix(TM) is a product lifecycle management solution that enables users to centralize control over all documents and data flows related to the product life cycle, and to monitor access to them. The adapter for eMatrix enables bidirectional, real-time integration between the eMatrix global collaboration platform and other enterprise applications.

The adapter communicates with the eMatrix collaboration server, which is installed separately from the application itself. For details on downloading and installing the eMatrix collaboration server, refer to the eMatrix documentation.

The adapter works with three integration brokers; the WebSphere Message Brokers, the WebSphere InterChange Server (ICS), and the WebSphere Application Server (WAS). For more information about the relationship of the integration broker to the adapter, see *IBM WebSphere Business Integration System Administration Guide*.

This chapter contains the following sections:

## Adapter architecture

The adapter for eMatrix consists of a connector, the Java Program Object (JPO), and an Object Discovery agent (ODA) with the adapter framework. Together they enable eMatrix to exchange information with an integration broker, and thereby to communicate with external applications such as accounting and CRM packages.

The adapter includes three primary components:

- **Connector**

  Acts as an intermediary between the integration broker and the application in the transfer of data. The connector is metadata-driven.
- **Java Program Object (JPO)**

  There may be more than one. They run within eMatrix and they store information about events in the eMatrix database.
- **Object Discovery Agent (ODA)**

  A design-time tool that queries the application to determine existing eMatrix business object types and relationships. These form the basis of the WebSphere business object definitions. The connector uses these definitions at run time to convert application data into business objects.

The adapter manipulates eMatrix business objects and connections within eMatrix. It may also execute MQL statements. Through the use of MQL commands, the adapter can perform almost any action supported by eMatrix.

The adapter communicates with the eMatrix application using a Java-based API called the eMatrix adapter development kit (ADK) and a separate eMatrix collaboration server. The eMatrix ADK enables the adapter to query and manipulate objects in the application by passing requests to the application through the eMatrix collaboration server.

## Architectural overview

The adapter mediates data flows to and from the integration broker and the application. The flows are:

- Request processing, initiated by the integration broke r (ICS, WebSphere message brokers, or WAS)
- Event processing, initiated by the eMatrix application

These flows are described in detail below.

## Request processing

Request processing starts with the integration broker. The integration broker passes business objects to the adapter for eMatrix, which communicates with the eMatrix collaboration server using its Java-based eMatrix ADK.Figure 1 on page 2provides an architectural overview of the data flow.

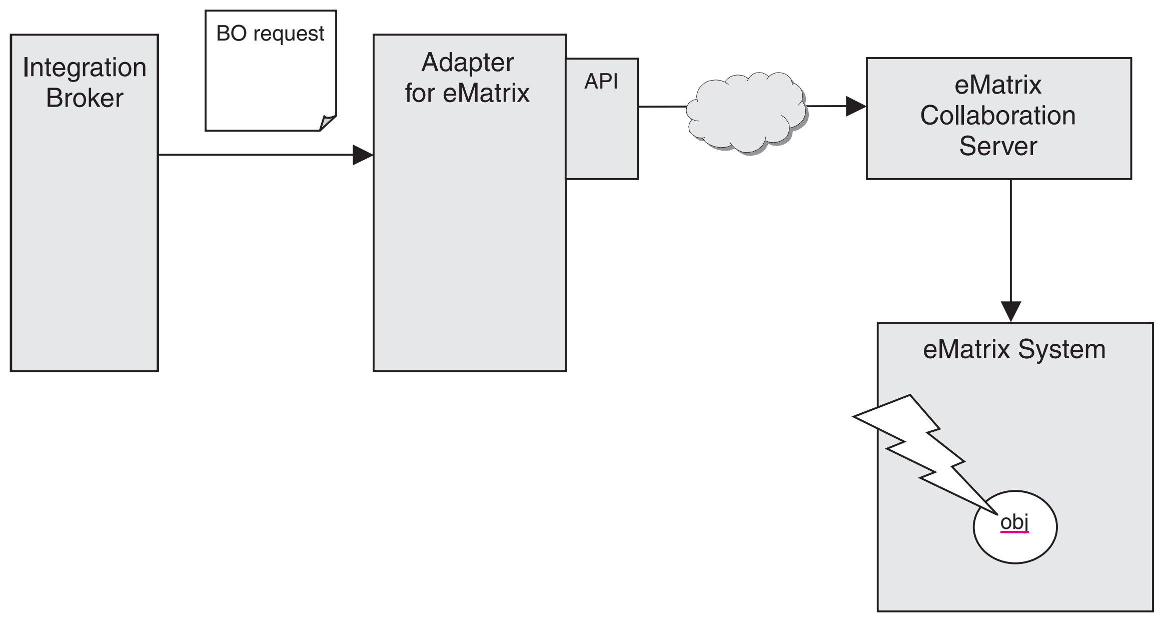


*Figure 1. Request processing in the eMatrix adapter*

Request processing takes place as follows.

1. The integration broker issues a request to the adapter in the form of a business object, which contains type information and a business object verb.
2. The adapter processes the business object by invoking the appropriate eMatrix ADK calls.

3. Each time an eMatrix ADK call is made, the collaboration server performs the specified action on the corresponding object in the eMatrix system and returns the result to the connector.
4. Once the adapter has finished processing the request, it returns a status code and if required, a business object with the resulting changes, to the integration broker.

## Event notification

The adapter for eMatrix uses event polling to detect events in the eMatrix system at run-time. Figure 2 on page 3 illustrates the sequence of events that occurs when the application is polled by the integration broker through the adapter.

**Note:** For polling to work, you must first configure an eMatrix trigger for the event or events you wish to monitor. This is done during the initial setup. For details, refer to “Configuring the adapter” on page 11..

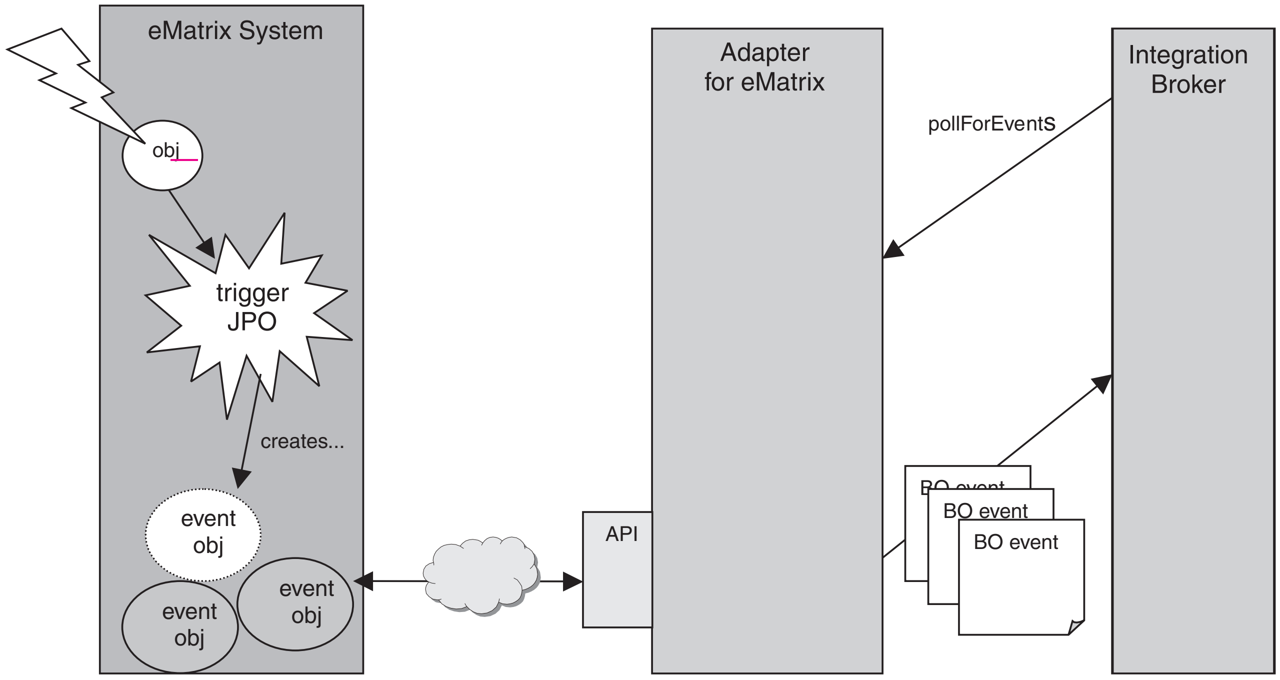


*Figure 2. Event notification in the eMatrix adapter*

Event notification takes place as follows.

1. An action is performed in the eMatrix system, causing a trigger to fire. Triggers can be invoked in many different ways, for example, such as modifying an eMatrix object with a configured trigger.
2. When the trigger is fired, the eMatrix system executes the associated Java Program Object (JPO), which creates a new event object in eMatrix.
3. At specified intervals, the adapter polls eMatrix for any new event objects that the JPO has created.
4. The adapter retrieves the event object.

5. It then retrieves the relevant eMatrix entities, translates them to business objects, and publishes them to the integration broker. The adapter can retrieve both eMatrix business objects and relationships.

## How the adapter works

The adapter for eMatrix is bi-directional. It can process events originating from the eMatrix system, and requests sent by the integration broker to the eMatrix system.

### Event notification

The adapter detects and processes events through a polling mechanism. This allows you, as the user, to set event priorities and control the frequency of event detections and the number of events that the adapter can process. With this level of control, you can balance the adapter load within the larger system and smooth the path of any critical events. These values are set in the adapter's configurable properties (see "Configuring the connector" on page 12).

The adapter makes use of Java Program Objects (JPOs) and eMatrix business object definitions to store and transmit information about events. IBM provides two default JPOs, which you may use if you choose. They are:

- The WBIEventLogger JPO, which creates the business object type that records events
- The InstallEventTables JPO, which creates the business object type that stores event

The text assumes that you are using the default WBIEventLogger JPO. For details on the structure of eMatrix objects, refer to Chapter 4, "Understanding business objects," on page 19.

The WBIEventLogger JPO is the crux of event notification. You can define triggers to invoke this JPO whenever any action is associated with a specific business object or relationship. The JPO uses the object ID and the specified options to create a new event business object or relationship in eMatrix.

For example, if you want to track events involving an object type called "Part Assembly", you can configure eMatrix to invoke the JPO when such actions occur. When the trigger fires, the JPO records the PartAssembly ID and the type of action taken during the event.

For details on defining triggers, refer to "Configuring the JPOs" on page 11..

**Note:** Event notification is an asynchronous process. The adapter does not support synchronous event notification.

### Event retrieval

When the adapter polls the application, it queries eMatrix for unprocessed events. It checks for all events that have a value of READY_FOR_POLL for the configuration property EventStatus. The adapter retrieves the event records that the WBIEventLogger JPO has created.

All events exist in the same vault. The adapter reads the EventVault property value in at startup.

The adapter then retrieves the associated eMatrix object and sends it to the integration broker. The adapter may also archive the event in eMatrix once it has been published to the integration broker.

