IBM WebSphere Business Integration Adapters

Implementing Adapters with WebSphere Message Brokers

IBM WebSphere Business Integration Adapters

Implementing Adapters with WebSphere Message Brokers

Note!
Before using this information and the product it supports, read the information in “Notices” on page 183.

20February2004

This edition of this document applies to the IBM WebSphere Business Integration Adapter Framework, version 2.4, and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

Contents

About this document

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 how to install, configure, deploy, and manage WebSphere Business Integration adapters with supported WebSphere Business Integration message brokers. The message brokers supported are the following: WebSphere MQ Integrator 2.1, WebSphere MQ Integrator Broker 2.1, and WebSphere Business Integration Message Broker 5.0.

Notes:

1. In this document, “application” refers to an enterprise software product that a partner of IBM develops and sells, or that a customer of IBM develops and uses. An application participates in an IBM WebSphere Integration Adapters solution, but it is not part of the IBM WebSphere Integration Adapters product.
2. Illustrations in this manual are only examples used to show structure and concepts. They do not necessarily document specific business integration scenarios.

Audience

This document is for customers and consultants who are implementing or administering WebSphere Business Integration adapters using a WebSphere message broker as the integration broker for the adapter. It is assumed that the reader already knows how to configure and administer the message broker that will be used, and has a solid understanding of WebSphere MQ messaging and message flows.

If you are already familiar with WebSphere Business Integration adapters that use another integration broker, you should still read this book carefully because there are significant differences between their functionality and behavior other integration broker environments and the WebSphere message broker environment.

Related documents

To deploy and manage WebSphere Business Integration adapters with a WebSphere message broker as the integration broker, you might need to refer to documentation that spans the libraries of several different products:

- Individual WebSphere Business Integration adapters
- One of the following: WebSphere MQ Integrator Broker, WebSphere MQ Integrator or WebSphere Business Integration Message Broker
- WebSphere MQ.

Information about related books and instructions for accessing them are provided below.

WebSphere Business Integration adapters publications

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

You can install the documentation or read it directly online at the following Web site: `http://www.ibm.com/websphere/integration/wbiadapters/infocenter`

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

WebSphere message broker publications

This implementation guide provides the information necessary to deploy a business integration system consisting of WebSphere Business Integration adapters and an *existing* WebSphere message broker (one of WebSphere MQ Integrator 2.1, WebSphere MQ Integrator Broker 2.1, or WebSphere Business Integration Message Broker 5.0.)

Detailed information about installing, configuring, and managing the message broker, as well as creating and managing message flows, can be found at the IBM web sites listed below:

- **For general information about the WebSphere MQ family of products:**

  `http://www.ibm.com/software/integration/mqfamily`
- **For WebSphere MQ Integrator 2.1:**

  `http://www.ibm.com/software/integration/mqfamily/library /manualsa/manuals/wsmqsiv21.html`
- **For WebSphere Business Integration Message Broker:**

  `http://www.ibm.com/software/integration/wbimessagebroker/library`

System Manager publications

Many configuration and administrative activities for WebSphere Business Integration adapters can be performed using a graphical user interface called System Manager. For information about System Manager beyond what is covered in this manual, refer to the following two guides:

- *IBM WebSphere Interchange Server System Administration Guide*
- IBM WebSphere Interchange Server Implementation Guide

These are available at the following web site:

`http://www.ibm.com/websphere/integration/wicserver/infocenter`

Typographic conventions

This document uses the following conventions:

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

blue text	Blue text, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click any blue text to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
`[ ]`	In a syntax line, square brackets surround an optional parameter.
`...`	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.
`< >`	In a naming convention, angle brackets surround individual elements of a name to distinguish them from each other, as in `<server_name><`*`connector_name`*`>tmp.log`.
/, \	In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes. All product pathnames are relative to the directory where the product is installed on your system.
`ProductDir`	Represents the directory where the product is installed.
`%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.
Windows: **UNIX:** **AIX:** **Solaris:** **HP-UX:**	Paragraphs beginning with any of these indicate notes listing operating system differences.

Summary of Changes

This chapter contains information about changes to *Implementing Adapters with WebSphere Message Brokers* for the current release.

New with WebSphere Business Integration Adapter Framework v 2.4.0

February 2004

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

- Appendix E, "System Manager and the Eclipse Workbench," on page 129 has been added explaining System Manager and the Eclipse Workbench.
- Information about starting and stopping connectors in Chapter 7, "Administering the business integration system," on page 57 and in Appendix D, "Connector startup options," on page 125 has been updated.

December 2003

This manual has been updated to include the major changes listed below:

- The title of this book has changed. It was previously called *Implementing Adapters with WebSphere MQ Integrator Broker.*
- The scope of this document has changed to reflect that several different message brokers (WebSphere MQ Integrator 2.1, WebSphere MQ Integrator Broker 2.1, and WebSphere Business Integration Message Broker 5.0) are now supported for use as integration brokers for adapters. Previously only one broker, the WebSphere MQ Integrator Broker was supported.
- System Manager is now used to deploy adapter projects to the integration broker.
- During configuration, you can now select whether your project will use ″long″ or ″short″ XML namespaces in its serialized business objects.
- The manual now mentions using System Manager to start, stop and pause adapters.
- Changes made to accommodate new installation and packaging of adapters.

New in WebSphere Business Integration Adapters v 2.3.1

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

- The title of this book has changed. It was previously called *Implementation Guide for WebSphere MQ Integrator Broker.*
- The message domain in RFH2 message headers can now be set to XML through a new standard configuration property. This is described in Appendix A, "WebSphere MQ message formats," on page 99.
- Appendix H, "Upgrading WebSphere Business Integration adapters," on page 173 has been updated to contain instructions for updating to release 2.3.1.
- The previously existing chapters ″Installing WebSphere Business Integration Adapters on Windows″ and Installing WebSphere Business Integration Adapters on UNIX″ have been replaced with a single chapter, ″Installing WebSphere Business Integration Adapters.″ Details of WebSphere Integration Adapter installation are contained in a separate manual, ″Installation Guide for WebSphere Business Integration Adapters.″

New in WebSphere Business Integration Adapters v 2.2.0

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

- Appendix H, "Upgrading WebSphere Business Integration adapters," on page 173 includes information about upgrading WebSphere Business Integration Adapters to version 2.2.0.
- "Guaranteed event delivery" on page 31 provides information about the guaranteed-event-delivery feature, which ensures that each event is never processed more than once.
- In"Performing a silent installation or uninstallation of WebSphere Business Integration Adapters" on page 54 explains how to perform an installation or uninstallation of WBIA without human intervention.
- In Chapter 7 "Performing a silent installation or uninstallation of WebSphere Business Integration Adapters" on page 63 explains how to perform an installation or uninstallation of WBIA without human intervention.
- "Using WBIA batch files to configure WebSphere MQ queues:" on page 76 provides information on a new batch file you can use to configure WebSphere MQ queues in the business integration system.
- "Clearing messages from WebSphere MQ queues" on page 67 explains how to use a new batch file for clearing messages from WebSphere MQ queues.
- The following names have changed:
  - WebSphere MQ Integrator is now WebSphere MQ Integrator Broker
  - MQSeries is now WebSphere MQ.
- References to Borland VisiBroker have been removed because this product is no longer needed for WebSphere Business Integration Adapters that use WebSphere MQ Integrator Broker.

New in WebSphere Business Integration Adapters v 2.1.0

The changes made in WebSphere Business Integration Adapters 2.1 do not affect the content of this document.

New in WebSphere Business Integration Adapters v 2.0.1

This book has been updated to include the following changes:

- Appendix H, "Upgrading WebSphere Business Integration adapters," on page 173 describes how to upgrade from WBIA V 2.0 to
  V 2.0.1.
- "Installing for Windows systems" on page 49 explains how to uninstall the WebSphere business integration system or selected adapters in a Windows environment.
- "Uninstalling WebSphere Business Integration Adapters" on page 62" explains how to uninstall the WebSphere business integration system or selected adapters in a UNIX environment.
- Appendix C, "Standard configuration properties for connectors," on page 109 lists standard configuration properties for connectors that use WebSphere MQ Integrator as the integration broker.
- Appendix G, "Using Visual Test Connector," on page 159 describes how to use the Visual Test Connector tool to verify that the business integration components you have defined and configured are working correctly.
- "Creating a message broker project" on page 82 has been revised and enhanced.

- Information about changes to support internationalization has been added.

Part 1. Overview and concepts

Chapter 1. Overview of WebSphere Business Integration adapters

As connectivity configurations and topologies have grown in complexity and size, the ability to move information between applications, using data transformations and routing rules, has become more critical than ever before. By using a WebSphere message broker as the integration broker for your WebSphere Business Integration adapters , you can achieve the cross-application information connectivity you need to attain your business goals.

WebSphere Business Integration adapters use modular components and application-independent business logic. This modular approach provides many benefits over traditional custom integration efforts—including faster deployment, easier application upgrades, and reusability of both business process flows and application access code. The **business integration system** the adapters support is distributed and flexible, with customization features that make it possible to meet site-specific and application-specific needs.

This implementation guide explains how to deploy and manage a WebSphere business integration system featuring WebSphere Business Integration adapters with supported WebSphere message brokers functioning as integration brokers for the adapter. The following WebSphere message brokers are supported as integration brokers:
- WebSphere MQ Integrator 2.1.0
- WebSphere MQ Integrator Broker 2.1.0
- WebSphere Business Integration Message Broker 5.0

Note: Throughout this document, the term ″message brokers″ is used to refer specifically to the software products listed above and no others. When ″message brokers″ is used, the feature being described applies to all of the products listed above.

This guide is divided into two parts:
- Part 1 presents a conceptual overview of WebSphere Business Integration adapters, the major components of the adapters portfolio, and the process by which they interact with the integration broker to create a business integration system.
- Part 2 offers task-oriented information to help you install, configure, and administer WebSphere Business Integration adapters with WebSphere message brokers.

This chapter describes the architecture of a business integration system and introduces the components of an adapter. It contains the following sections:
- “What is the WebSphere business integration system?” on page 4
- “What are WebSphere message brokers?” on page 5
- “What are WebSphere Business Integration adapters?” on page 5
- “How the WebSphere business integration system works” on page 6

A note about documents you need

To perform all the tasks necessary to deploy a business integration adapter, you need to use this book together with other books in the WebSphere Business Integration adapters library, particularly:

- The *Business Object Development Guide*
- The adapter user guides for the adapters you are deploying.

If you are developing a custom adapter, you also need to use either or both of the following books:

- *Connector Development Guide for Java*
- *Connector Development Guide for C++* .

For a complete listing of the books in the library, see "WebSphere Business Integration adapters publications" on page viii.

What is the WebSphere business integration system?

To compete effectively in e-business, an enterprise must meet two challenges:

- It must move business information among diverse sources to perform business exchanges.
- It must process and route business information among disparate applications in the enterprise environment

A business integration system addresses both these needs with flexibility and extensibility.

At the highest-level, the **WebSphere business integration system** consists of an integration broker (a supported WebSphere message broker, WebSphere InterChange Server, or WebSphere Application Server) and a set of adapters that allow heterogeneous business applications to exchange data through the coordinated transfer of information, in the form of business objects.

Figure 1 on page 5 shows a simplified representation of a WebSphere business integration system.