Information about events is held in the eMatrix objects called “<*prefix*>_Event” for unprocessed events, and “<*prefix*>_Archived_Event” for processed events. The adapter uses status codes to track the state of events.

If the adapter terminates unexpectedly, it may report “in-doubt” events when it is restarted. An event is labeled in-doubt when the adapter is unable to determine whether it was successfully published to the integration broker or not. The adapter must process these events before it attends to any normal events. For details on what actions it takes, refer to “Processing errors” on page 39.

## Sample scenario

The adapter for eMatrix includes a sample scenario that illustrates how the adapter handles request processing and event notification. For information on installing, configuring and running this scenario, refer to Appendix C, “Sample scenario,” on page 77.

# Chapter 2. Installing the adapter

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

- "Adapter environment"
- "Overview of installation tasks" on page 8
- "Installing the adapter and related files" on page 8
- "Installed file structure" on page 8

## Adapter environment

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

- "Broker compatibility"
- "Adapter platforms"
- "Adapter dependencies" on page 8
- "Globalization" on page 8

### Broker compatibility

The adapter framework that an adapter uses must be compatible with the version of the integration broker (or brokers) with which the adapter is communicating. Version 2.1 of the adapter for eMatrix 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, versions 4.1.1, 4.2, 4.2.1, 4.2.2
  - WebSphere MQ Integrator, version 2.1.0
  - WebSphere MQ Integrator Broker, version 2.1.0
  - WebSphere Business Integration Message Broker, version 5.0
  - WebSphere Application Server Enterprise, version 5.0.2, with WebSphere Studio Application Developer Integration Edition, version 5.0.1

See *Release Notes* for any exceptions.

**Note:** For instructions on installing your integration broker and its prerequisites, see the following guides.
For WebSphere InterChange Server (ICS), see *IBM WebSphere InterChange Server System Installation Guide for UNIX* or *for Windows.*
For WebSphere message brokers, see *Implementing Adapters with WebSphere Message Brokers.*
For WebSphere Application Server, see *Implementing Adapters with WebSphere Application Server.*

### Adapter platforms

The adapter is supported on the following software.

**Operating systems:**

- AIX 4.3.3, AIX 5.1, AIX 5.2
- Solaris 7.0, Solaris 8.0
- HP UX 11.0, HP UX 11i
- Windows 2000

**Databases:**
- DB2 7.2
- Oracle 8.1.7.4, Oracle 9.2

**Third-party software:**
- eMatrix Server versions 9.6.01, 10.0.0.0
- eMatrix Collaboration Server versions 9.6.01, 10.0.0.0

### Adapter dependencies

The adapter client library/API resides in eMatrixServletRMI.jar version 10.0.0.0.

The adapter requires access to the eMatrix ADK at run time. The ADK is distributed with the eMatrix collaboration server and in a stand-alone version.

If you have the adapter installed on the same machine as the eMatrix collaboration server, you need not download the stand-alone version of the ADK. However, if the adapter and collaboration server are on two different machines, you must install the stand-alone ADK on the same machine as the adapter.

### Globalization

This adapter is DBCS (double-byte character set)-enabled and is translated.

## Overview of installation tasks

These are the steps you follow to install the adapter for eMatrix.
1. Make sure the integration broker is installed.
2. Make sure all the prerequisite software is installed.
3. Make sure you know the names and locations of any remote eMatrix servers.
4. Install the adapter for eMatrix and related files.
5. Configure the adapter for eMatrix.

The installation instructions in this guide cover steps 4 and 5.

## 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

Table 1 on page 9 shows the file structure used by the adapter and lists the files that are installed on the system.

**Notes:**

1. This document uses (\) backslashes as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes.
2. All product path names are relative to the directory where the product is installed on your system.

*Table 1. Adapter files installed on system*

| **Subdirectory of %*ProductDir*%** | **Description** |
|---|---|
| \bin\Data\App\eMatrixConnectorTemplate | Template for the eMatrix connector |
| connectors\eMatrix | Contains:<br>• CWeMatrix.jar<br>• start_eMatrix.bat |
| connectors\eMatrix\dependencies | Contains default JPOs:<br>• InstallEventTables.java<br>• WBIEventLogger.java |
| connectors\eMatrix\samples | Contains sample connector configuration files:<br>• EmatrixConnector.cfg<br>• PortConnector.cfg<br>• repos<br>• sound_card_update.bo |
| connectors\eMatrix\samples\repos | Contains sample business and relationship objects:<br>• wbi_bundle.xsd<br>• wbi_computer.xsd<br>• wbi_computer_bundle_to_computer.xsd<br>• wbi_computer_computer_to_pci_component.xsd<br>• wbi_computer_to_pci_component.xsd<br>• wbi_cpu.xsd<br>• wbi_cw_event.xsd<br>• wbi_expand_bo.xsd<br>• wbi_input.xsd<br>• wbi_network_card.xsd<br>• wbi_sound_card.xsd<br>• wbi_sound_card_sound_card_to_sound_chip.xsd<br>• wbi_sound_chip.xsd<br>• wbi_usb_2_0_card.xsd |
| connectors\messages | Contains the eMatrixConnector.txt file, which lists error and other messages. |
| ODA\eMatrix | Contains:<br>• eMatrixODA.jar<br>The Object Discovery Agent, which has the ODA code<br>• start_eMatrixODA.bat |
| ODA\messages | Contains the eMatrixODAAgent.txt file, which lists error and other messages. |

# Chapter 3. Configuring the adapter

This chapter describes how to configure and start the IBM WebSphere Business Integration adapter for eMatrix. It contains the following sections:

## Configuring the adapter

After installation and before startup, you must configure the adapter components as follows.

### Configuring the JPOs

The JPO triggers require certain configuration properties to be set before they will work. To set these configuration properties, you have to edit the source code for each JPO.

The JPOs installed with the adapter for eMatrix are WBIEventLogger and InstallEventTables. After you have installed them, you must locate each JPO in the eMatrix Business Modeler and edit the source code.

The fields that need to be edited are clearly marked. Their names and default values are listed in Table 2 on page 11.

*Table 2. JPO configuration properties*

| **Property name** | **Description** | **Default value** |
|---|---|---|
| wbiPrefix | The prefix used to provide all WebSphere Business Integration eMatrix meta-entities (types, attributes and policies) with their own namespace within eMatrix. It must match the value specified for the WBIPrefix connector configuration property. | wbi_ |
| vault | Name of the eMatrix vault in which new events will be created. Must match the value specified for the EventVault connector configuration property. | WBI_Events |
| adapterUser | The ID that the adapter uses when making requests in the eMatrix system. Must match the value specified for the ApplicationUserName connector configuration property. Important for preventing “Ping-pong” (see below). | adapter |

#### Preventing the ping-pong effect

When the adapter modifies an object in the eMatrix system for which a trigger is assigned, that trigger may in turn generate a new event to be published back to the broker. The following mechanism prevents this from happening.

When a JPO is invoked, it receives a Context object that contains, among other things, the userID associated with the user who invoked the JPO. This user ID is set in the JPO property adapterUser. When it is triggered, the WBIEventLogger JPO checks the user ID it receives from the eMatrix Context object.

If the user was the adapter for eMatrix, the value of the JPO's adapterUser property will match the value of the connector's ApplicationUserName property. In this case, the JPO will ignore the event, and it is not logged. If the user IDs do not match, the JPO creates a new event.

**Note:** It is important to set the adapterUser property in the JPO and the ApplicationUserName property in the adapter to the same values, otherwise you get the ping-pong effect.

## Configuring the connector

The connector component of the adapter has two types of configuration properties: standard configuration properties, which apply to most adapters, and application-specific configuration properties, which apply only to your adapter. You must set the values of these properties before running the connector.

### Standard configuration properties

To configure the standard connector properties, use the Connector Configurator tool. Details are given in Appendix B, "Connector Configurator," on page 61.. This tool provides a graphical user interface for configuring the connector. Click on the **Standard Config Properties** tab to add or modify configuration properties.

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

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

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

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

### Application-specific configuration properties

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

To configure these properties, use Connector Configurator. Click the **Application Config Properties** tab to add or modify configuration properties. For more information, see Appendix B, "Connector Configurator," on page 61.

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 eMatrix*

| Property | Description | Possible values | Default value | Required |
|---|---|---|---|---|
| ApplicationUserName | Name used when connecting to the eMatrix system. | <any> Must match the value set for the adapterUser JPO configuration property | WBI | Yes |
| ApplicationPassword | Password used when connecting to the eMatrix system. | <any> | WBI | Yes |
| EventVault | Vault in which the adapter searches for events. | <any> Must match the value set for the vault JPO configuration property | WBIEventVault | Yes |
| inDoubtEvents | Recovery strategy for in-doubt events. If no value is given, the adapter will fail on start-up. | One of: • FailOnStartup • Reprocess • LogError • Ignore | Reprocess | Yes |
| ArchiveProcessed | If True, the adapter archives all events. If False or undefined, the adapter deletes events from eMatrix after they have been processed. | true or false | false | |
| DefaultVault | Default vault in which the adapter creates new objects and manipulates existing ones. | <any> | DefaultWBIVault | |
| DefaultPolicy | Default policy used by the adapter when it creates new objects in eMatrix. | <any> | DefaultWBIPolicy | |
| KeepRelations | If true, the adapter maintains relationships when it updates business objects. If false, it destroys any relationships that are not defined in Update requests. | true or false | false | |
| UseDefaults | The adapter uses the defaults set in the business object definition if UseDefaults is set to true. | true or false | true | |
| PollQuantity | The maximum number of events that the adapter can process for each poll call. | <any positive integer> | 1 | |

*Table 3. Application-specific configuration properties for eMatrix (continued)*

| Property | Description | Possible values | Default value | Required |
|---|---|---|---|---|
| WBIPrefix | The prefix of all business objects and attributes related to event notification in the eMatrix system. **Note:** No pre-existing business objects or attributes may start with this value. | <any> Must match the value specified for the JPO wbiPrefix configuration property. | wbi_ | |
| EmatrixServer | The name of the remote business object server. | <any> | :bos | |
| Hostname | The URL of the collaboration server. | <any> | localhost:1099 | |

## Creating multiple connector instances

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

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

### Create a new directory

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

```
ProductDir\connectors\connectorInstance
```

where `connectorInstance` uniquely identifies the connector instance.

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

```
ProductDir\repository\connectorInstance
```

#### Create business object definitions

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

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

   ```
   ProductDir\repository\initialConnectorInstance
   ```

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

### Create a connector definition

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

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

### Create a start-up script

To create a startup script:

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

   dirname
2. Put this startup script in the connector directory you created in "Create a new directory" on page 14.
3. Create a startup script shortcut (Windows only).
4. Copy the initial connector's shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

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

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

## Starting the connector

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

```
ProductDir\connectors\connName
```

where *connName* identifies the connector. The name of the startup script depends on the operating-system platform, as Table 4 shows.

*Table 4. Startup scripts for a connector*

| Operating system | Startup script |
|---|---|
| UNIX-based systems | `connector_manager_connName` |
| Windows | `start_connName.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*.

## Starting the eMatrix adapter

The connector starts when the start_eMatrix.bat or start_eMatrix.sh script is run. These start files contain two variables that must be set before the adapter will run. They are:

- EMADK: This must be set to the location of the eMatrix ADK .jar file.
- EM_LIB: This must be set to the path of the directory containing the eMatrix shared library files.

For example:

```
set EMADK=c:\ematrix96\java\lib\eMatrixServletRMI.jar
set EM_LIB=c:\ematrix96\bin\winnt
```

**Note:** You do not need to set EM_LIB if you are running the eMatrix collaboration server on a separate machine and you have installed the stand-alone eMatrix ADK .jar file on it.

## 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.

## Using log and trace files

The adapter components provide several levels of message logging and tracing.

The **connector** uses the adapter framework to log error, informational and trace messages. Error and informational messages are recorded in an external log file, and the trace messages and trace level (0 to 5) are recorded in a trace file. The default message file name is eMatrixConnector.txt.

Table 5 describes the type of messages logged at each tracing level.

*Table 5. Trace file tracing levels*

| Trace Level | Description |
|---|---|
| 0 | • Logs errors and fatal errors from the eMatrix ODA<br>• Logs warnings that require a system administrator’s attention<br>• Logs important messages from the application |
| 1 | Traces all entering and exiting messages for method |
| 2 | Traces the ODA’s properties and their values |
| 3 | Traces the names of all business objects |
| 4 | Traces business object properties and the values received |
| 5 | • Indicates the ODA initialization values for all of its properties<br>• Traces the business object definition dump |

You configure both the log and trace files names, as well as the trace level, in Connector Configurator. Refer to Appendix B, “Connector Configurator,” on page 61 for details.

The **ODA** has no logging capability; error messages are sent directly to the user interface. Trace files and the trace level are configured in Business Object Designer. The process is described in “Configure the agent” on page 31. The ODA trace levels are the same as the connector trace levels, shown in Table 5 on page 17 above.

For more details on error handling and troubleshooting, refer to Chapter 6, “Troubleshooting and error handling,” on page 39.

# Chapter 4. Understanding business objects

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

The chapter contains the following sections:

- "Defining metadata" on page 19
- "Overview of business object structure" on page 19
- "Business objects for eMatrix" on page 20
- "Generating business objects" on page 26

## Defining metadata

The adapter for eMatrix is metadata-driven. In the WebSphere business integration system, metadata is defined as application-specific information that describes its data structures. The metadata is used to construct business object and relationship definitions that 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. You can make changes to the business object definitions by editing them in Business Object Designer.

Application-specific metadata may represent business object types, relationships, their properties, or attributes. Actual data values for each business object are conveyed at run time. The data values are encapsulated in the business objects that are passed between adapter and broker.

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

**Note:** If you need to make changes to the business object structure, IBM recommends that you make them to the corresponding object in eMatrix, using the ODA. Refer to Chapter 5, "Generating business object definitions," on page 29 for details.

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

## Overview of business object structure

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

- A type name
- Supported verbs
- Attributes
- Application-specific information (ASI)

An application-specific business object definition is a particular type 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. For instance, attributes may reference child WebSphere Business Integration business objects that represent relationships.

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.

Hierarchical business objects can contain child business objects that have single or multiple cardinality. The child objects are stored in their parents' attributes. Attributes that contain child business objects with multiple cardinality are arrays of business objects. Attributes that represent child business objects with single cardinality contain only one such object.

## Business objects for eMatrix

Valid application-specific business objects in the WebSphere Business Integration system model the essential components of the eMatrix system. The following eMatrix components can be modelled as WebSphere Business Integration business objects:

- **Business object types**:
  Used to organize information. Instances of business objects store information.
- **Relationships**: Defines a relationship between two business objects
- **Commands**:
  Used to perform custom operations or invoke eMatrix programs

A WBIA business object definition specifies the components it represents in the `ematrix_class` value of its application-specific information (ASI), defined at the business object level. The three valid`ematrix_class` values, corresponding to the eMatrix components listed above, are discussed in the next section.

### Business object types

eMatrix business object types are represented in the WBIA system by business object definitions with business object-level ASI that specify an`ematrix_class` with the value of `business_object`. The ASI must also contain an `object_type` parameter that specifies the name of the eMatrix business object type that this particular WBIA business object will model. For example:

```
ematrix_class=business_object; object_type=<name of eMatrix business object type>
```

The eMatrix business object types, and hence their business object definitions in the WBIA system, contain two sets of fields. They are:

- Properties, which are fixed and shared by all eMatrix business object types. They include business object owner, policy and revision.
- Attributes, which are defined by the user (not to be confused with WBIA system attributes).

The eMatrix properties available to you through the business object are listed in Table 6 on page 21,.

**Note:** Use an attribute of type "String" to represent a date. For supported date formats, refer to the eMatrix documentation.

*Table 6. eMatrix business object properties*

| Property | Description | Expected attribute type |
|---|---|---|
| object_id | Unique ID for eMatrix object | String |
| Name | Name of eMatrix object | String |
| Revision | eMatrix object revision number | String |
| Policy | The governing policy for the eMatrix object | String |
| Owner | The owner of the eMatrix object | String |
| State | The current state of the eMatrix object | String |
| Vault | The eMatrix vault that contains this object | String |
| to_relationship | One or more "to" eMatrix relationships that this object supports | Child object modeling an eMatrix relationship Cardinality = 1...n |
| from_relationship | One or more "from" eMatrix relationships that this object supports | Child object modeling an eMatrix relationship Cardinality = 1...n |

eMatrix lets the user identify objects either through an object ID or through a combination of type, name and revision. Therefore, all WBIA business objects must contain either:

- An attribute that maps to the property `object_id`, or
- Two attributes, mapping to the properties `name` and `revision`, respectively.

These will be key values in the business object definition. For illustrations, refer to "Business object definition examples" on page 21.

To specify that a particular WBIA business object attribute should be mapped to an eMatrix attribute:

1. Define the attribute as a simple type, such as a string or integer
2. Include

   ```
   attr=<eMatrix attribute name>
   ```

   in the business object definition attribute-level ASI.

## Business object definition examples

Table 7 on page 21 shows a business object definition that includes two user-defined eMatrix attributes, `PartNumber` and `Size`, and maps to the ASI property `object_id`.

*Table 7. Business object definition #1*

| Attribute name | Type | Default value | Application-specific information | IsKey? |
|---|---|---|---|---|
| ObjectId | String |  | prop=object_id | Yes |
| State | String |  | prop=state | No |
| PartNumber | String | ASX31 | attr=PartNumber | No |
| Size | Integer | 15 | attr=Size | No |

Table 8 on page 22shows the same business object definition, but it now maps to the eMatrix properties, `name` and `revision`.

*Table 8. Business object definition #2*

| Attribute name | Type | Default value | Application-specific information | IsKey? |
|---|---|---|---|---|
| Name | String | AllSeasonX31 | prop=name | Yes |
| Revision | String | 1 | prop=revision | Yes |
| State | String | | prop=state | No |
| PartNumber | String | ASX31 | attr=PartNumber | No |
| Size | Integer | 15 | attr=Size | No |

## Relationships

eMatrix relationships are represented in the WBIA system by business object definitions with business object-level ASI that specify an `ematrix_class` with the value of `relationship`. The ASI must also contain an `object_type` parameter that specifies the name of the eMatrix relationship that this particular WBIA business object will model. For example:

```
ematrix_class=relationship; object_type=<name of eMatrix relationship type>
```

Like the business object definitions described above, the business object definitions that reflect eMatrix relationships may contain a combination of eMatrix properties (fixed) and attributes (user-defined).

Table 9 on page 22 lists the relationship properties.

*Table 9. eMatrix relationship object properties*

| Property | Description | Expected attribute type |
|---|---|---|
| relationship_id | Unique ID for eMatrix relationship | String |
| to_object | eMatrix object to which this relationship extends | Child object modeling an eMatrix business object Cardinality = 1 |
| from_object | eMatrix object from which this relationship extends | Child object modeling an eMatrix business object Cardinality = 1 |

All relationship business object definitions must contain:

- An attribute that maps to the property `relationship_id`
- Instances of`to_object` or `from_object` attributes (see Note below)
- Attributes with application-specific information categorizing them as `to_object` or `from_object` child business object containers.

**Note:** A relationship object may contain many instances of the `to_objectorfrom_objectattribute`, as a single relationship can support many object types. However, if the relationship is defined as a `to_relationship` in the parent business object, then the relationship object may contain only `to_object` identifiers. Likewise, if a `from_relationship` is defined in the parent business object, the relationship object may contain only `from_object` identifiers.

The relationship definition takes this form:

```
--Business_Object_A
      | -- to_relationship = Relationship_A_to_B
                      | -- to_object = Business_Object_B
```

or

```
--Business_Object_A
      | -- from_relationship = Relationship_A_to_B
                      | -- from_object = Business_Object_B
```

See “Relationship examples” on page 23for an illustration.

## Defining relationships

The adapter can only recognize relationships between different objects if you define these relationships explicitly in the business object definition. When business objects are related, the adapter requires a relationship object to connect the business objects. If business objects are not connected by a relationship object, they are assumed to be unrelated.

For example, if you define this business object:

```
--Business_Object_A
      | -- to_relationship = Relationship_A_to_B
                      | -- to_object = Business_Object_B
```

and then perform a Retrieve on the parent business object (object A), the adapter will retrieve A and **only** those objects of type B that are related to A through the A-to-B relationship.

However, if you define this business object:

```
--Business_Object_D
      | -- Business_Object_E
```

and then perform a Retrieve on the parent business object (object D), the adapter will retrieve D and objects of type E regardless of whether they are related to D. In this case, business object E must have attributes containing valid search criteria, or you will get an error.

You can combine related and unrelated business objects in a definition, depending on the business scenario and relationships. For example, you can define this business object:

```
--Business_Object_A
      | -- to_relationship = Relationship_A_to_B
                      | -- to_object = Business_Object_B
                    | -- Business_Object_D
```

Here, business objects A and B are related to each other, but business object D is not related to either of them.

## Relationship examples

The two examples shown below are related in the same way, though the relationship is defined differently in each business object definition.

```
--Car
      | -- to_relationship = CarToTireRelationship
                      | -- to_object = Tire
```

```
--Tire
      | -- from_relationship = CarToTireRelationship
            | -- from_object = Car
```

## Command objects

eMatrix commands are represented in the WBIA system by business object definitions with business object-level ASI that specify an `ematrix_class` with the value of `command`. For example:

```
ematrix_class=command
```

Command business object definitions are structured as follows:

```
-- Command_object
        | -- Input_Object
        | -- Output_Object
```

The input and output objects are child objects of the top-level command. The input object definition, which encapsulates the command string and its parameters, is required. However, the output object definition, which encapsulates the output returned by the command, is optional.