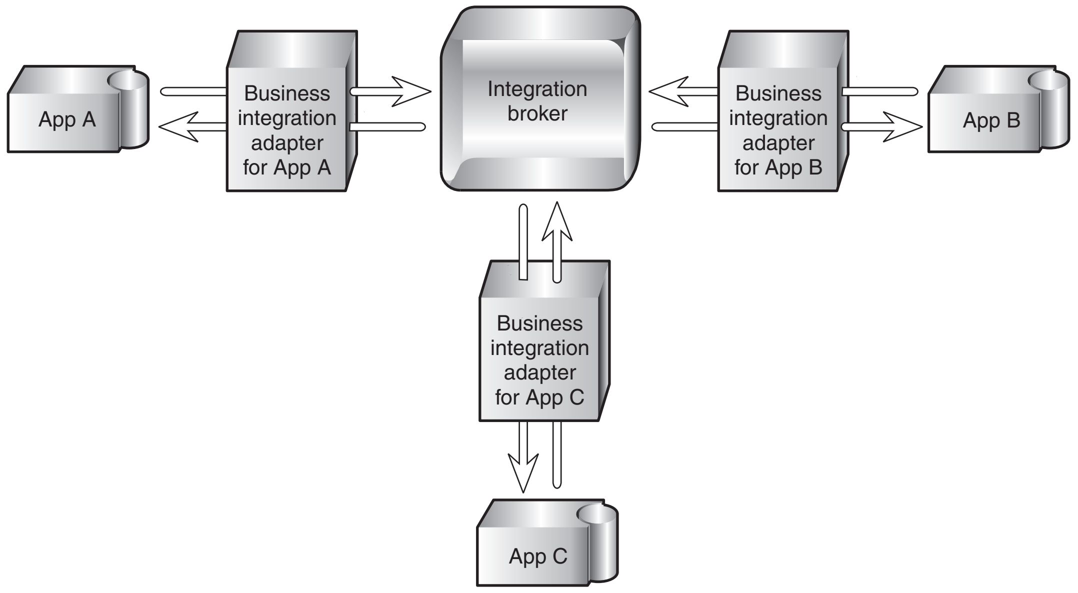

App A
Business integration adapter for App A
Integration broker
Business integration adapter for App B
App B
Business integration adapter for App C
App C

Figure 1. High-level view of a WebSphere business integration system

What are WebSphere message brokers?

WebSphere message brokers are a subset of the integration brokers that can be used in the WebSphere business integration system. Message brokers extend the basic connectivity and transport capabilities of WebSphere MQ messaging. The message brokers supported as integration brokers for adapters are the following:

- WebSphere MQ Integrator 2.1.0
- WebSphere MQ Integrator Broker 2.1.0
- WebSphere Business Integration Message Broker 5.0

WebSphere message brokers enable diverse applications to exchange information in dissimilar forms by handling the processing required for the information to arrive in the right place, in the correct format, in accordance with user-defined rules. Data exchange is performed by the broker without requiring applications to have any knowledge of the data conventions or requirements of the applications receiving their data.

What are WebSphere Business Integration adapters?

The WebSphere Business Integration adapters portfolio consists of a collection of software programs, application programming interfaces (APIs), and tools you can use to enable applications to exchange business data through the WebSphere message brokers. Each business application requires its own application-specific adapter to participate in the business integration system.

Each adapter includes:

- A connector that links the application to the integration broker
- Tools with graphical user interfaces to help you configure a connector and create the business object definitions needed for the application.

- An Object Discovery Agent (ODA), which runs against an application's data store to create business object definitions, which you can then refine. Note that WebSphere Business Integration adapters for some applications do not include an ODA.
- An Object Discovery Agent Development Kit (ODK), which consists of a set of APIs you can use to develop an ODA.

A separately-available Adapter Development Kit (ADK) provides a framework for developing custom adapters in cases where a prebuilt connector for a particular legacy or specialized application is not available from IBM.

How the WebSphere business integration system works

In the WebSphere business integration system, connectivity for moving data between applications and the integration broker is supplied by **connectors** using Java Message Service (JMS) over WebSphere MQ queues. A connector can reside on any machine from which it can access the necessary queues and communicate with the application.

Each connector consists of two parts—the **connector framework** and the **application-specific component**:

- The connector framework interacts with the integration broker using WebSphere MQ queues.
- The application-specific component interacts directly with an application.

The subcomponents of a connector are shown in Figure 2 on page 7..

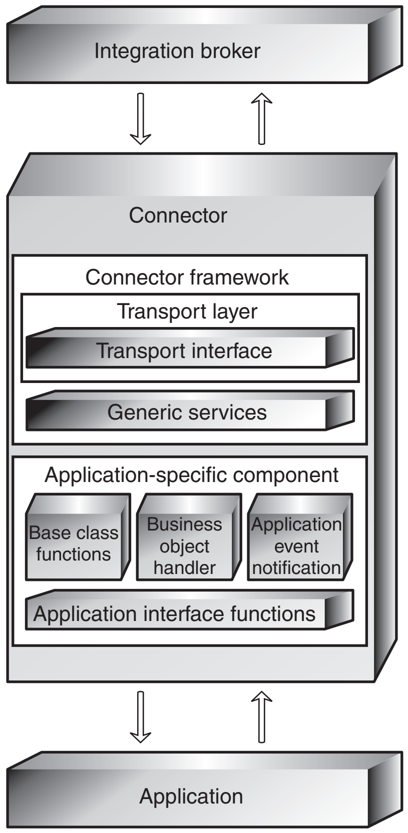


Figure 2. Subcomponents of a connector

Data is exchanged between applications by means of **application-specific business objects**, which are transported between the connector framework and the integration broker as WebSphere MQ messages (also referred to as business object messages).

Business objects encapsulate and transmit business data for the several purposes. They convey:

- New or changed data from a source application to a destination application.
- Requests for data made by a source application to a destination application.
- Data returned by an application in response to a request for data.

Instructions, associated with each piece of data, encoded as **metadata**, specify the location in the application's database where the data is to be found, created, or updated. New instances of business objects are created by the application-specific component based on templates called **business object definitions**, which specify the structure and organization of the business object's attributes, values, and metadata.

Because application-specific information and other metadata in the business object definition guide the actions of the application-specific component, such an application-specific component's behavior can be described as **metadata-driven**. An application-specific component that is metadata-driven is flexible because it has no hard-coded instructions for each type of business object that it supports. Without recoding or recompiling, the application-specific component automatically supports new business object definitions, as long as the corresponding application data can be accurately described by the connector's metadata syntax.

Within the integration broker, **WebSphere MQ message flows** define the steps in the processing of business object messages by the integration broker. They specify the set of actions, or rules, executed between receipt of the message by the integration broker, and delivery of the message to the destination application.

Data flow in the business integration system

In the business integration system, data flow—the movement and processing of data sent from one application or entity to another— can occur either as an asynchronous or a synchronous exchange between applications on a local network.

An application might need to exchange data with another application to communicate changes in its data store or to obtain data.

The exchange of data in the business integration system consists of these steps:

1. Event notification
2. Integration broker processing
3. Request processing.

Each of these is explained in more detail below.

Event notification

The process of conveying changed application data to the integration broker is called **event notification**. Most applications that participate in the business integration system are modified during the configuration process to include an **event store**, such as a table for logging the application's data changes and data requests. To detect that an application has newly changed data to share or that it needs information from another application, the connector framework initiates a **poll call** at periodic intervals. The poll call asks the application-specific component to check for changes to the application's event store.

If there has been a change since the last poll call, the application-specific component determines if a business object definition exists to represent the changed data or the data request. The presence of a suitable business object definition in the connector's local repository is an indicator that this particular change or request needs to be communicated to another application. The application-specific component sends the application data, in the form of a business object, to the connector framework. This is referred to as an **event delivery**, because a change to an application's data or a request for data is considered an event.

Figure 3 on page 9 shows a connector and its supporting infrastructure detecting a change to the application's data store and constructing an application-specific business object to convey the changed data to the integration broker.

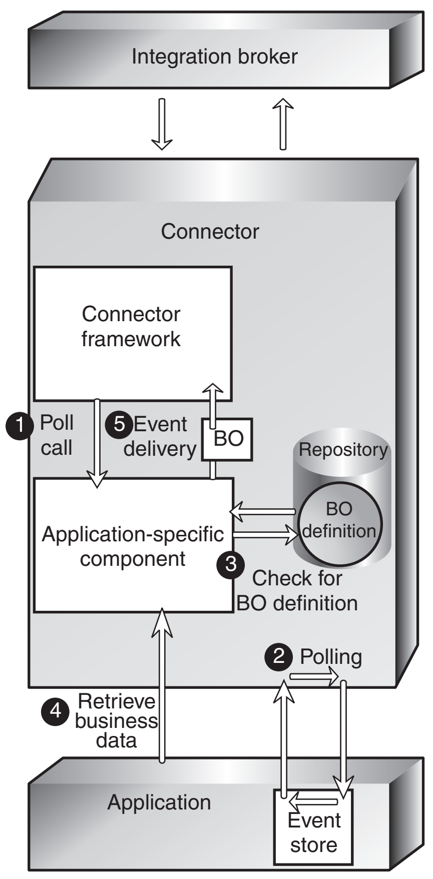


Figure 3. The connector detecting and delivering an event.

The numbers in the figure show the sequence of steps:

1. The connector framework initiates to the application-specific component to have it check for changes to the application's event store.
2. The application-specific component polls for changes to the application's event store.
3. The application-specific component determines whether the changed data maps to a supported business object definition.
4. The application-specific component instantiates a business object and uses it to retrieve the changed data.
5. The application-specific component initiates an event delivery to transfer the business object to the connector framework.

When the connector framework of the source application receives the application-specific business object, it converts the business object to a WebSphere MQ message that can be placed on a WebSphere MQ queue for receipt by the integration broker. A **data handler is** used by the connector framework to transform the business object into a message in the appropriate XML-based wire format for the destination WebSphere MQ queue. Figure 4 on page 10 shows this process.

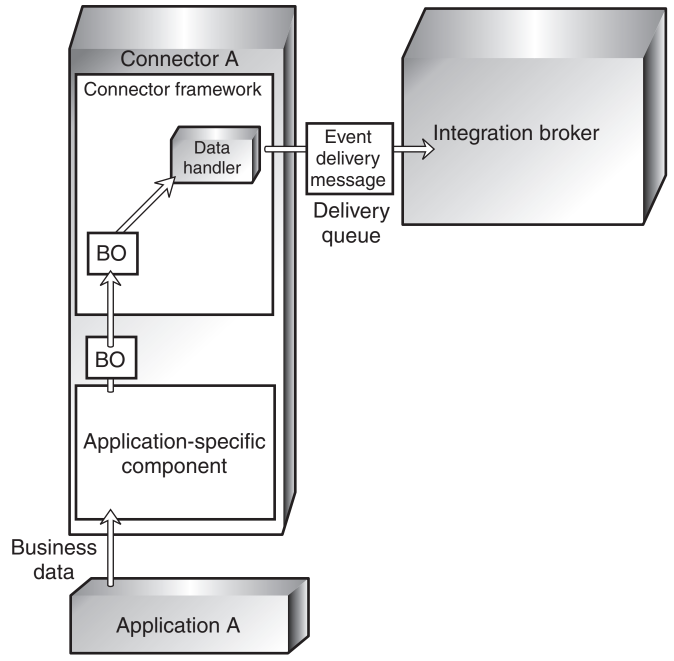


Figure 4. The connector framework transforming a business object into an MQ message.

Integration broker processing

Once the message is placed on the WebSphere MQ event delivery queue for the integration broker, the integration broker removes the message from the queue, and passes it through the message flow for the queue. Processing might involve:

- Transforming the message by computing a value
- Extracting fields from the data within the message
- Routing the message to one or more destinations
- Archiving the message in a message warehouse
- Updating database information from the message content
- Transforming the message's content or structure so that it can be processed by the destination application.

The resulting message, now called a **request**, is placed on a WebSphere MQ request queue to be transferred to the connector framework of the destination application. Figure 5 illustrates the processing performed by the integration broker.