The input child object must contain the following attributes:

- `type=input`

  which defines it to the adapter as the input
- `command=<`*`command string to run in eMatrix`*`>`

  You can use the command string to refer to any attribute in the child object by enclosing the attribute name in percent signs (% <*name*>%). See the example below.

If a command returns results, the object must have an output attribute defined to receive the data. The output attribute may be a string or a child object. If you use a child object as an output attribute, you must specify a data handler to enable the adapter to map the incoming data. To specify a data handler, you can use one of these attribute pairs:

- dh_mo=<*data handler meta-object name*>;dh_mimetype=<*mimetype*>
  or
- dh_classname=<*data handler classname*>

See "Command examples" for an illustration.

A command object hierarchy cannot contain or be contained in a hierarchy containing a collection of eMatrix business objects and relationships. It must stand on its own.

**Note:** The only verb supported for command business objects is Create.

### Command examples

Here is an example of a command object definition.

```
--Command_Object
      | -- Input_Object (ASI = 'type=input;command=approve %ObjectId% signature
 %signature% comment %Comment%'
                      | -- ObjectId = 1000
            | -- Signature = Loan Approved
            | -- Comment = Approved up to $20,000
```

The actual command processed in eMatrix, after the adapter had made its substitutions, would be:

```
approve 1000 signature Loan Approved comment Approve up to $20,000
```

Here is an example of a command object that executes a program that returns XML.

```
--Command_Object
      | -- Input_Object (ASI = 'type=input;command=exec program BuildReport
 %ObjectIde%'
                      | -- ObjectId = 1000
      | -- Output_Object (ASI =
'type=output;dh_mo=Default_DataHandler_MO:dh_mimetype=text/xml'
                      | -- AttrA
            | -- AttrB
```

## Business object verbs

Every business object contains a verb, which describes how the data in the application-specific business object should be handled by the eMatrix adapter.

The adapter identifies a business object using either its unique objectId or a combination of type, name and revision. These are all user-defined properties. For details, see “Business object types” on page 20.

The adapter interprets the same verb differently for different classes of eMatrix objects. It checks the value of `ematrix_class` in the business object definition before processing the business object.

Table 10 on page 25 lists the business object verbs available to the adapter for eMatrix and describes the actions that the adapter performs with each one.

*Table 10. Business object verbs and actions*

| Verb | eMatrix class | Action |
|---|---|---|
| Create | Business objects | Creates one or more new business objects in eMatrix, along with any relationships. Uses name, type and revision specified in the request. Uses vault and policy values if specified, otherwise uses the default.<br><br>Creates any specified child objects and relationships. |
| | Relationships | Cannot create relationships independently, because all child objects must have the same verb as the parent object. To create a new relationship between objects with an already existing relationship, use Update. |
| | Commands | Executes the command in the eMatrix system and returns any results to the request object. |
| Update | Business objects and relationships | Updates one or more specified business objects and their relationships. For an example, see below. |
| Delete | Business objects | Deletes the specified business object and automatically deletes any relationships bound to it. All deletes are physical deletes. |
| | Relationships | Deletes the relationship denoted by the relationship ID. Disconnects the business objects bound to the relationship; does not delete them. |
| Retrieve | Business objects and relationships | Retrieves the entire business object from eMatrix, including relationships and any eMatrix business objects that correspond to attributes in the WBIA business object.<br><br>The business object returned to the integration broker should accurately reflect the structure in the business object definition. |
| RetrieveBy Content | Business objects and relationships | Retrieves the entire business object from eMatrix, using key and/or non-key information. If it finds more than one match, it returns the first one and sends a message saying “Multiple hits”. |
| Exists | Business objects and relationships | Verifies whether the top-level business object exists in eMatrix. |

The verb is contained within the application-specific business object. Its value is set by the broker that creates the business object. The adapter will perform different operations on the eMatrix application, depending on the verb and the contents of the business object.

Once the adapter has processed the business object, and invoked the requisite actions in the eMatrix system, it returns a status code. It may also return a new business object containing information about the state of the application to the integration broker.

### Attribute properties

The WBIA business object definitions have various properties that you can set on their business object attributes. The tables below list the attributes and their values.

Table 11 lists simple attributes.

*Table 11. Simple attributes*

| **Attribute** | **Property** |
|---|---|
| Name | Specifies the business object field name. |
| Type | Specifies the business object field type. |
| MaxLength | 255 by default. |
| IsKey | Each business object must have at least one key attribute, which you specify by setting the key property to true for an attribute. |
| IsForeighKey | Not used. |
| Is Required | Not used. |
| AppSpecInfo | Stores the application-specific information (ASI), such as attr=<attribute name>. |
| DefaultValue | Specifies a default value that the connector uses for a simple attribute in the inbound business object if the attribute is not set and is a required attribute. |

### Application-specific information

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

Application-specific information can be specified on the business object, on each business object attribute, and for each verb. For simple attributes, the ASI specifies either the type of the eMatrix attribute or the name of the eMatrix property (such as name, revision, or vault). For child business objects, the ASI specifies the direction of the relationship (for object hierarchies representing eMatrix business objects and relationships).

## Generating business objects

Business Object Designer provides a graphical interface that enables you to generate business object definitions for use at run time. For details, see Chapter 5, “Generating business object definitions,” on page 29

If the object model in eMatrix is changed, use the ODA in Business Object Designer to modify the original business object definition to reflect the update, or to create a new definition.

# Chapter 5. Generating business object definitions

This chapter describes the Object Discovery Agent (ODA) for eMatrix, and how to use it to generate business object definitions for the IBM WebSphere Business Integration Adapter for eMatrix.

This chapter contains the following sections:

## Overview of the ODA for eMatrix

An ODA (Object Discovery Agent) enables you to generate application-specific business object definitions. A business object definition is a template for a business object. The ODA examines specified application objects, "discovers" the elements of those objects that correspond to business object attributes, and generates business object definitions to represent the information. Business Object Designer provides a graphical interface to access the Object Discovery Agent and to work with it interactively.

The Object Discovery Agent (ODA) for eMatrix supports both business object and relationship types. The ODA connects to the eMatrix database using the initialization properties that you provide in Business Object Designer.

It then retrieves all the business object type names from the eMatrix database and presents them in Business Object Designer (see xxx). You can expand any of the business object types to see the relationships associated with them, and you can also expand these relationships. You can then select one or more of these business object types (parent or child) and relationship types and generate a business object definition for it.

The Business Object Designer wizard automates the process of creating these definitions, using the eMatrix business object attributes retrieved by the ODA. Business object definitions created for child business object and relationship types are associated with the parent business object definition. You can view or make modifications to a business object definition before you save it to the server.

You use the ODA to generate business object definitions at two different stages:

1. **When the system is first set up:** You must create a set of business object definitions for all the events defined within eMatrix.
2. **Whenever an object or relationship type used in an event is modified in eMatrix:** You must run the ODA in Business Object Designer to pick up the changed metadata for the object and export a new business object definition to the broker repository.

## Generating business object definitions

This section describes how to use eMatrix ODA in Business Object Designer to generate business object definitions. For detailed information on launching and using Business Object Designer, see *IBM WebSphere Business Integration Adapters Business Object Development Guide*.

### Before you begin

The eMatrix RMI collaboration server must be installed before you can run the eMatrix ODA. Once the collaboration server is installed, you must run the RMI daemon for the ODA to connect to the server.

You can run the ODA and the server on different machines. If you do, you need to know the hostname, port and name of the collaboration server on the remote machine.

### Starting the ODA

The ODA for eMatrix has a default name of eMatrixODA. The name can be changed by changing the value of the AGENTNAME variable in the start script.

- To start the ODA from Windows, select:

  **Start>Programs>IBM WebSphere Business Integration Adapters Adapters>Object Discovery Agent>eMatrix Object Discovery Agent**
- To start the ODA from the command line, run this command:

  ```
  start_eMatrixODA
  ```

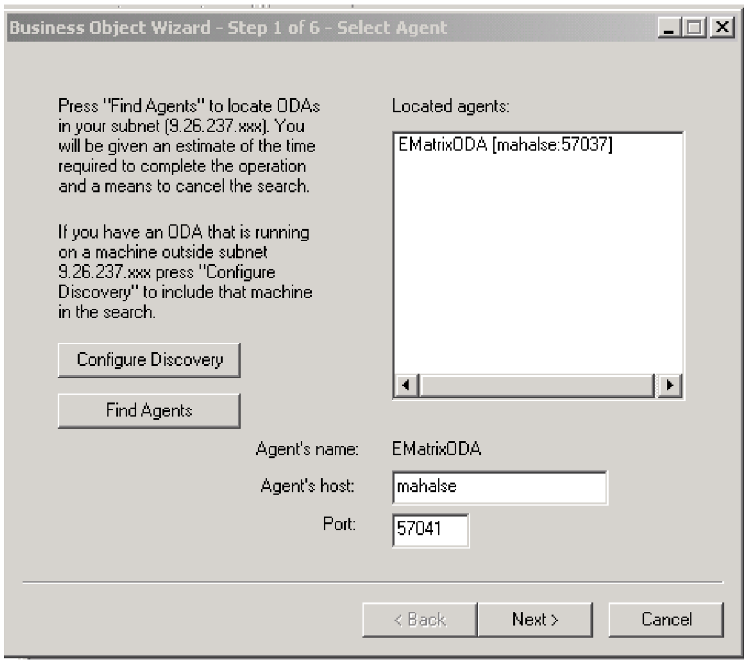
### Running Business Object Designer

Business Object Designer provides a wizard that guides you through the steps to generate a business object definition using the ODA. The steps are as follows:

#### Start Business Object Designer

To run Business Object Designer from Windows:

1. Select **Start>Programs>IBM WebSphere Business Integration Adapters> Tools>Business Object Designer**
2. The Business Object Designer main window opens.
3. To choose an ODA, Select **File > New Using ODA**. The *Business Object Wizard - Step 1 of 6 - Select Agent* screen appears (see Figure 3 on page 31).



Press "Find Agents" to locate ODAs in your subnet (9.26.237.xxx). You will be given an estimate of the time required to complete the operation and a means to cancel the search.

If you have an ODA that is running on a machine outside subnet 9.26.237.xxx press "Configure Discovery" to include that machine in the search.

Located agents:
EMatrixODA [mahalse:57037]

Agent's name: EMatrixODA
Agent's host: mahalse
Port: 57041

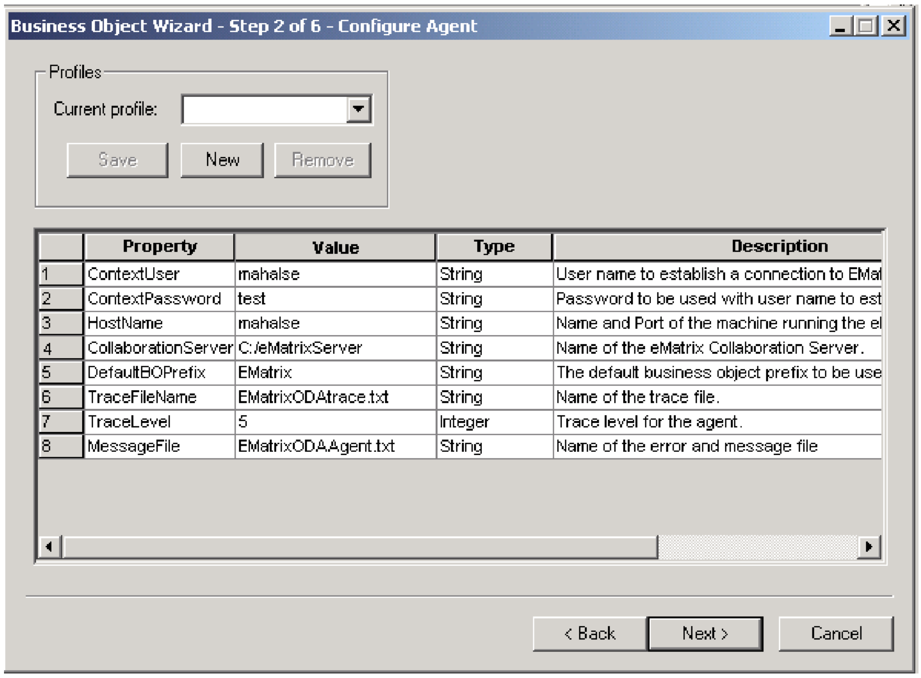
*Figure 3. Select Agent screen*

4. Click **Find Agents**. The system finds ODAs running on your system, including eMatrixODA, and lists them in the **Located agents:** window.
5. Select eMatrixODA and click **Next**.

## Configure the agent

The *Business Object Wizard - Step 2 of 6 - Configure Agent* screen appears.

In this screen you enter the information that the ODA needs to establish a context to communicate with the eMatrix database. The required properties and their values are shown in Table 12 on page 32.



| | Property | Value | Type | Description |
|---|---|---|---|---|
| 1 | ContextUser | mahalse | String | User name to establish a connection to EMat |
| 2 | ContextPassword | test | String | Password to be used with user name to est |
| 3 | HostName | mahalse | String | Name and Port of the machine running the el |
| 4 | CollaborationServer | C:/eMatrixServer | String | Name of the eMatrix Collaboration Server. |
| 5 | DefaultBOPrefix | EMatrix | String | The default business object prefix to be use |
| 6 | TraceFileName | EMatrixODAtrace.txt | String | Name of the trace file. |
| 7 | TraceLevel | 5 | Integer | Trace level for the agent. |
| 8 | MessageFile | EMatrixODAAgent.txt | String | Name of the error and message file |

*Figure 4. Configure Agent screen*

1. Use the Profiles menu to create a new profile. You may also select an existing profile.
2. Type the name of each property, its value, type and description.

   **Note:** If you use a profile, the property values are filled in for you.

*Table 12. Property values for configuring the ODA*

| Property name | Property type | Default value | Description | Required |
|---|---|---|---|---|
| ContextUser | String | Creator | The user name that is used to establish a context object to connect to the eMatrix database. | Yes |
| ContextPassword | String | None | The password that is used to establish a context object to connect to the eMatrix database. May be left empty. | Yes |
| HostName | String | | The host name and port for the eMatrix collaboration server. For example; `katari:1099` | Yes |
| CollaborationServer | String | | The name of the eMatrix collaboration server. For example; `:bos` | Yes |
| DefaultBOPrefix | String | EMatrix | The default prefix for the business object definition. | Nos |
| TraceFileName | String | <agentname> Trace.txt | The name of the trace file. | No |
| TraceLevel | Integer | 5 | Trace level enabled for the ODA. | Yes |
| MessageFile | String | <agentname> Agent.txt | The name of the error and message file. | Yes |

All messages displayed by the eMatrix ODA appear in the `eMatrixODAAgent.txt` file, which has a standard message file format.

**Note:** If the name of the message file is not correctly specified, the ODA will run without messages.

For more information on the range of trace levels and their meaning, refer to

**Creating a Profile**

You can also can save all the values you enter on this screen to a profile. Instead of re-typing all the data the next time you run the ODA, you simply select a profile from the drop-down menu and use the saved values.

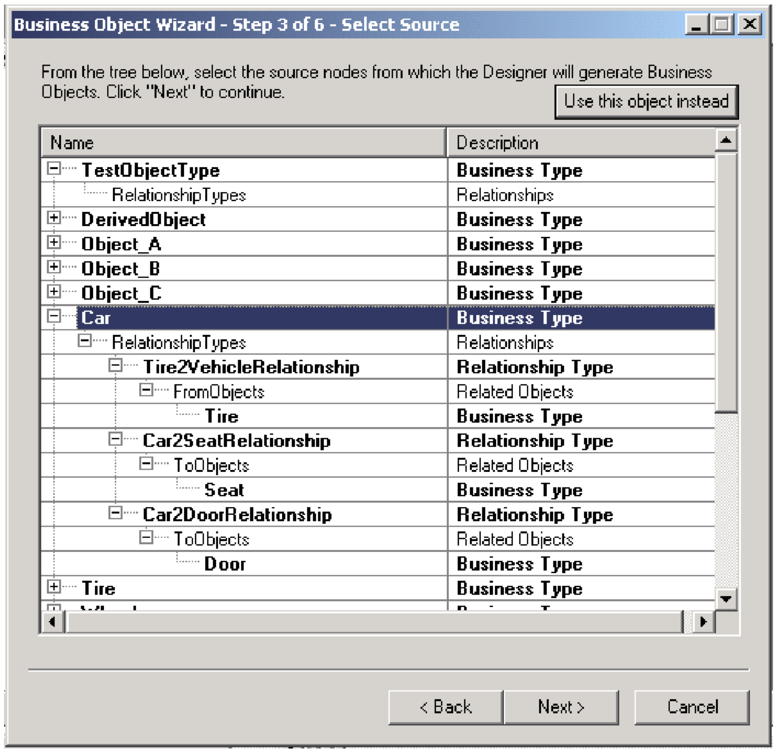
You can save multiple profiles, each with a different set of specified values for the same properties.

When you have finished, click **Next**.

## Select a business object

The *Business Object Wizard - Step 3 of 6 - Select Source* screen appears. Use this screen to select any number of business object or relationship types for which the ODA will generate business object definitions.

The screen lists the types that have been defined in eMatrix. If eMatrix has not been initialized, the list will be empty.



From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

| Name | Description |
|---|---|
| TestObjectType | Business Type |
| RelationshipTypes | Relationships |
| DerivedObject | Business Type |
| Object_A | Business Type |
| Object_B | Business Type |
| Object_C | Business Type |
| Car | Business Type |
| RelationshipTypes | Relationships |
| Tire2VehicleRelationship | Relationship Type |
| FromObjects | Related Objects |
| Tire | Business Type |
| Car2SeatRelationship | Relationship Type |
| ToObjects | Related Objects |
| Seat | Business Type |
| Car2DoorRelationship | Relationship Type |
| ToObjects | Related Objects |
| Door | Business Type |
| Tire | Business Type |

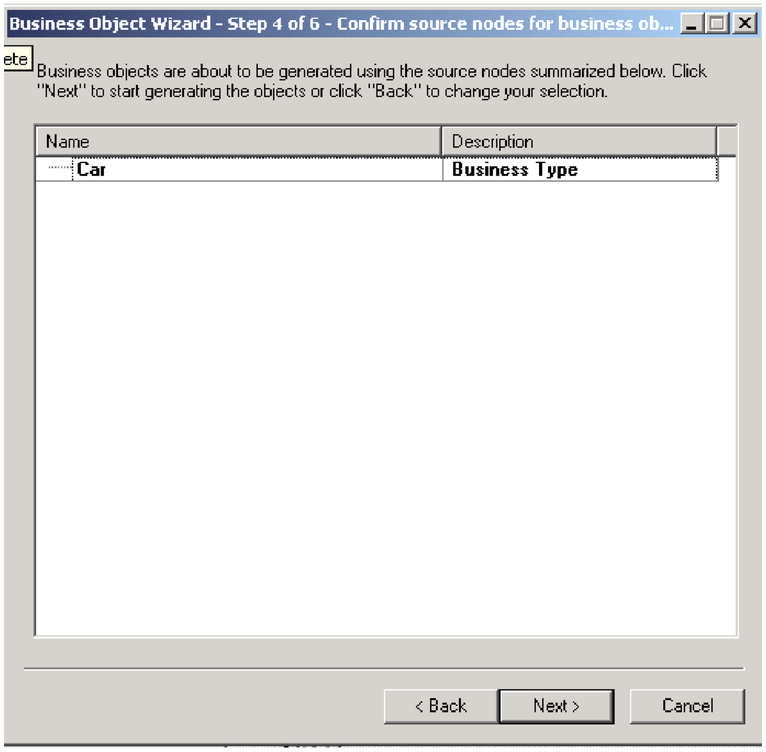
Figure 5. Select Source screen

- Expand a parent business object to see a list of its relationship types.
- Expand a relationship type to see a list of the business objects to or from which it extends.

1. Select the business object types you want to use.
2. Click **Next**.

## Confirm selection

The *Business Object Wizard - Step 4 of 6 - Confirm source nodes for business object definitions* screen appears. It shows the source nodes you selected.



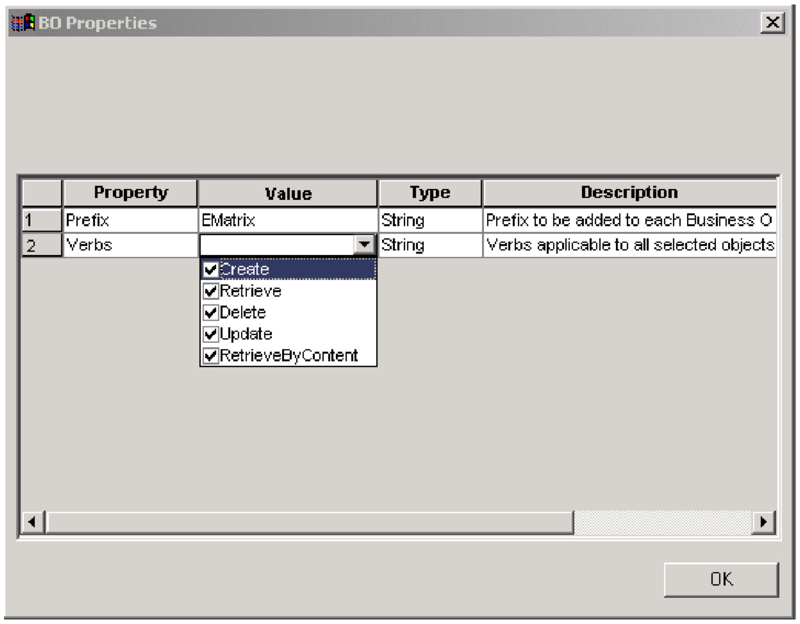
Business Object Wizard - Step 4 of 6 - Confirm source nodes for business ob...