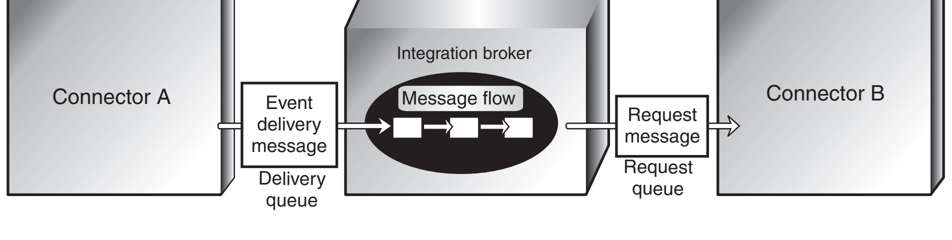


Figure 5. Integration broker processing

Request processing

Once the request has been placed on the queue for the destination connector, a **listening mechanism** notifies the connector framework that a WebSphere MQ message has arrived on its request queue and needs to be processed. The connector framework invokes the data handler to convert the WebSphere MQ message into a business object that can be processed by the destination application, as shown in Figure 6.

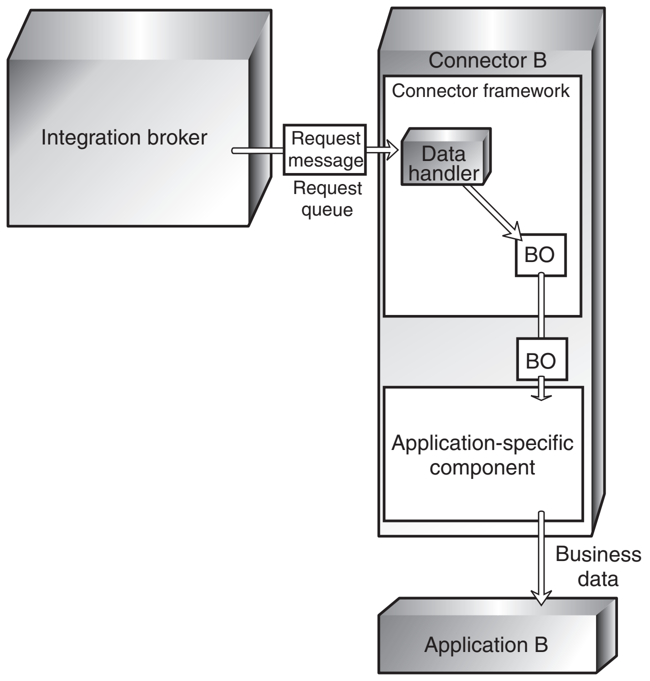


Figure 6. Request processing by the connector.

In some cases, the request might require a **response** from the destination application. Generally, a response is used to:

- Return data that the source application has requested from the destination application
- Return information to the source application about a new business entity (such as a customer or an order) that the source application has asked the destination application to create.

If a response is needed, the application-specific component modifies the request business object to carry the information and sends the business object back to the connector framework. The connector framework calls the data handler to convert the business object to a WebSphere MQ message and places the message on the reply-to queue specified in the originating request message. A correlation ID in the response message identifies the message to which it is responding. Figure 7 on page 12 illustrates how response processing is performed.

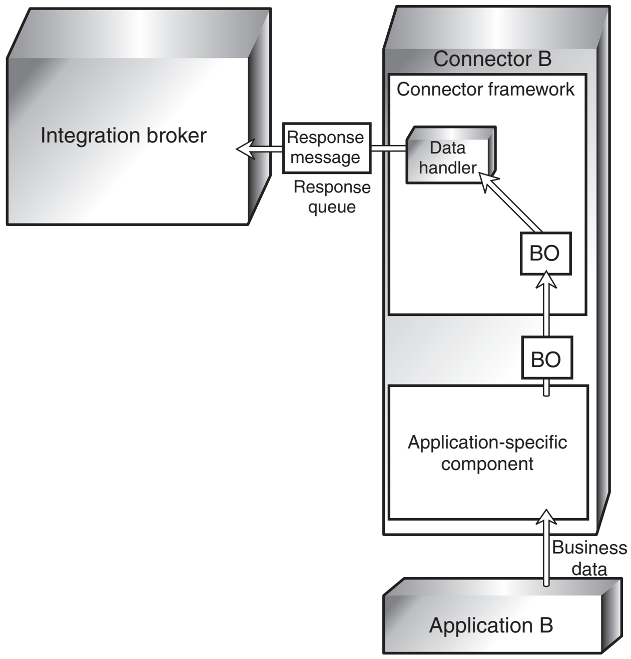


Figure 7. Response processing by the connector.

Summary of the business integration process

Now that you have learned about each step in the business integration process, you can step back for a look at the system as a whole. Figure 8 shows a summary diagram of the business integration system. Two scenarios, illustrating its operation, are presented below.

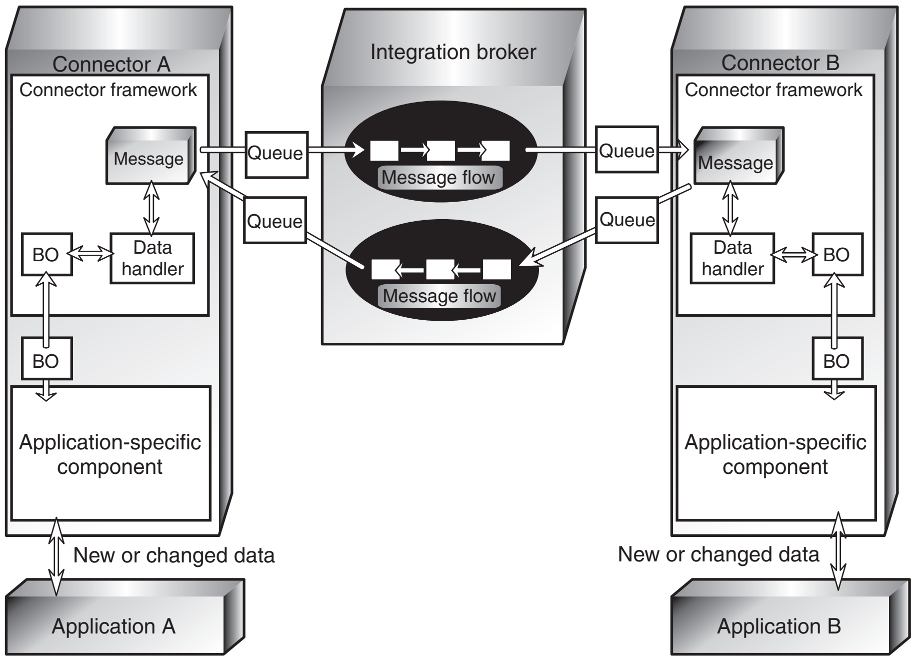


Figure 8. Data flow in the WebSphere business integration system.

Example of sending changed data to another application

As an example, here are the steps by which the business integration system enables application A to send changed data to application B for synchronization:

1. Connector A’s application-specific component detects a change to data in application A. It determines that a business object definition exists for communicating this change and uses the business object definition to construct a business object to carry the changed information.
2. The application-specific component passes the business object to the connector framework.
3. The connector framework invokes the data handler to transform the business object into a WebSphere MQ message of the correct XML-based wire format and places the message on a WebSphere MQ queue for the integration broker.
4. The integration broker receives the message and passes it to the message flow.
5. After the message is processed by the message flow, the integration broker places the resulting message on the WebSphere MQ queue for the connector for application B.
6. Connector B’s connector framework removes the message from the queue and calls the data handler to convert it to a business object that can be processed by the application-specific component.
7. Application B updates its customer information to reflect the change of address.

If application A were requesting data from application B instead of notifying it of a data change, application B would need to send a response back to application A. The following example illustrates this scenario.

Example of obtaining data from another application

Here are the steps by which the business integration system enables application A to retrieve information about a customer’s most recent purchase from application B.

1. Connector A’s application-specific component detects that application A has requested data from application B. It determines that a business object definition exists for communicating this request and uses the business object definition to construct a business object for the requested information.
2. The application-specific component passes the business object to the connector framework.
3. The connector framework invokes the data handler to transform the business object into a WebSphere MQ message of the correct wire format and places the message on a WebSphere MQ queue for the integration broker.
4. The integration broker receives the message and passes it to the message flow.
5. After the message is processed by the message flow, the integration broker places the resulting message on the WebSphere MQ queue for the connector for application B.
6. Connector B’s connector framework removes the message from the queue and calls the data handler to convert it to a business object that can be processed by the application-specific component.
7. Connector B’s application-specific component retrieves the information specified in the request and passes it back to the connector framework as a business object.
8. Connector B’s connector framework invokes the data handler to transform the business object to a response message and places it on the reply-to queue specified in the originating request.

The chapters that follow describe in more detail the business integration components introduced here and the process by which they enable applications to share data.

Chapter 2. Business objects

A business object reflects a data entity—a collection of data treated as a unit. For example, a data entity can be equivalent to an employee record, containing all the basic information about the employee - the name, address, telephone number, employee number, position code, salary, and so forth.

The business integration system creates business objects that reflect the information contained in entities. In this book, a data entity is often referred to in the context of the kind of business information it contains—for example, an *employee entity* or a *customer entity*.

Business object definitions are the templates from which the application-specific component creates a particular instance of a business object.

This chapter introduces business objects in more detail and explains how they are used by the business integration system to carry data between applications. It includes the following sections:
- "Roles of a business object"
- "Types of business objects" on page 18
- "Business object definitions and business objects" on page 18
- "A closer look at business objects" on page 21
- "Ways to create or modify business object definitions" on page 23

Roles of a business object

A business object can act as an event, a request, or a response.

Event

A business object can report the occurrence of an **application event**, an operation that affected a data entity in an application. The application event might be the creation, deletion, or change in value of that collection of data.

When a connector detects an application event and sends a business object to the integration broker, the role of the business object is to represent the event. So, it is called an **event** in the business integration system.

For example, a connector might poll an application for new employee entities on behalf of the integration broker. If the application creates a new employee entity, the connector sends an event business object to the integration broker.

Request

Requests are typically generated as follows. The integration broker sends a business object message as a **request** to the connector framework, instructing it have the application-specific component insert, change, delete, or retrieve some data in an application.

Response

When a connector finishes processing a request, it usually returns a **response** to the integration broker. For example, when a connector receives a request to create an employee record in the destination application, it sends a business object with the created employee data and a status indicator that shows that the create was successful.

Structure of a business object

A business object is a self-describing unit that contains a type (its name), processing instructions (a verb), and data (attribute values).

Figure 9 is an example of a simple business object, showing its type, verb, and attribute values.

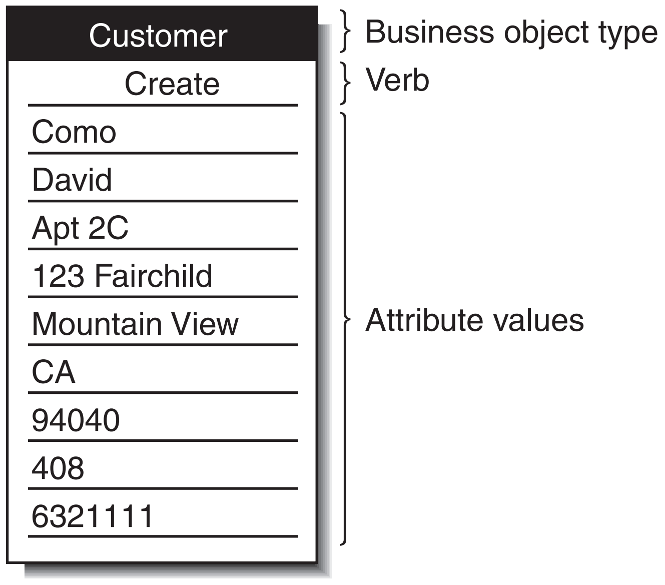


Figure 9. Business object components.

The next sections describe these components.

Business object type

Each business object has a type name that identifies it within the business integration system. For example, the type might be Customer, Employee, Item, or Contract.

Business object verbs

A business object verb specifies an action in relation to the attribute values. The verb can indicate various types of actions, depending on the role of the business object. Table 1 lists the three business object roles and describes the meaning of the verb in a business object that has each role.