Business objects are about to be generated using the source nodes summarized below. Click "Next" to start generating the objects or click "Back" to change your selection.

| Name | Description |
|---|---|
| Car | Business Type |

Figure 6. Confirm source nodes screen

Click **Back** to make changes or **Next** to confirm the list is correct.



|  | Property | Value | Type | Description |
|---|---|---|---|---|
| 1 | Prefix | EMatrix | String | Prefix to be added to each Business O |
| 2 | Verbs |  | String | Verbs applicable to all selected objects |

- ✓ Create
- ✓ Retrieve
- ✓ Delete
- ✓ Update
- ✓ RetrieveByContent

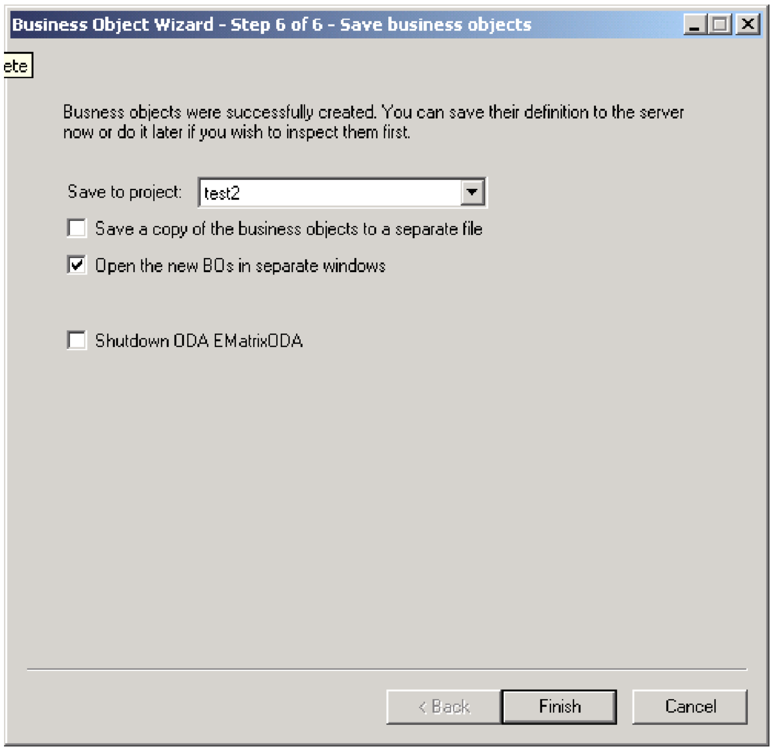
*Figure 7. Generating business objects screen*

The next screen requires you to set the business object properties. Here you can select the business object prefix and the verb or verbs that the business objects will use. Deselect any of the verb values that you do not want your business objects to use.

Click **OK** to move on.

## Save the business object definition

The *Business Object Wizard - Step 5 of 6 - Save business objects* screen appears.


Business Object Wizard - Step 6 of 6 - Save business objects
ete
< Back
Finish
Cancel

Busness objects were successfully created. You can save their definition to the server now or do it later if you wish to inspect them first.

Save to project: test2

☐ Save a copy of the business objects to a separate file

☑ Open the new BOs in separate windows

☐ Shutdown ODA EMatrixODA

*Figure 8. Save business objects screen*

Here you choose the location to which your business object definition will be saved.

You can optionally save the generated business object definitions to a file. To do so:

1. Check **Save business objects to a file**. A dialog box appears.
2. Type the location in which you want the copy of the new business object definitions to be saved.

Business Object Designer saves the files to the specified location.

**Note:** If you create new custom objects in eMatrix, you must add new business object definitions for them, using the ODA as described in this chapter.

If you have finished working with the ODA, you can shut it down by checking **Shutdown ODA: eMatrix ODA** before clicking **Finish**.

## Uploading your files

The newly created business object definition files must be uploaded to the integration broker once they have been created. The process depends on whether you are running ICS, WebSphere message brokers, or WAS.

For ICS: If you have saved your business object definition files to a local machine and need to upload them to the repository on the server, refer to the *Implementation Guide for WebSphere InterChange Server*.

For WebSphere message brokers: You must export the business object definitions out of Business Object Designer and into the integration broker. For details, refer to *Implementing Adapters for WebSphere MQ Integrator Broker.*

For WAS: You must export the business object definitions out of Business Object Designer and into the integration broker. For details, refer to *Implementing Adapters for WebSphere Application Server.*

# Chapter 6. Troubleshooting and error handling

This chapter describes how the adapter handles some common errors and offers advice on troubleshooting common problems.

This chapter covers:
- "Processing errors" on page 39
- "Troubleshooting" on page 40

## Processing errors

In general, whenever the adapter encounters an error, it logs an error message through the connector API.

All error messages are stored in an external message file, as described in "Using log and trace files" on page 17.

For all operations, if a connection cannot be established to eMatrix or if the connection is lost during processing, the adapter returns the code APPRESPONSETIMEOUT and logs a fatal error. This causes the connector to be terminated and an e-mail notification is sent to the administrator.

### Unresolved events

The adapter takes action to recover from unresolved or incomplete events depending on the value of the configuration property inDoubtEvents.

The options are listed in Table 13.

*Table 13. Processing in-doubt events*

| Value of "inDoubtEvents" | Action to take |
|---|---|
| FailOnStartUp | Log a fatal error and terminate. |
| Reprocess | Publish the event to the broker like a normal event. |
| LogError | Ignore the event and log an error. |
| Ignore | Ignore the event. |

### Polling errors

Several errors may occur in the polling process. The most common are listed in Table 14.

*Table 14. Processing polling errors*

| Error | Action |
|---|---|
| Cannot publish event because it is not subscribed. | Logs an error message, may archive the event, and continues processing events. |
| Cannot publish event because connector controller is not active. | Sets event status to 0 and returns SUCCESS code. |
| Error is specific to this event, for example, invalid value specified in event. | Logs an error message, may archive the event, and continues processing events. |

*Table 14. Processing polling errors (continued)*

| Error | Action |
|---|---|
| Broker rejects event with an error message. | Logs an error message, may archive the event and the error message, and returns FAIL code. |
| Cannot communicate with eMatrix; other unrecoverable errors | Logs a fatal error messages and returns APPRESPONSETIMEOUT code |

## Verb processing errors

The adapter requires a unique business object ID or combination of values (type, name and revision) to identify a business object. If the values that are supplied are incorrect or do not match any existing business object, an error is reported.

These are some of the most common verb processing errors. For a full list, refer to the *IBM WebSphere Business Integration Connector Development Guide for Java* or *for C++*.

- When creating business objects, if the adapter is successful, it posts the newly created business object to the integration broker and returns the value SUCCEED.
  If a business object with the specified values already exists, the adapter returns VALDUPES and does not create the new business object.
- When updating business objects, if the adapter is successful, it posts the newly updated business object to the integration broker and returns the value VALCHANGE.
- If any error occurs during processing, the adapter returns the FAIL code to the broker, and logs an explanatory error message.

## Troubleshooting

This sections describes some simple troubleshooting techniques for common problems you may encounter when using the adapter for eMatrix.

## Event notification

This section describes four common problems that may arise during event notification. They are:

- Faulty event generation
- Incorrect eMatrix event business object names
- The event vault not found
- Incorrect business object hierarchies

### Checking for event generation

When you debug the event notification functionality, you will first want to check whether you are successfully generating any events. To do this:

1. Perform an action that you know will invoke a trigger which calls WBIEventLogger.
2. Next, perform a query to see if the event was generated; for example:
   ```
   temp query bus wbi_Event **;
   ```
3. If no events are being generated, make sure you have installed a trigger that will invoke WBIEventLogger.
   You can check whether a trigger was installed by opening up the appropriate

business object type or relationship definition in the eMatrix Business Modeler. See Appendix C, “Sample scenario,” on page 77 for an example of how to install an event notification trigger.

**Note:** No events are generated when the user name of the user whose actions invoke the trigger matches the `adapterUser` constant specified in WBIEventLogger. The `adapterUser` constant specifies the user name that the adapter should be using to log in to the eMatrix system.
The point of ignoring such events is to prevent situations where the adapter modifies an entity in the eMatrix system and thereby causes an event to be logged. We assume that the adapter is aware of its own activities and does not need to log them. Keep this in mind when testing.

4. If you wish to test whether a trigger that executes WBIEventLogger is being invoked, set the following property in the matrix.ini file:

   ```
   MX_MQL_TRACE=TRUE
   ```

   When you perform an action which invokes the trigger you’ll see a message similar to this:

   ```
   18:15:00.826 MQL t@2900 Session: mx3f20221364f87ac7
   18:15:00.826 MQL t@2900 Program: WBIEventLogger
   18:15:00.826 MQL t@2900 args: -method recordEvent ${OBJECTID} n=wbi_cpu
   v=Update p=1
   18:15:08.417 MQL t@2900 End Program:MQL<3>
   ```

   This tells you that the trigger has executed WBIEventLogger with the arguments listed above.

## Checking eMatrix event business object names

If you know that events are being generated, but you still get an error every time you poll the eMatrix application, take a closer look at the name of the event business object. The business object name should look something like this:

```
wbi_22032.41802.50936.705_1058897313350_1892833611
```

If you notice upper-case words, such as FROMOBJECT, appearing in business object names, you may have specified an invalid MARCO value as an argument to the trigger. To resolve this:

1. Check all business object type definitions in the Business Modeler.
2. Review the triggers they use to invoke WBIEventLogger and make sure any MARCO values you provide as arguments are correct.

## Finding the event vault

If you receive the following error during event polling, it means the adapter cannot find the EventVault referenced in your connector properties. The error is:

```
System Error: #1600039: vault 'WBI_Events' does not exist
```

You may get this error for one of two reasons:

- You are referencing the wrong vault in your properties, or
- You have not created the vault.

Run InstallEventTables to make sure your vault was created, and check the name of EventVault in the connector properties.

## Checking business object hierarchies

When you create a business object hierarchy that will be used for event notification, make sure the hierarchy does not include any child business object definitions that represent an eMatrix business object type and whose parent also represents an eMatrix business object type.

If such an object exists, the adapter will fail when polling and log a message similar to this:

```
An error occurred while trying to validate your WBI business object
instance hierarchy.
```

The reason for this is that business objects which represent eMatrix business objects cannot contain attributes that also represent eMatrix business objects but are null.

## Request processing

This section describes two common problems that may arise during request processing. They are:

- Incorrect string values in commands
- Incorrect hierarchical business object values

### Processing strings in commands

The values of string-typed attributes of business objects representing eMatrix commands are not quoted when they are interpolated into the command template prior to command processing. Therefore, an unquoted attribute value containing white space will be interpreted as two words of an MQL command.

### Values in business object hierarchies

When sending a request, you may have child business objects in the hierarchy representing eMatrix business objects that have parent objects also representing eMatrix business objects. Make sure that any child business object is not null. The adapter will throw an error if such a hierarchy is encountered.

## General tips

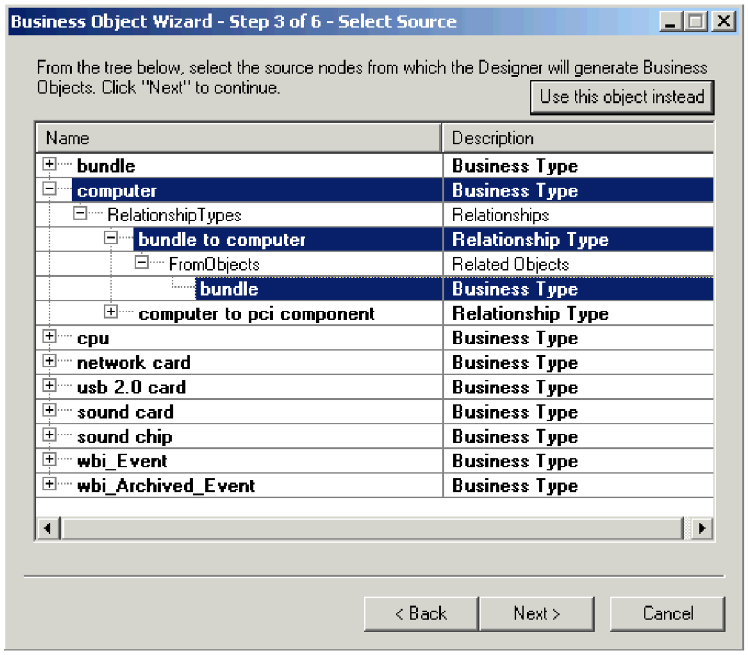
Here are two general problems and their resolution.

### Unsupported business objects

You may receive a message like this in your log or trace file:
Unable to find version “*.*.*” of business object definition “wbi_computer_bundle_to_computer”
To resolve it, you need to add the business object mentioned in the message to the list of supported business objects in Connector Configurator.

### Incomplete business object relationships

When you use the ODA, you may wish to model two business object types and their connecting relationship. Be sure to select all three in a hierarchical fashion as displayed in Figure 9 on page 43.



From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

| Name | Description |
|---|---|
| bundle | Business Type |
| computer | Business Type |
| RelationshipTypes | Relationships |
| bundle to computer | Relationship Type |
| FromObjects | Related Objects |
| bundle | Business Type |
| computer to pci component | Relationship Type |
| cpu | Business Type |
| network card | Business Type |
| usb 2.0 card | Business Type |
| sound card | Business Type |
| sound chip | Business Type |
| wbi_Event | Business Type |
| wbi_Archived_Event | Business Type |

*Figure 9. Selecting business object relationships*

If you do not do this, the ODA will create an incomplete model of the types and their shared relationship.

# 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 15 on page 47 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 15. 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 15. 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 15. 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<br><br>`no` (to disable polling)<br><br>`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><br>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*`/`<br>`SYNCHRONOUSREQUESTQUEUE` | Component restart | Delivery Transport is JMS |

Table 15. Summary of standard configuration properties (continued)

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

## Standard configuration properties

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

### AdminInQueue

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

The default value is `CONNECTORNAME/ADMININQUEUE`.

### AdminOutQueue

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

The default value is `CONNECTORNAME/ADMINOUTQUEUE`.

### AgentConnections

Applicable only if `RepositoryDirectory` is <REMOTE>.

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

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

### AgentTraceLevel

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

## ApplicationName

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

## BrokerType

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

## CharacterEncoding

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

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

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

## ConcurrentEventTriggeredFlows

Applicable only if `RepositoryDirectory` is <REMOTE>.

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

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

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

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

## ContainerManagedEvents

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

The default value is `No value`.

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

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

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

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

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

## ControllerStoreAndForwardMode

Applicable only if `RepositoryDirectory` is <REMOTE>.

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

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

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

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

The default is `true`.

## ControllerTraceLevel

Applicable only if `RepositoryDirectory` is <REMOTE>.

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

## DeliveryQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default value is `CONNECTORNAME/DELIVERYQUEUE`.

## DeliveryTransport

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

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

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

### WebSphere MQ and IDL

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

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

### JMS

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

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

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

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

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

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

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

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

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

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

## DuplicateEventElimination

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

This property can also be set to `false`.

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

## FaultQueue

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

The default value is `CONNECTORNAME/FAULTQUEUE`.

## JvmMaxHeapSize

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

The default value is 128m.

## JvmMaxNativeStackSize

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

The default value is 128k.

## JvmMinHeapSize

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

The default value is 1m.

## jms.FactoryClassName

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

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

## jms.MessageBrokerName

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

The default is `crossworlds.queue.manager`.

## jms.NumConcurrentRequests

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

The default value is 10.

## jms.Password

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

There is no default.

## jms.UserName

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

There is no default.

## ListenerConcurrency

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

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

## Locale

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

A locale name has the following format:

*`ll_TT.codeset`*

where:

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

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

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

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

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

## LogAtInterchangeEnd

Applicable only if RespositoryDirectory is <REMOTE>.

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

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

## MaxEventCapacity

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

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

## MessageFileName

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

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

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

## MonitorQueue

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

The default value is CONNECTORNAME/MONITORQUEUE

## OADAutoRestartAgent

Valid only when the RepositoryDirectory is <REMOTE>.

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

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

The default value is `false`.

## OADMaxNumRetry

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

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

The default value is `1000`.

## OADRetryTimeInterval

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

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

The default is `10`.

## PollEndTime

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

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

## PollFrequency

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

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

The default is `10000`.

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

## PollQuantity

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

## PollStartTime

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

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

## RequestQueue

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

The default value is `CONNECTOR/REQUESTQUEUE`.

## RepositoryDirectory

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

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

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

## ResponseQueue

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

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

## RestartRetryCount

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

The default is `3`.

## RestartRetryInterval

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

The default is 1.

## SourceQueue

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

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see "ContainerManagedEvents" on page 51.

The default value is `CONNECTOR/SOURCEQUEUE`.

## SynchronousRequestQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default is `CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE`

## SynchronousResponseQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default is `CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE`

## SynchronousRequestTimeout

Applicable only if `DeliveryTransport` is JMS.

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

The default value is 0.

## WireFormat

Message format on the transport.

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

## WsifSynchronousRequest Timeout

WAS integration broker only.

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

The default value is 0.

## XMLNameSpaceFormat

WebSphere message brokers and WAS integration broker only.

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

The default value is `short`.

# Appendix B. Connector Configurator

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

You use Connector Configurator to:
- Create a connector-specific property template for configuring your connector
- Create a configuration file
- Set properties in a configuration file

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

The topics covered in this appendix are:
- "Overview of Connector Configurator" on page 61
- "Starting Connector Configurator" on page 62
- "Creating a connector-specific property template" on page 63
- "Creating a new configuration file" on page 65
- "Setting the configuration file properties" on page 68
- "Using Connector Configurator in a globalized environment" on page 74

## 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 62).

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 63 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 67.)

## 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 63.
- To use an existing file, simply modify an existing template and save it under the new name.

### Creating a new template

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

To create a template:

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

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

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

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

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

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

## Specifying general characteristics

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

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

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

## Specifying values

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

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

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

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

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

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

### Setting dependencies

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

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

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

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

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

## Creating a new configuration file

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

To select a broker:

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

You can also do this:

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

## Creating a configuration file from a connector-specific template

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

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

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

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

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

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

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

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

## Using an existing file

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

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

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

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

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

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

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

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

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

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

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

## Completing a configuration file

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

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

1. Under the **Standard Properties** tab, select the value field for the `BrokerType` property. In the drop-down menu, select the value `ICS`, `WMQI`, or `WAS`.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 70..
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 46.

## Specifying supported business object definitions

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

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

### If ICS is your broker

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

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

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

To delete a business object from the supported list:

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

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

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

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

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

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

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

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

#### If a WebSphere Message Broker is your broker

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

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

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

#### If WAS is your broker

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

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

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

## Associated maps (ICS only)

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

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

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

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

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

- **Business Object Name**

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

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

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

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

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

  To explicitly bind a map:

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

## Resources (ICS)

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

### Messaging (ICS)

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

### Setting trace/log file values

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

To change the logging and tracing values:

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

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

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

### Data handlers

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

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

## Saving your configuration file

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

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

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

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

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

## Changing a configuration file

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

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

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

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

## Completing the configuration

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

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

## Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

# Appendix C. Sample scenario

This appendix contains a working sample of an eMatrix scenario. It demonstrates how you can use the adapter for eMatrix to integrate eMatrix with other applications and technologies supported by WebSphere Business Integration adapters.

Topics covered in this appendix include:

- "Overview" on page 77
- "Assumptions" on page 77
- "Installing the scenario" on page 77
- "Running the scenario" on page 81

## Overview

In this simple example, the eMatrix adapter exchanges information with the Visual Test Connector. We will assume that the Visual Test Connector represents a computer parts procurement system. As parts become available, their availability is communicated to the computer manufacturer, who uses the eMatrix adapter to automatically update their eMatrix system. When the computer manufacturer's parts requirements change, the changes are automatically detected and sent to the procurement system.

## Assumptions

The assumptions are:

1. You have installed, and are experienced with, one of the supported integration brokers (see Chapter 2 for details).
2. You have installed and are experienced with eMatrix version 9.6.0.1 and one of its collaboration servers.

## Installing the scenario

The first step is to set up eMatrix and create the eMatrix business object types.

**Note:** %SAMPLE_FOLDER% refers to the folder in which you found this document.
%WBIA% refers to the folder containing your current WebSphere Business Integration Adapters installation.

### Defining eMatrix business object types

First, you define the types which represent the parts of a computer system. To do this:

1. Open the eMatrix MQL shell.
2. Cut and paste the following commands into the shell:

```
#BEGIN
#1. create the attributes

add attribute 'component cost'                  type real;
add attribute 'in stock'                        type boolean;
add attribute 'date available'                  type date;
add attribute 'component manufacturer'           type string;
add attribute megahertz                         type integer;
add attribute slot                              type integer;

#2. create the types

add type bundle;
add type computer;
add type cpu attribute 'component cost' attribute 'in stock' attribute
'date available'
attribute 'component manufacturer' attribute megahertz;
add type 'network card' attribute 'component cost' attribute 'in stock'
attribute 'date available' attribute 'component manufacturer';
add type 'usb 2.0 card' attribute 'component cost' attribute 'in stock'
attribute 'date available' attribute 'component manufacturer';
add type 'sound card' attribute 'component cost' attribute 'in stock'
attribute 'date available' attribute 'component manufacturer';
add type 'sound chip' attribute 'component cost' attribute 'in stock'
attribute 'date available' attribute 'component manufacturer'
attribute megahertz;
#3. create the relationships
add relationship 'bundle to computer' to type computer cardinality n from
type bundle cardinality n;
add relationship 'computer to pci component' to type 'network card',
'sound card', 'usb 2.0 card' cardinality n from type computer
cardinality 1 attribute slot;
add relationship 'sound card to sound chip' to type 'sound chip'
cardinality 1 from type 'sound card' cardinality 1;
#4. create policy

add policy 'computer manufacturing' type bundle, computer, cpu,
'network card', 'usb 2.0 card', 'sound card', 'sound chip' state
'pre production' state 'ready for production';

#5. create vault

add vault 'wbi computer manufacturer'

#6. create the 'adapter' person

add person adapter password wbia type business, system access all admin all;

#END
```

Note that step 6 above gives the user 'adapter' permission to perform all operations in the system. Not all these privileges are required to run this example. The following MQL command provides all those necessary:

```
add person adapter password wbia type business access all
admin none;
```

However, if you use the eMatrix adapter for tasks not covered in these samples, you may need the privileges provided in step 6.