Table 1. Meanings of business object verbs.

Role of business object	Meaning of verb
Event	Describes what happened in an application. For example, in an event, the Create verb indicates that the source application created a new data entity.
Request	Tells the connector how to interact with the application in order to process the business object. For example, the Update verb is a request to the connector to update the data entity.

Table 1. Meanings of business object verbs. (continued)

Role of business object	Meaning of verb
Response	Lists the verb specified in the associated request. For example, in a response, the Retrieve verb indicates that the connector obtained the attribute values from the application.

Note: The IBM convention is to use the format *business-object-type.verb* to indicate a particular type of business object with a particular verb. For example, Customer.Create is a Customer business object with the Create verb.

Business object attribute values

A business object contains **attribute values** that represent data fields associated with the data entity, such as Last Name, First Name, Employee ID, or Invoice Status.

Some attributes, instead of containing data, contain **child business objects** or **arrays of child business objects**. Figure 10 illustrates the structure of a Contract business object. The Line Item information in the contract is in an array of child business objects.

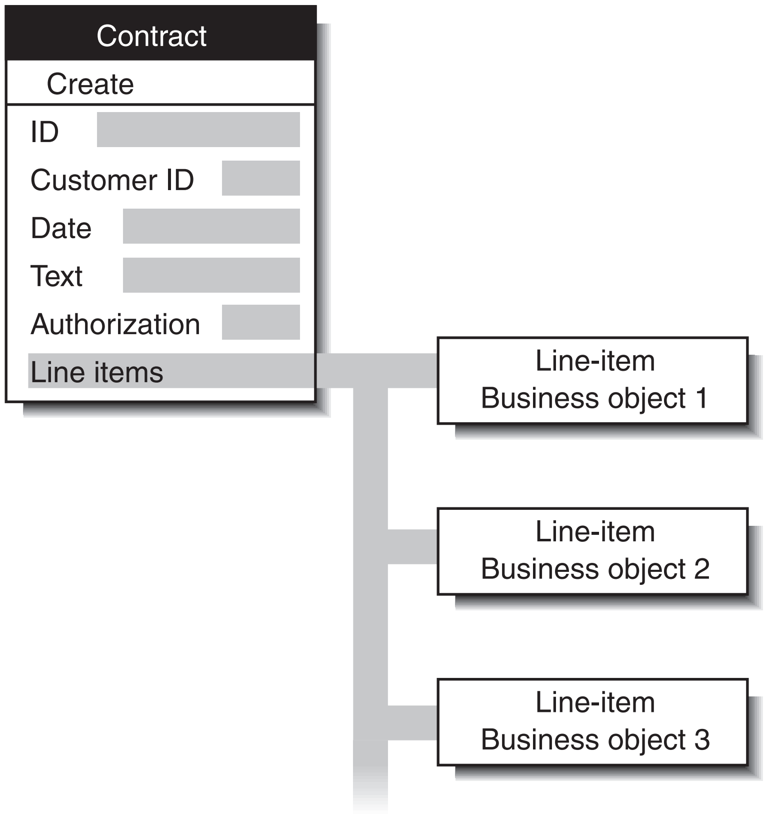

Contract
Create
ID
Customer ID
Date
Text
Authorization
Line items
Line-item Business object 1
Line-item Business object 2
Line-item Business object 3

Figure 10. Business object with child business objects.

A business object that contains child business objects or arrays of child business objects is a **hierarchical business object**. One whose attributes contain only data is a **flat business object**.

Types of business objects

The full-scale business integration system, which uses WebSphere InterChange Server (ICS) as its integration broker, includes two kinds of business objects: application-specific and generic. However, a WebSphere Business Integration adapter running on a message broker as the integration broker uses only application-specific business objects, not generic business objects. Therefore, all references to business objects throughout this book refer to application-specific business objects. Many of the books in the IBM WebSphere Business Integration (WBI) Server documentation set cover both environments and therefore refer to both types of business objects.

- **Application-specific business objects** reflect the data entity attributes and the data model of a specific application or other programmatic entity.
- **Generic business objects** contain sets of business-related attributes that are common across a wide range of applications, and are not tied to any specific application's data model. Generic business objects are not used by WebSphere Business Integration adapters running on message brokers but are discussed in the books that are included in both the WebSphere Business Integration adapters library and the WebSphere InterChange Server library.

When an application-specific component detects an application event such as an update, it retrieves the appropriate data entity from the application and transforms it into a business object.

Note: When documentation refers to a business object whose name includes an application name, such as Clarify_Contact or Oracle_Customer, it refers to an application-specific business object. A Clarify_Contact business object, for example, contains the set of information that the Clarify application stores about a contact. In another application, a contact entity might store a somewhat different set of information, store the information in a different order or format, or have a different name.

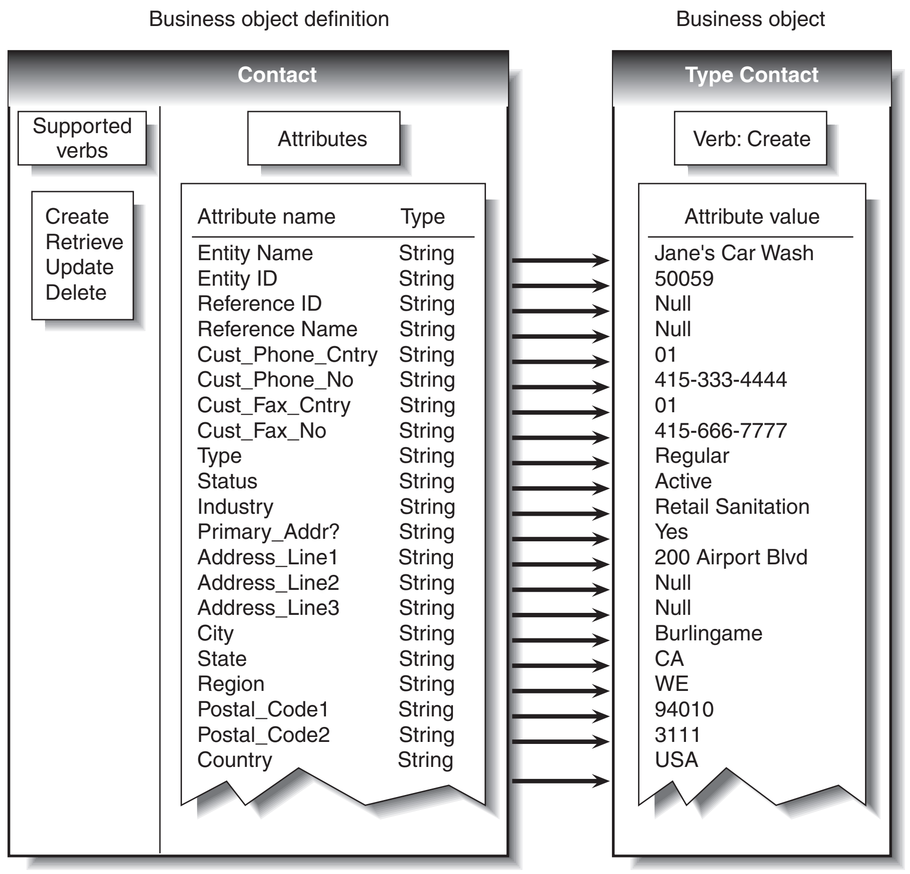
After an application-specific component has built a business object, it sends the business object to the connector framework. The connector framework calls the data handler to convert the business object to a WebSphere MQ message to be dispatched to the integration broker.

Business object definitions and business objects

Chapter 1, "Overview of WebSphere Business Integration adapters," on page 3, introduced business objects but only mentioned briefly the distinction between business object definitions and instances of the business objects themselves. Let's look more closely at that distinction now:

- A **business object definition** specifies the types and order of information in each entity WebSphere Business Integration Adapters for WebSphere MQ Integrator Broker handles, and the verbs that it supports. The local repository for the connector stores business object definitions.
- A **business object** is an instance of the definition, containing actual data. Business objects are created at runtime and not stored in the repository.

Figure 11 on page 19 illustrates the relationship between a business object definition and a business object.



Attribute name	Type	Attribute value
Entity Name	String	Jane's Car Wash
Entity ID	String	50059
Reference ID	String	Null
Reference Name	String	Null
Cust_Phone_Cntry	String	01
Cust_Phone_No	String	415-333-4444
Cust_Fax_Cntry	String	01
Cust_Fax_No	String	415-666-7777
Type	String	Regular
Status	String	Active
Industry	String	Retail Sanitation
Primary_Addr?	String	Yes
Address_Line1	String	200 Airport Blvd
Address_Line2	String	Null
Address_Line3	String	Null
City	String	Burlingame
State	String	CA
Region	String	WE
Postal_Code1	String	94010
Postal_Code2	String	3111
Country	String	USA

Figure 11. Business object definition and business object.

Components of a business object definition

In simplified terms, a business object is characterized by its type, its attribute values, and its verbs.

Overall, a business object definition is identified by its name. The name indicates the business object definition type, such as Customer, VantiveCase, or Invoice. A business object can also have application-specific information (metadata) that helps the application-specific component process it. All business objects also contain attributes and verbs, as the next sections describe.

Attributes

Attributes in a business object definition describe the values connected with the entity, such as Last Name, Employee ID, Case Number, Amount, or Date Initiated. At runtime, attributes are filled in with actual data.

For example, an Employee business object definition might contain attributes for the employee’s name, address, employee ID, and other relevant information. The attributes of a business object are analogous to the fields of a form or columns in a database table.

An attribute can also refer to a child business object or to an array of child business objects, such as an array of line items in a contract or part references in an invoice.

ObjectEventId attribute: The ObjectEventId attribute is a required attribute and is the last attribute in every business object.

When a connector publishes an event, it uses the ObjectEventId attribute of the business object definition to store a unique value that identifies the specific business object instance that is being created.

The value of the ObjectEventId attribute is generated and handled by the business integration system, which uses it to identify and track the flow of the specific event through the system.

Basic and compound attribute types: If an attribute's type is a basic data type, such as String, Boolean, Double, Float, or Integer, the attribute value is a discrete piece of data, such as the value of a field in a database. Examples include LastName, CustomerID, PartNumber, AssignedTo, and Price.

If an attribute's type is the name of another business object definition (a compound type), the attribute value is a child business object or an array of child business objects. Examples include Customer, Contract, and Oracle_Contact.

Attribute properties: A number of **properties** define the value that the attribute represents. Without showing all possible properties, Figure 12 illustrates the place of attribute properties in a business object definition.

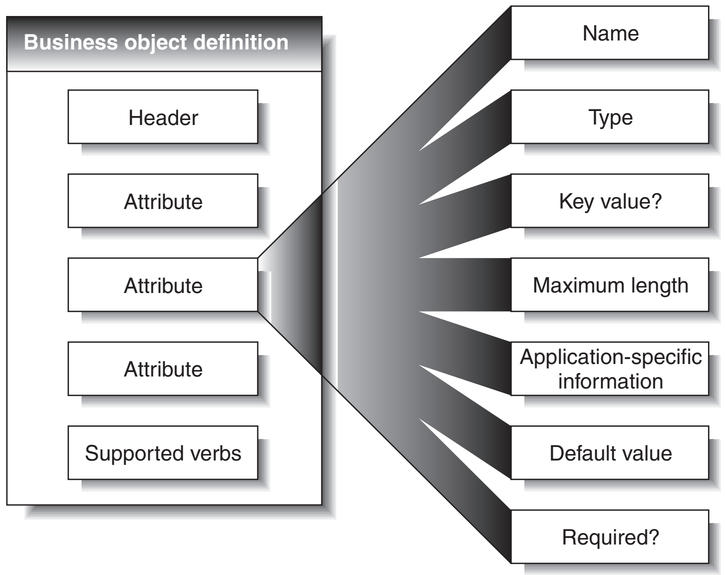


Figure 12. Attribute properties.

The set of properties for a particular attribute depends on whether the attribute type is basic or compound; that is, an attribute's properties differ depending on whether the attribute refers to a single unit of data or to a child business object.

Verbs

Verbs indicate actions on the data in the business object. A business object definition contains a list of verbs; a business object contains only one verb.

The most common verbs associated with business object definitions are Create, Retrieve, Update, and Delete.

The meaning of a verb differs according to the role of the business object. The verb can describe an application event, make a call, make a request, or identify the result of a previous request.

Note: Some applications do not support requests for hard deletes. For such applications, the business integration system performs the equivalent logical deletion, which is usually an update to inactive status. Furthermore, even if an application supports hard deletes, you can configure the business integration system so that it converts Delete verbs to Update verbs when sending requests to that application.

A closer look at business objects

A business object contains the data that an application-specific component moves into or out of a particular application. Therefore, each business object definition reflects the application's data model and the application-specific component's access method.

Even when two application-specific business objects refer to similar application entities, differences appear in the way that attributes are organized and in the application-specific information for them.

Attribute organization

Applications often organize the same information in different ways. For example, Application A stores a telephone number and fax number for a contact in four fields, but Application B stores the same numbers in two fields.

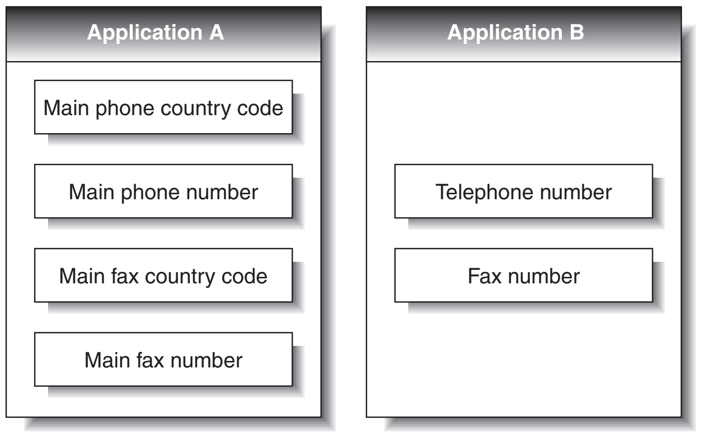


Figure 13. Telephone data in two applications.

The business object definitions for the Application A business object and the Application B business object have different attributes to reflect this difference.

Application-specific information

Business objects also differ because each can optionally contain built-in processing instructions for its application-specific component. Referred to as application-specific information (or metadata), it can consist of any information that the application-specific component needs to process the business object.

A business object definition can have application-specific information that applies to the entire business object, to each attribute, and to each verb. At each place where application-specific information appears in a business object definition, it provides information that the connector uses in its interactions with the application.

Application-specific information for a business object

Application-specific information for the business object provides information that the application-specific component uses when processing the business object as a whole.

Application-specific information for an attribute

Often, application-specific information that applies to an attribute identifies the attribute value's location in the application. The application-specific component uses this identifier when building API calls to the application to retrieve or enter the attribute value.

Application-specific information takes different forms for different applications. Sometimes the application-specific component can reference the attribute location by means of the application's form and field names; other times the reference is more complex.

Table 2 provides examples of parameters that might be included in an attribute's application-specific information. These parameters would be relevant only to a business object that represents data in a database table.

Table 2. Example name-value parameters for attribute application-specific information

Parameter	Description
`TN=`*`TableName`*	The name of the database table.
`CN=`*`col_name`*	The name of the database column for this attribute.
`FK=[..]fk_attributeName]`	The value of the Foreign Key property defines a parent/child relationship.
`UID=AUTO`	This parameter notifies the connector to generate the unique ID for the business object and load the value in this attribute.
`CA=`*`set_attr_name`*	The Copy Attribute property instructs the connector to copy the value of one attribute into another. If *`set_attr_name`* is set to the name of another attribute within the current individual business object, the connector uses the value of the specified attribute to set the value of this attribute before it adds the business object to the database during a Create operation.
`OB=[ASC|DESC]`	If a value is specified for the Order By parameter and the attribute is in a child business object, the connector uses the value of the attribute in the `ORDER BY` clause of retrieval queries to determine whether to retrieve the child business object in ascending order or descending order.
`UNVL=`*`value`*	Specifies the value the connector uses to represent a null when it retrieves a business object with null-valued attributes.

A single attribute's application-specific information might combine several of the example parameters listed above. This example uses semicolon (`;`) delimiters to separate the parameters:

```
TN=LineItems;CN=POid;FK=..PO_ID
```

The application-specific information in this example specifies the name of the table, the name of the column, and that the current attribute is a foreign key that links the child business object to its parent.

In exceptional cases, application-specific information for attributes is unnecessary. For example, some applications provide very direct and easy to use designations for units of data. Imagine that an application identifies sample fields as Table 3 illustrates.

Table 3. Sample application identifiers

Attribute	Application's identifier for the field containing the value
Customer ID	XCustomerID
Customer name	XCustomerName
Status	XStatus
Industry	XIndustry

In the example that Table 3 illustrates, it is easy for the application-specific component to associate an attribute with its identifier in the application because the rules for conversion are so regular: add the X or subtract the X. Therefore, the attributes in business objects for this application may not need application-specific information.

Application-specific information for verbs

A business object definition can include application-specific information for each verb that it supports. The application-specific information tells the application-specific component how to process the business object when that verb is active.

Ways to create or modify business object definitions

Each connector requires a set of business object definitions to define the data that is to be communicated to other applications. When the application-specific component is required to send data to the integration broker, it instantiates a new business object from one of the business object definitions it supports. One step in the process of configuring a connector is to select the business object definitions to be supported. First, however, you need to create or otherwise generate business object definitions for the application.

Creating business object definitions

There are several ways to construct or obtain business object definitions for an application.

- If an Object Discovery Agent (ODA) exists for your application, you can use it to build business object definitions. An ODA examines the structure and organization of the application's stored data and constructs business object definitions based on what it finds. If an ODA does not exist for your application, you can use the Object Discovery Agent development kit (ODK) to build an ODA.
- You can use the Business Object Designer tool to create business object definitions, either by modifying those generated by an ODA or by constructing them from scratch.

Other resources

The *Business Object Development Guide* provides detailed information about creating business object definitions.

In addition, many adapters include sample business objects. If samples are included, they are located in the product directory under:

Windows:

`\connectors\`*`ConnName`*`\Samples`

UNIX:

`/connectors/`*`ConnName`*`/Samples.`

Modifying business object definitions

You might need to modify a business object definition for several reasons - to capture additional application data, to stop collecting data found to be unnecessary, or to respond to changes to another application. The Business Object Designer tool, described in the *Business Object Development Guide*, is the most convenient way to make these modifications.

Chapter 3. Connectors

A connector mediates between an application and the integration broker on a local network. It can be specific to an application—such as SAP R/3, version 4—or to a data format or protocol, such as XML or WebSphere MQ. It consists of an application-specific component and a connector framework.

All connectors share certain common behaviors, differing only in the manner in which they interact with applications and with business objects. This chapter is an introduction to both the common behavior of connectors and to the areas in which they differ. It includes the following sections:
- "Connector startup"
- "Event notification"
- "Request processing" on page 31
- "Guaranteed event delivery" on page 31
- "Connector configuration" on page 36
- "Connector development" on page 37

In some environments, connectors are "black boxes"; you can simply install, configure, administer, and use a connector without much concern for its internals. If you need to create a custom connector, however, you need more detailed knowledge of connector behavior. For information on creating or modifying a connector, refer to the *Connector Development Guide for Java* or the *Connector Development Guide for C++*.

Connector startup

A connector must be explicitly started using a startup script which can be invoked from the command line or from a Windows shortcut.

Each connector has a local repository, which holds the connector's configuration file and a separate XML schema document for each business object definition. The repository is a directory in the local file system where the application-specific component is installed.

During startup, the connector does the following:
1. Loads the supported business object definitions and configuration properties from its local repository.
2. Connects to the application.

Note: An integration broker need not be running when you start a connector. However, no data can be transferred until the integration broker is active.

Event notification

A connector whose application provides triggering events must learn about those events and send the associated data to the integration broker. Figure 14 on page 26 illustrates a connector's interactions with respect to event notification.

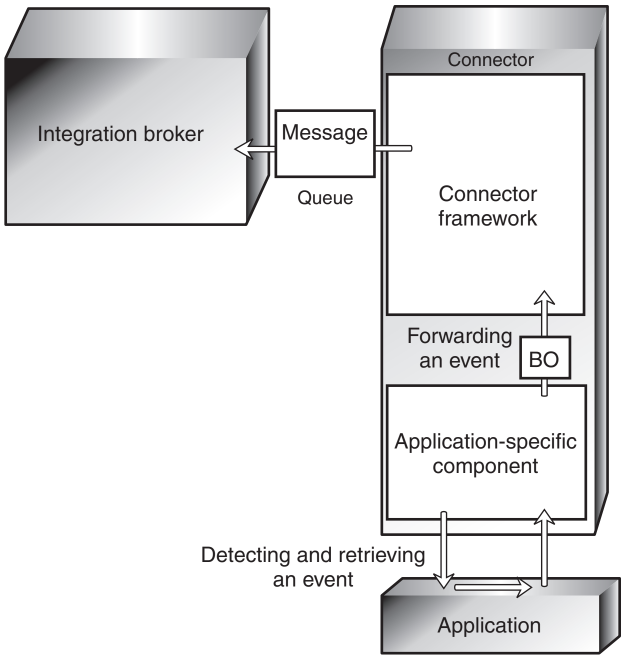


Figure 14. Event notification in a connector.

The ways in which application-specific components detect and retrieve events differ from one connector to another. However, the way in which application-specific components send events to the connector framework, and the way in which the connector framework deliver those events to the integration broker, is standard across all connectors.

The following subsections describe general concepts regarding the operation of most connectors, including:

- How connectors use application event-notification mechanisms
- How connectors detect and process events.

This discussion is not intended to describe the specific implementation of any particular connector.

Setting up the application's event-notification mechanism

To a connector, an application event is any operation that affects the data of an application entity that is associated with a WebSphere Business Integration adapters business object definition. There are other types of events in applications; for example, a mouse click is an event to an application's window system or forms interface. The connector, however, is interested only in a pre-defined subset of the data-level events that create, update, delete, or otherwise affect the content of the application's data store.

Some applications explicitly trap and report events, providing user-friendly event management and configurable event text. Other applications, without a concept of discrete, reportable events, might silently update their databases when something happens. WebSphere Business Integration adapters provide connectors for both types of applications.

For most connectors, some application configuration is needed to set up an **event-notification mechanism** for the connector's use. An event-notification mechanism maintains an ordered list of operations that take place in the application. It might have the physical form of an application event queue, an e-mail inbox, or a database table.

What types of event-notification mechanisms do connectors use? The next sections illustrate some general approaches.

When applications have event support

If an application is event-based, it probably has an event-notification interface for use by client applications such as connectors. The application might also permit you to configure the text of the event report. For such applications, setting up the connector’s event-notification mechanism is a normal application setup task.

For example, imagine that an application lets you install a script that executes when a particular type of event occurs and that the script can place a notification in an event inbox. To install the connector for that application, you create a user account for the connector, write or obtain scripts for handling the events you want to track, install the scripts, specify the type of event that triggers each script, and create the inbox. When you are done, the application-specific component periodically retrieves the inbox contents to check for new events.

Figure 15 illustrates an application configuration that includes an event inbox.