## Creating eMatrix business object types

Next, you create the data that represents the computer system of the hypothetical computer manufacturer. To do this:

1. Open the eMatrix MQL shell.
2. Cut and paste the following commands into the shell:

```
#business objects
add bus bundle 'home office and entertainment' 1
    policy              'computer manufacturing'
    vault               'wbi computer manufacturer';
```

```
add bus computer 'gamer deluxe' 1
    policy               'computer manufacturing'
    vault                        'wbi computer manufacturer';
add bus cpu 'max processor' 1
    policy               'computer manufacturing'
    vault                        'wbi computer manufacturer'
    megahertz                    2000
    'in stock        '            true
    'date available    '     9/30/01
    'component manufacturer' acetech
    'component cost    '     67;
add bus 'sound card' 'super sonic' 1
    policy                      'computer manufacturing'
    vault                       'wbi computer manufacturer'
    'in stock        '           false
    'date available    '     6/12/03
    'component cost    '     50
    'component manufacturer' 'feedback inc.';
add bus 'sound chip' 'super sound chip' 1
    policy                     'computer manufacturing'
    vault                      'wbi computer manufacturer'
    'component manufacturer' ChipsWeMake
    megahertz                  100;
add bus 'usb 2.0 card' 'usb enabler' 1
    policy                     'computer manufacturing'
    vault                       'wbi computer manufacturer'
    'in stock        '           true
    'date available    '     2/10/02
    'component manufacturer' usbworks
    'component cost    '     12;
add bus 'network card' 'net connector' 1
    policy                      'computer manufacturing'
    vault                        'wbi computer manufacturer'
    'in stock        '            true
    'date available    '     8/12/02
    'component cost    '     10
    'component manufacturer' 'tcp specialists';
#relationships
connect bus bundle 'home office and entertainment' 1 relationship
'bundle to computer' to computer 'gamer deluxe' 1;
connect bus computer 'gamer deluxe' 1 relationship 'computer to pci component'
to 'sound card' 'super sonic' 1
slot 1;
connect bus computer 'gamer deluxe' 1 relationship 'computer to pci component'
to 'usb 2.0 card' 'usb enabler' 1
slot 2;
connect bus 'sound card' 'super sonic' 1 relationship 'sound card to
sound chip' to 'sound chip' 'super sound chip' 1;
```

## Installing the JPOs

You will need to install two JPOs, one to create the WBIA event and archive event types, and one to act as a trigger when detecting events.

1. Open the eMatrix Business Modeler and log in as a user with the ability to create a program.
2. Select **Object>new>program**.
3. In the text box titled **Name**, enter **WBIEventLogger**.
4. Choose **Java** for the **Type**.
5. Click the **Code** tab.
6. Cut and paste the code from the %WBIA%\connector\eMatrix\dependencies\WBIEventLogger.java file into the text box on this tab.

You may need to set the 'wbiPrefix' and 'vault' values in the source code. Please review the comments in the code for more information.

7. Click **Create**.

Follow the same steps to create the InstallEventTables program, with these differences:

- Name the program InstallEventTables.
- Cut and paste the code from the %WBIA%\connector\eMatrix\dependencies\InstallEventTables.java file.

## Installing the triggers

In this section, you create a trigger which will detect when the hypothetical computer design changes. You create the trigger on the 'computer to pci component' relationship. To do this:

1. In the eMatrix Business Modeler, click on the binoculars in the upper left-hand corner of the window.
2. Select **Relationship** and click **Find**.
3. Double-click the 'computer to pci component' business object.
4. In the new window, select the **Triggers** tab and click **Add**.
5. Select the **Create** trigger and click **OK**.
6. In the new window, type **WBIEventLogger** in the **Action** text box.
7. Type the following into the **Input** text box:
   ```
   -method recordEvent ${FROMOBJECTID} n=wbi_computer_poll v=Update
   ```
8. Click **OK**, then click **Edit** to finish creating the trigger.

## Installing the event business objects

The final step in installing the scenario is to install the event business objects. To do so, execute this command from the MQL command line:

```
exec program InstallEventTables;
```

## Configuring the eMatrix adapter

The first step is to configure the eMatrix connector. To do this:

1. Run the Connector Configurator.
2. Open `%SAMPLE_FOLDER%\EmatrixConnector.cfg` and set these properties:
   ```
   RepositoryDirectory property = %SAMPLE_FOLDER%\repos
   PollFrequency property = key
   ```
3. Set the ApplicationUserName and ApplicationPassword properties to the user on whose behalf the connector will be running.
4. Set the eMatrixServer property to the name of the eMatrix server you will be connecting to.
5. Set the HostName property to the URL, including the port, of the eMatrix collaboration server that the adapter will be communicating with.
6. Set the EventVault property to the name of the vault which will store your events.
   You may create a new vault to store events. To do so, run the MQL eMatrix client and type 'add vault <*vaultname*>' in the text box at the bottom of the window. Make sure that the this value matches the value of the **vault** string in the WBIEventLogger JPO. By default both values should be 'WBI_Events'.
7. Set the KeepRelations property to 'true'.

8. Open %SAMPLE_FOLDER%\PortConnector.cfg and set these properties:
   ```
   RepositoryDirectory = %SAMPLE_FOLDER%\repos
   ```

### Supporting business objects

For the adapter to use business objects, it must first support them. To provide support for the business objects you want to use, do this:

1. With the Connector Configurator open, select the **Supported Business Objects** tab and add these business objects:
   - wbi_computer
   - wbi_computer_poll
2. Check the **Agent supported** box next to each business object.
3. Repeat the steps above with the Port connector.

### Creating queues

This sample scenario requires that several queues be defined in your queue manager. To create the necessary queues, type:

```
RUNMQSC crossworlds.queue.manager
```

from the command line and issue the following commands:

```
DEFINE QL('ADMININQUEUE')
DEFINE QL('ADMINOUTQUEUE')
DEFINE QL('DELIVERYQUEUE')
DEFINE QL('FAULTQUEUE')
DEFINE QL('REQUESTQUEUE')
DEFINE QL('RESPONSEQUEUE')
DEFINE QL('EMATRIXCONNECTOR/RESPONSEQUEUE')
DEFINE QL('SYNCHRONOUSREQUESTQUEUE')
DEFINE QL('SYNCHRONOUSRESPONSEQUEUE')
DEFINE QL('PORTCONNECTOR/SYNCHRONOUSREQUESTQUEUE')
DEFINE QL('PORTCONNECTOR/SYNCHRONOUSRESPONSEQUEUE')
```

Update the start script by opening either the start_eMatrix.bat or start_eMatrix.sh file (depending on your platform) and setting the EMADK and EM_LIB variables in these files.

## Running the scenario

This example provides scenarios for two activities; request processing and event notification.

## Scenario 1: request processing

In the request processing scenario the procurement system informs the eMatrix system that one of the components used in a design is now available. The system sends a business object that represents the updated system component.

### Start the Visual test connector

1. Select **Start>Programs>WebSphere Business Integration Adapters>Tools>Visual Test Connector.** The connector starts.
2. In the connector window, select **File>Create/Select Profile**. The Connector Profile window appears.
3. If you have not already done so, create a profile that emulates the Port Connector in the profile window, as follows:
   - Select **File>New Profile**.
   - Click **Browse** and select the %SAMPLE_FOLDER%\PortConnector.cfg file.

- In the Connector Name text box, type Port Connector.
  From the Broker Type drop-down menu, select WMQI.
- Click **OK**.

4. If you have already created a profile, select it and click **OK**.
5. Once you have a profile, select **File>Connect**.

### Start the adapter for eMatrix

Start the adapter for eMatrix by running this command:

```
%WBIA%\connectors\eMatrix\start_eMatrix.bat eMatrix wmqi
-c%SAMPLE_FOLDER%\EmatrixConnector.cfg
```

You can substitute any combination of alphanumeric characters (starting with a letter) for the value of “wmqi”.

### Start the eMatrix components

Start the eMatrix Collaboration server. Make sure that the instance of Oracle in which the eMatrix system exists is running. Then open an MQL Command window.

### Check the current status in eMatrix

At the MQL command line, type this command:

```
expand bus 'computer' 'gamer deluxe' 1 select bus
'attribute[in stock]';
```

Note that the supersonic sound card is not in stock.

### Send a request

To do this, follow these steps.

1. In the Visual Test Connector, select **Edit>Load BO**.
2. Select the file %SAMPLE_FOLDER%\sound_card_update.bo.
3. Enter any name in the popup window and click **OK**.
4. Click **Send BO** in the upper left-hand corner of the Visual Test Connector. The icon contains one yellow arrow.
   You will see text messages appear in the window containing the adapter for eMatrix. These message indicate that the request is being processed.

### Check the updated status in eMatrix

Type the following at the MQL command line:

```
expand bus 'computer' 'gamer deluxe' 1 select bus
'attribute[in stock]';
```

Note that the supersonic sound card is now in stock.

### Reset the status in the database

In order to run this scenario again, you must reset the supersonic sound card so that it is not in stock.To do so, run this MQL command:

```
modify bus 'sound card' 'super sonic' 1 'in stock' false;
```

## Scenario 2: event notification

In this scenario, the computer manufacturer changes the configuration of one of its computer systems. The change will trigger a notification to be forwarded to the procurement system.

1. Repeat steps 1 through 3 from scenario 1 above.
2. Follow the steps below.

## Update the computer configuration

The computer configuration has changed. The computer manufacturer is adding a network card to the computer. At the MQL command line, run:

```
connect bus 'computer' 'gamer deluxe' 1 relationship
'computer to pci component' to 'network card' 'net

connector' 1 slot 3;
```

## Poll the event

Go to the command window containing the adapter for eMatrix. At the bottom of the window, type ’p’. This causes the adapter to poll.

## View the event

The adapter will record the new computer configuration and forward it to the procurement system (in this case, the Visual Test Connector).

Look in the window pane on the right of the Visual Test Connector. You will see a line containing the text ’wbi_computer.Update’. This is the new computer configuration represented as a WebSphere business object. You can double-click on it to see its contents.

## Reset the computer configuration

To run this scenario again, you must disconnect the eMatrix business objects. To do so, run this command:

```
disconnect bus 'computer' 'gamer deluxe' 1 relationship
'computer to pci component' to 'network card'

'net connector' 1;
```

This completes the installation and testing of the sample scenarios. If you have performed all the steps above successfully, you should have a working sample that enables the adapter for eMatrix.

# Notices

## 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

IBM WebSphere Business Integration Adapter Framework V2.4.0.