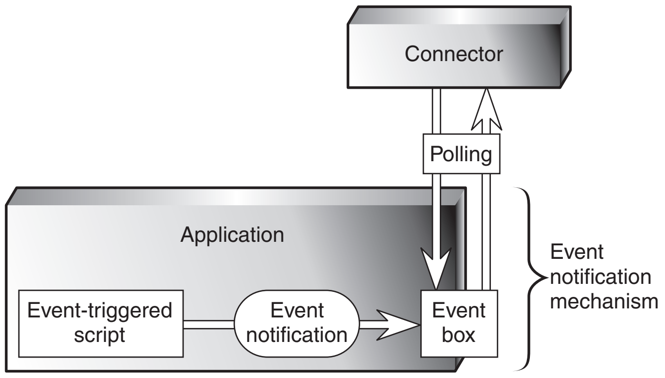


Figure 15. Example: Using an event inbox for event notification.

Another application might have an internal workflow system that can generate mail messages or write to an event queue when a particular operation occurs. Figure 16 on page 28 illustrates an application that has its own business object repository where business objects and events are defined. In the figure, Customer is a business object and Create, Delete, and Update are the types of events associated with it.

When a business object event such as Customer.Update takes place, the event is sent to the workflow system, which places an entry in an event table in the application database.

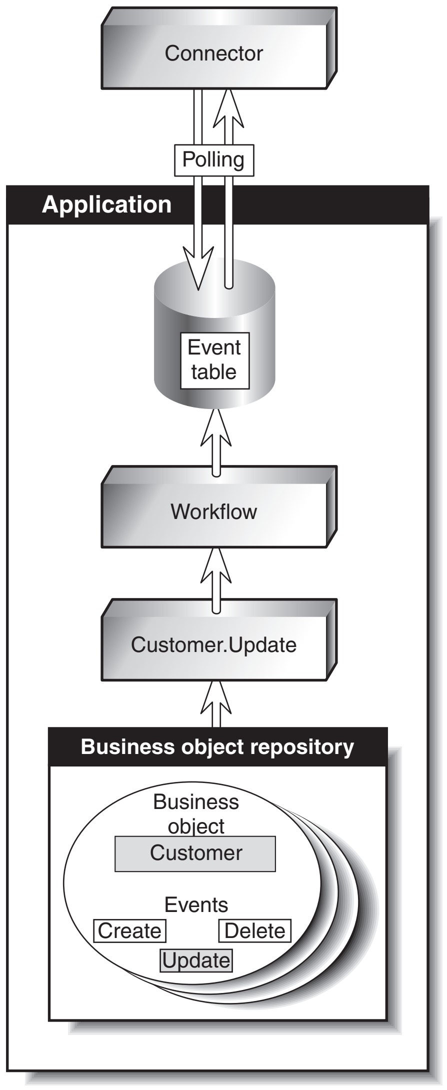


Figure 16. Example: Using application workflow for event notification.

When applications lack event support

The preferred method for a connector to interact with application events is through the application's API, which provides a framework that enforces the application's data model and logic. However, some application APIs do not provide native support for event notification.

One way that a connector can receive event notifications from such an application is to interact with the application database. For example, you can set up a trigger on an Employee table that detects updates to the rows. When an update occurs, the trigger inserts information about the update into a table, created when you deploy the connector. Each new row that appears in the event table represents an event notification. The connector can use SQL queries to retrieve new events from the table.

Figure 17 on page 29 illustrates this approach.

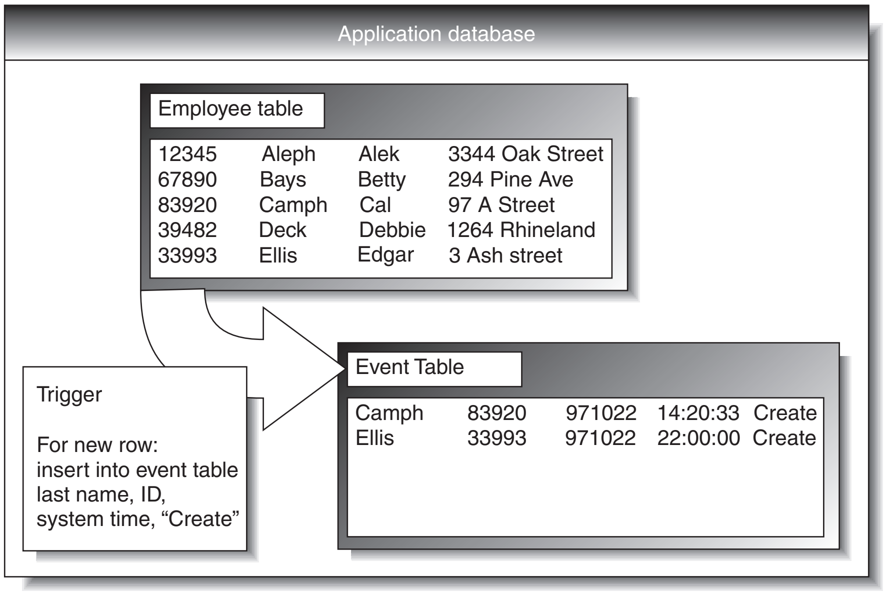

Application database
Employee table
12345 Aleph Alek 3344 Oak Street
67890 Bays Betty 294 Pine Ave
83920 Camph Cal 97 A Street
39482 Deck Debbie 1264 Rhineland
33993 Ellis Edgar 3 Ash street
Trigger
For new row:
insert into event table
last name, ID,
system time, "Create"
Event Table
Camph 83920 971022 14:20:33 Create
Ellis 33993 971022 22:00:00 Create

Figure 17. Example: Using the database for event notification.

In Figure 17, the application database has a trigger on the creation of records in the Employee Table. Each time the application inserts a new record, the trigger creates a row in the event table. The row contains the key values of the new employee record (last name and employee ID), the system time, and the event type, Create.

Detecting an event

A connector's application-specific component learns about application events through its event-notification mechanism, the most common of which is polling for new events in the event store. The polling method is specific to the application, based on the event- notification mechanism that the connector uses.

Polling is configurable. When you use the Connector Configurator tool to configure a connector, you can:

- Adjust the frequency with which the application-specific component polls the application
- Specify the hours during which the application-specific component polls the application.

For most connectors, you can also specify the number of events to be processed per poll call.

An application-specific component need only poll an application if another application is interested in that application's events. If a particular application is not the source of events, you can stop the application-specific component from polling by setting its polling frequency to "no" using the Connector Configurator. To learn more about the Connector Configurator, refer to either of the Connector Development Guides or to the adapter user guide for the adapter you are deploying.

Processing an event

After detecting an event, the connector's application-specific component:

- Associates the application event with a business object definition and creates an instance of that business object

- Sets the verb and key value attributes in the business object
- Retrieves application data and populates the business object’s attributes
- Sends the business object to the connector framework
- Archives the event (optional).

Associating an application event with a business object definition

When an application-specific component retrieves an event, it must determine which business object definition and verb represent the event.

The application-specific component uses the event text to associate the event with a business object definition and verb, as Table 4 shows.

Table 4. Event text and business object formation

Type of data in the application event	Examples	Use
Application entity type	Customer, Part, Item	Determining the associated business object definition
Operation that occurred	Create, Update, Delete	Determining the active verb of the business object

For example, a connector can associate the following event text with an Employee.Create business object:

```
1997.10.19.12:50.22 employee created lname="como" id="101961"
```

From left to right, the event text consists of:

- A time stamp that helps to uniquely identify the event
- The application entity “employee”
- The operation “created”
- The employee’s last name and ID, unique identifiers (key values) with which the application-specific component can retrieve the rest of the employee information.

Note that this example is simple; other types of event text might require more processing by the application-specific component.

Building an application-specific business object

If the connector’s been configured to support the business object definition for the event, the application-specific component builds a business object, uses a key value to retrieve application data, and fills in the business object with the application data. “Business object construction and deconstruction” on page 33 describes the process of building a business object.

Sending the application-specific business object to the connector framework

The application-specific component sends the business object to the connector framework without needing to know the identity of the application that will receive the information carried in the business object.

Archiving events

Application event archives are useful for troubleshooting and record-keeping. An event archive contains status information about each event, such as:

- Successfully sent to the integration broker

- Processing failed

If an application provides an event archiving feature, the connector generally uses it. A connector for an application that does not support event archiving might have its own event archive. For example, if a connector’s event-notification mechanism is like the database mechanism illustrated in Figure 17 on page 29, a database trigger could copy deleted events to an archive table. that you create when you deploy the connector.

Guaranteed event delivery

Guaranteed event delivery ensures that critical events, such as financial transactions, are processed correctly, regardless of any service interruptions that might occur. This feature enables the connector framework to guarantee that each event is detected and transmitted only once from the source connector’s event store to the destination connector’s request queue.

Without guaranteed event delivery, a small window of possible failure exists between the time that the connector detects an event and the time all necessary processing has completed. If a failure occurs in this window, the event has been sent but its event record remains in the event store. When the connector restarts, it finds this event record still in the event store and treats it as a new event, causing the event to be sent twice.

For more information about guaranteed event delivery and to learn how to enable it for the adapter you are deploying, refer to the adapter’s user guide.

Request processing

The integration broker also sends requests, in the form of WebSphere MQ messages, to the connector framework. A request can ask the destination application to do either of the following:
- To retrieve business data and return it to the integration broker.
- To update the application’s data store.

For example, the integration broker might send a connector a request message to delete a contract, update a part, or create a customer.

When the connector framework receives an integration broker’s request, it converts the message into a suitable business object and forwards it to the application-specific component. For example, if the integration broker sends a request to delete a contract, the application-specific component receives the request in the form of a Customer.Delete business object. The application-specific component translates the business object into an application request—typically a set of calls to the API—and then returns the results, if needed.

Figure 18 on page 32 illustrates a connector’s interactions with respect to handling messages from the integration broker.

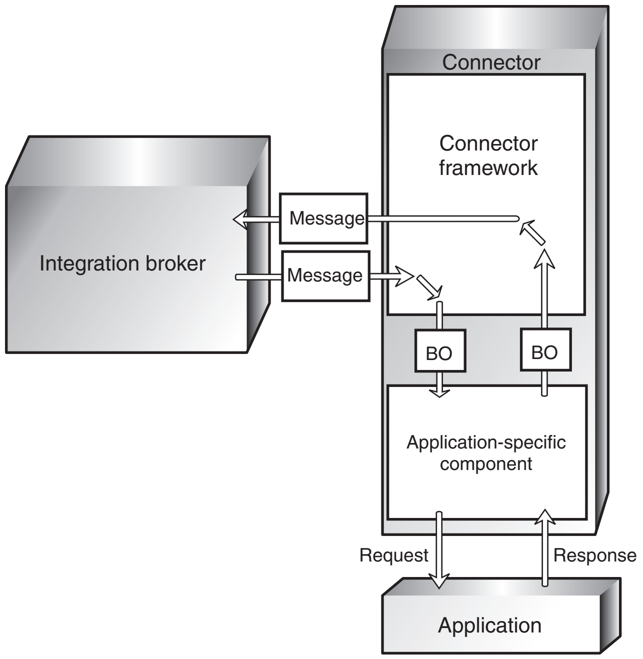


Figure 18. Connector interactions for request handling.

When an application-specific component receives a request, it determines how to process the request based on three types of information:

- The verb of the business object
- Metadata that is contained in the business object definition itself and used in the construction and deconstruction of the business object
- Application-specific information for the verb.

These factors are described in the topics that follow.

Verb-based processing

A connector's application-specific component responds to the Create, Retrieve, Update, or Delete verb in a request according to the logic and API of its application. The application-specific components of different connectors might handle the same type of request differently, although the result is logically the same.

For some connectors, only one method is required for performing operations on a business object, regardless of what verb the request contains. But for many connectors, each verb requires a different method.

When an application-specific component receives a request, it invokes the method in the application that matches the business object's active verb. For example, when a application-specific component receives an AppAEmployee.Update business object, it invokes the Update method on the AppAEmployee object. The Update method interacts with the application in order to perform the update.

Figure 19 on page 33 illustrates some verb handling methods.

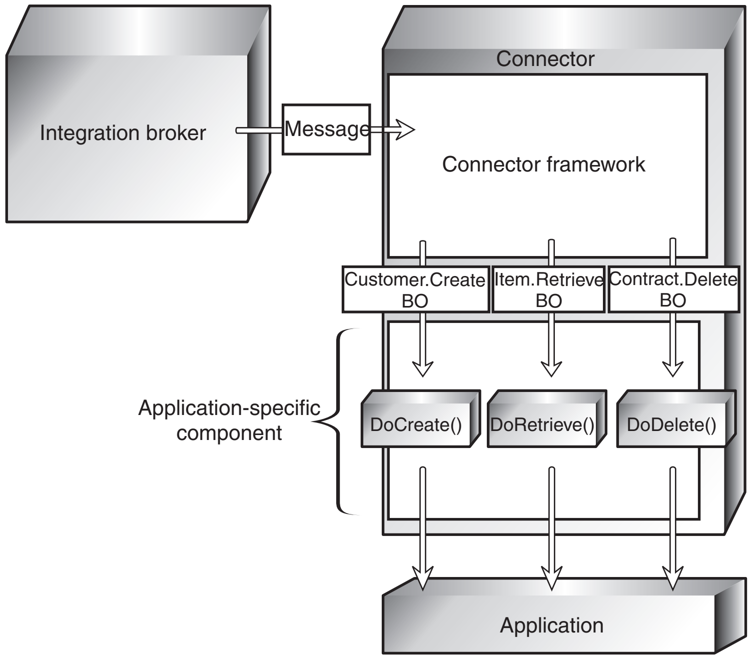


Figure 19. The processing of requests.

When the connector in Figure 19 receives a Customer.Create, Item.Retrieve, or Contract.Delete request, it invokes its DoCreate(), DoRetrieve(), or DoDelete() method, respectively. Each method builds the appropriate commands for the application to perform the specified operation.

Business object construction and deconstruction

A connector's application-specific component accomplishes its event notification and request-handling tasks by constructing and deconstructing business objects:

- When an application-specific component detects an event that it must send to the integration broker, it constructs a business object that represents the event.
- When an application-specific component receives a business object that represents a request from the integration broker, it deconstructs the business object to create an application request.

Business object metadata and connector actions

A connector's transformation of an application event to a business object and from a business object to an application request is driven by data definitions (**metadata**) that are defined when a business object is designed.

Application-specific components and business object metadata are designed to work together. The design of an application-specific component and its business objects is analogous to the design of a computer device in which certain functions can be implemented by either the software or hardware. The developer considers performance, extensibility, and other issues to decide where to implement key features.

Business object definitions include properties that specify the types, sizes, and default values for attributes. They also include application-specific properties that contain instructions to the application-specific component on how to process the business object.

For example, recall that Appendix G, "Using Visual Test Connector," on page 159, presents some examples of application-specific information for the attributes of a business object that represents a customer. Figure 15 on page 27 shows some of those examples.

Table 5. Example name-value parameters for application-specific information associated with attributes.

Parameter	Description
`TN=`*`TableName`*	The name of the database table
`CN=`*`col_name`*	The name of the database column for this attribute.
`FK=[..]fk_attributeName]`	The value of the Foreign Key property defines a parent/child relationship.
`UID=AUTO`	This parameter notifies the connector to generate the unique ID for the business object and load the value in this attribute.
`CA=`*`set_attr_name`*	The Copy Attribute property instructs the connector to copy the value of one attribute into another. If *`set_attr`*`_name` is set to the name of another attribute within the current individual business object, the connector uses the value of the specified attribute to set the value of this attribute before it adds the business object to the database during a Create operation.

When processing a business object, the application-specific component reads the definition and uses the application-specific information to build an application request. For more information on business objects, see the *Business Object Development Guide*.

Because the application-specific component is metadata-driven, its actions are controlled by application-specific information and other metadata in the business object definition. It does not have hard-coded instructions for each type of supported business object. Being metadata-driven gives the application-specific component the flexibility to automatically support any new or changed business object definition as long as the corresponding application data can be accurately described by the connector's metadata syntax.

An example of business object construction

The following process describes how an application-specific component creates a business object from its definition:

1. Obtains the business object definition from its local repository and uses it to create a business object instance.
2. The application-specific component loops through the business object instance attribute by attribute, using application-specific information to prepare an API call or build a query to obtain the application entity.
3. The application-specific component sends the request to the application and retrieves the results.
4. The application-specific component loops through the results, using the value of AppSpecificInfo to determine which retrieved value represents each business object attribute.

Figure 20 on page 35 is an example of an application-specific component that is building a business object from the definition. The application-specific component has retrieved an application event involving an item whose key value, the item number, is 123. The application-specific component must build an Item business object from the business object definition, which contains four attributes: Group, Description, Price, and ItemNum.

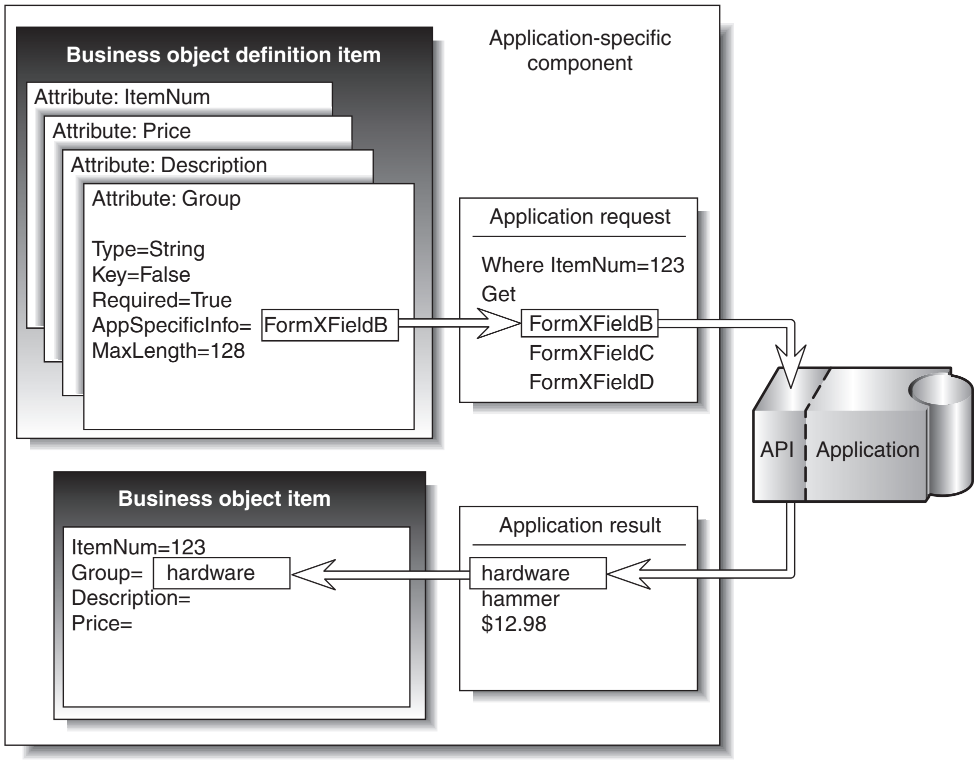


Figure 20. Building a business object in a connector.

Using the item number, 123, to identify the item, the application-specific component retrieves the values of the remaining attributes. Application-specific information identifies the form and field identifier for the required data.

For example, FormXFieldB identifies Group data. The application-specific component requests the value of Field B in Form X for item ID 123. The application-specific component then uses the returned value, “hardware,” to fill in the value of the business object’s Group attribute.

The process of deconstruction works in the opposite way. The application-specific component uses the business object definition to determine how to make an application request from the data contained in the business object that it received.

Application-specific information for verbs

Each verb in the business object definition can have application-specific information associated with it. The content of the verb’s application-specific information is unique to the particular connector. The application-specific information for the verb provides the connector’s application-specific component with additional instructions for processing the business object.

For example, application-specific information for the Retrieve verb in a business object definition might supply special input arguments to the Retrieve method in that application-specific component.

As an example, suppose that the MyApp application has three forms in which information about InventoryItem appears:
- InventoryItem-New
- InventoryItem-Change
- InventoryItem-Remove.

When the application-specific component for MyApp performs an operation on an inventory item, it must reference the correct form for that operation. In the InventoryItem business object definition, the application-specific information for the verb can be used to store the form name.

The combination of verb-specific methods and application-specific input to those methods gives an application-specific component unique instructions for processing.

Connector configuration

Before a connector can be used, you must use the Connector Configurator to define a configuration file that contains:
- The business objects that the connector supports
- The configuration properties for the connector.

There are two types of connector configuration properties, **standard properties** and **application-specific properties**.

Standard properties, which apply to all connectors, specify information such as:
- The integration broker used with the connector
- The location of the connector's local repository
- The name of the queue manager that manages the queues used by the connector.

Appendix C, "Standard configuration properties for connectors," on page 109 lists the standard configuration properties for connectors working with WebSphere message brokers.

Application-specific properties specify values that a particular application-specific component needs to establish a session with the application. They also direct certain aspects of the application-specific components processing behavior. Here are examples of application-specific properties for various connectors:
- The name or IP address of the machine running the application
- The name of the application database
- The login user ID and password the connector needs to use to access the application
- The name of the event inbox
- The number of events to be retrieved per polling event.

Some connector configuration properties can also be set at the command line when you start up the connector. Properties set on the command line override the values set in the connector's configuration file.

For more information on configuring a connector, refer to "Configuring the connector" on page 91. To learn more about starting up a connector, see "Starting a connector" on page 57.

Connector development

To modify or create a connector, you use class libraries, header files, and samples to create an application-specific component. You then use the Connector Configurator to create and modify business object definitions and create connector repository definitions.

Connector development involves defining the relationship between the application-specific component and a particular application. The actual coding of a application-specific component is usually a fairly straightforward process. The most challenging tasks are:

- Designing the application's event-notification method
- Defining business object definitions
- Defining the relationship between the business objects and the application objects.

For detailed information on connector architecture, modifications, and development, refer to the *Connector Development Guide for Java* or the *Connector Development Guide for C++*.

The next chapter, Chapter 4, "Data transport and the integration broker," on page 39, provides a more detailed look at the mechanisms and protocols used to transfer information among the business integration system's components.

Chapter 4. Data transport and the integration broker

When a WebSphere Business Integration adapter uses a message broker as its integration broker, WebSphere MQ queues and standard Java Messaging Service (JMS) software are used as the communication transport mechanism between the connector framework and the integration broker. Connectors read messages from and write messages to pre-defined queues, managed by the WebSphere MQ Queue manager. The integration broker communicates with the connector's connector framework using the same methods.

This chapter describes in detail the messaging interfaces and protocols used to exchange data among applications in the WebSphere business integration system. It includes the following sections:
- "The role of the integration broker"
- "Asynchronous data transport"
- "Synchronous data transport" on page 40
- "Interfaces for message exchange" on page 40

The role of the integration broker

In a business integration system that uses a WebSphere message broker, the integration broker has two jobs: routing messages between applications and processing messages using message flows. Specifically, the integration broker:
- Receives a message from the outbound queue of the sending application's connector.
- Passes the message to the message flow for processing. A single message flow, defined for each queue, processes all messages placed on that queue. The message flow, however, can include different instructions for processing each type of message it expects to handle. Each message contains a header with information about the kind of data it carries. The message flow uses the header information to determine which set of processing instructions to use for the message.
- Transfers the message to the inbound queue of the destination application's connector.

Communication between the integration broker and a connector can be **asynchronous** or **synchronous**.

Asynchronous data transport

Programs that use asynchronous messaging transport need not establish connections or wait for messages (discrete units of data); each program sends and receives messages by interacting asynchronously with the messaging service. The messaging service provides guaranteed delivery, storing the message if the destination program is unavailable and retrying until it is available. Asynchronous messaging is a useful communication protocol when an immediate response is less important than reliable delivery.

Synchronous data transport

Although none of the pre-packaged adapters currently use synchronous data transport, this option is available for customized connectors.

Programs that use synchronous messaging transport post request messages to the integration broker on a synchronous request queue and receive response messages from the integration broker from a synchronous response queue. A correlation ID on the response message identifies the request message to which it is responding. Response messages generally consist of business object messages and a status indicator that shows whether the request was processed successfully.

Interfaces for message exchange

This section describes the messaging interfaces used by the connector framework and the integration broker to transmit messages and the information needed to process them.

Several distinct types of messages are exchanged within the business integration system. Essential information needed to identify the different types of messages, as well as process and route them correctly, is stored in the message header and the message descriptor of each message. Message flows you create for your business integration system use the information presented below to recognize and correctly manage messages they are called upon to process.

The following types of messages are passed:

- Event delivery messages are sent by the connector framework to the WebSphere message broker to notify of an event in the source application.
- Request messages are exchanged between the connector framework and the WebSphere message broker to convey a request for data.
- Response messages are exchanged between the connector framework and WebSphere message broker to reply to a request for data.
- Administrative messages are exchanged between the connector framework and WebSphere message broker to convey administrative commands.

Message formats

Messages exchanged between the connector framework and the integration broker are formatted by the data handler, based on:

- The WireFormat standard property in the connector's configuration file
- The XML schema detailing the message body format
- The content of the message: a business object or an administrative message
- The origin and destination of the message.

Each message contains three components: a message descriptor (MQMD), a message header (MQRFH2), and a message body.

Message descriptor

The WebSphere MQ message descriptor (MQMD) contains the message ID and includes information needed for processing the message.

Message header

The MQRFH2 message header carries JMS-specific data that is associated with the message content. It can also carry additional information that is not directly associated with JMS. The message header contains the following folders:

- The <mcd> folders contains properties that describe the "shape" or "format" of the message. For example, the Msd property describes the format as being Text, Bytes, Stream, Map, Object, or "Null".
- The <jms> folder is used to transport JMS header fields, and JMS properties that cannot be fully expressed in the MQMD. This folder is always present in the messages implemented using JMS, which are sent by the connector framework. However, in the business integration system, this folder is irrelevant and is omitted by the integration broker when sending messages to the connector framework.
- The <usr> folder is used to transport any application-defined properties associated with the message. This folder is only present if the application has set some application-defined properties. In the business integration system, this folder is used to send return status information in a response message. The tables below identify the types of messages that require this folder.

Message body

The message body is formatted as specified by the XML schema specified for the message. In order for the data handler to find and use the correct XML schema for formatting a message, the following three names must be the same:

- The name of the XML schema stored in the connector's repository
- The name of the XML schema imported into the WebSphere message broker's message repository and saved as a message set definition.
- The value of messagetype in the message's MQRFH2 message header.

The message formats and the settings for particular properties for the different types of messages exchanged by the connector framework and the WebSphere message broker are listed in Appendix A, "WebSphere MQ message formats," on page 99..

Message queues

The WebSphere MQ queues that need to be defined and configured for use with the connector are described below.

Required types of queues

Separate sets of WebSphere MQ message queues are used for transporting business object messages and administrative messages between the connector framework and the WebSphere message broker. You must define queues with the following properties:

- **DeliveryQueue:** Delivers event delivery messages from the connector framework to the WebSphere message broker.
- **RequestQueue:** Delivers request messages from the WebSphere message broker to the connector framework.
- **FaultQueue:** Delivers fault messages from the connector framework to the WebSphere message broker. The connector framework places a message on this queue when it is unable to place the message on the reply-to queue.
- **SynchronousRequestQueue:** Delivers request messages that require a synchronous response from the connector framework to the WebSphere message broker. This queue is necessary only if the connector uses synchronous execution.

- **SynchronousResponseQueue:** Delivers response messages from the WebSphere message broker to the connector framework sent in reply to a synchronous request. This queue is necessary only if the connector uses synchronous execution.
- **AdminInQueue:** Delivers administrative messages from the WebSphere message broker to the connector framework.
- **AdminOutQueue:** Delivers administrative messages from the connector framework to the WebSphere message broker.

During connector configuration, you specify the name of each queue as a standard property in the connector's configuration file.

Queue manager

The connector uses a single queue manager to manage all of its interactions with queues. The standard properties in the connector's configuration file contain the queue manager information needed by the connector at startup. The connector uses this information to establish a connection to the queue manager it will use to communicate with the WebSphere message broker.

The WebSphere business integration system supports several queue managers and queue configurations. The connector can communicate with the queue manager in any of the following modes:

- **Bindings mode:** The integration broker and the connector communicate directly with the queue manager, without using a TCP/IP connection. The queue manager and the connector must be on the same machine and must use the same queue manager. This is the default mode.
- **Bindings mode with remote queue definitions:** If the integration broker and the connector are installed on separate machines, with each machine running its own queue manager, the connector and the integration broker can still communicate with their respective queue managers using bindings mode but remote queue definitions are also needed.
- **Client mode:** Communication occurs through a client connection that uses TCP/IP as its underlying transport. If the queue manager and the connector are on the different machines, the connector is limited to using client mode.

For more information

To learn more about WebSphere MQ messages see *WebSphere MQ: Using Java.* To learn more about WebSphere MQ queues, see *WebSphere MQ: Intercommunication* and *WebSphere MQ: Script Command (MQSC) Reference*.

Part 2. Deployment and administration

Chapter 5. Planning your implementation

This chapter provides an overview of the planning required to implement the WebSphere business integration system. It includes the following sections:

- "Developing the business process interfaces"
- "Stages of an implementation"
- "Development tools" on page 49

Developing the business process interfaces

WebSphere Business Integration adapters provide configurable, modular elements that enable connectivity between enterprise applications. These modular elements, which include connectors and business objects, work together to form business process interfaces used to send data from one application to another.

A critical aspect of implementing a business integration system is to identify and develop the business process interfaces that are needed. A typical implementation will use multiple business process interfaces.

Each interface addresses a specific business task that needs to be integrated. For a simplified example, assume that an enterprise uses Application A as its system of record, but uses Application B for billing. The business problem is to integrate data exchanges for several types of business information between the two applications. One crucial type of business information is customer data--the customer's name, address, and other details need to be synchronized between the two applications so that when data changes in one, it also changes in the other.

To accomplish this requires two interfaces:

- Application A customersync to Application B
- Application B customersync to Application A.

The enterprise might also need to track and record the items for which the customer is billed. This requires additional interfaces:

- Application A itemsync to Application B
- Application B itemsync to Application A.

Each interface can be distinguished from others in terms of its source application in combination with the type of business data being exchanged. Each interface also has its own event-notification mechanism, trigger, and business object that initiate flows out of the source application.

Stages of an implementation

The implementation of a business integration system is performed in stages. The exact details and the nature and timing of deliverables produced in each stage may vary according to the organization that is performing the implementation. However, viewed at a high level, there are several broad stages that are used in any implementation of a WebSphere business integration system. These include:

- Discovery and assessment of requirements
- Evaluation of available components

- Design of new (custom) or extended components
- Development and configuration
- Validation
- Deployment.

Discovering and assessing business goals

This stage begins the implementation process by identifying the business goals for the project, the system requirements, and the overall scope of the development effort.

Discovery starts at a high level and moves to lower levels of detail. It starts with the following high-level questions:

- What are the specific business problems that need to be resolved for the enterprise?
- What enterprise-level business processes need to be integrated to resolve the business problem?

  Ask the following questions and others that might be applicable to the enterprise:
  - What are the applications involved in the interface and their versions?
  - Which ones are the source applications?
  - Which ones are the destination applications?
  - Which application is the system of record?
  - Are the applications in production or still being developed?
  - Are the applications developed in-house or are they packaged applications?
- What is the technology environment—including applications, databases, and APIs—in which the business processes need to be integrated?

  Determine the characteristics of the technology environment. Examine each of the following:
  - Database
  - Platform and operating system
  - APIs that exist for the applications
  - Location of all the application client and server platforms
  - Network environment
  - Application versions
  - Anticipated transaction volume.

To identify the interfaces needed for the implementation and the components that will be used, you will research information in lower levels of detail, identifying and describing the specific business processes that you intend to implement, the business logic and data transformations that are required, and details of the applications and databases that will interact. Your research may include the following information-gathering tasks:

- Identify and describe business processes that need to occur in order to solve the business problem. Ask the following questions:
  - What is a normal process flow in your business process?
  - What event initiates the transaction or data flow?
  - What applications are involved in the business process?
  - Which organizations own the business process, applications, and data?
  - What are the inputs and outputs?

  - Are there prerequisites/dependencies for the data?
  - Are there filtering requirements?
  - If there are multiple destination points, what determines where the data is sent?
  - Is the interface bi-directional?
  - What is the frequency of the transaction process?
  - Is there a time frame in which the transaction process needs to complete? Do other processes depend upon it?
  - What is the volume of data?
  - Is the interface real time or batch?
  - Is the interface synchronous (requiring a reply) or asynchronous (fire and forget)?
  - What is the error-handling procedure?
- Describe the structure of the data entity or entities exchanged between applications in the business process.
- Identify data transformations that are required between source and destination applications.
- Identify the programmatic events (things that happen in the process, such as routing logic and filtering logic) that occur or need to occur in the business process.
- Illustrate the business process flow with a flow diagram. The flow diagrams will help you analyze the functions that need to be performed. You will use them to create message flows for routing and transforming the data to be exchanged among applications.

Evaluating existing components and designing new ones

Evaluation and design are dependent upon the detailed information gathered during discovery.

When you have determined the detailed requirements of an interface and the integration components that it comprises, you are ready to evaluate existing components to see if any will meet your needs. You may find that for some requirements, components already exist and can be used as is, that for other requirements existing components need to be extended (revised according to your needs), and that for other requirements you will need to create new (custom) integration components.

Evaluate each component both individually and in terms of how it relates to other components in the overall interface. You cannot complete the design of one component until you have also begun the design of the components with which it interacts in the interface.

For detailed information about designing components, see the following books:
- *Business Object Development Guide*
- *Connector Development Guide for Java*
- *Connector Development Guide for C++*

Developing and configuring the business integration system

In this stage, you create any new integration components that are required and configure components for each business process interface that is to be implemented.

This is an iterative process that may require you to reconfigure components or revise their design. As you create and configure the components of an interface, you perform unit tests to determine that individual components function correctly. When an entire interface has been configured, you perform string tests to determine that all the components of an individual interface function correctly together.

Overall development flow

At a high level, it is recommended that integration components be developed in the following general order:

- Connectors

  Connectors and the business objects that they use interact directly with the application itself. For this reason, and because creation of a new connector usually takes more time than the creation of any other component, the connector should either be identified (if one already exists) or created first. If no custom connectors need to be created, you can start the development process with application-specific business objects. Before you decide what connectors need to be developed or extended, be sure that you have investigated which ones are already available from IBM, and that you understand the relevant licensing terms for the site.
- Business Objects

  Because other components will be dependent upon business objects, develop these first.

  You create a custom application-specific business object in multiple iterations, testing each iteration with the connector, and then adding functionality and re-testing. You configure the connector during the first unit test, and through iterations of development and unit testing, you will complete the implementation of the business object and the configured connector.

Sequence of tasks

Development and configuration is typically performed in a prescribed sequence of steps.

1. Develop custom connectors (if required)
2. Develop or extend application-specific business objects
3. Configure WebSphere MQ Integrator Broker to work with connectors
4. Configure connectors
5. Develop message flows
6. Unit test application-specific business objects
7. Configure connectors with business objects
8. String test the interface.

Validating the business integration system

During the validation stage, system testing is performed in a controlled test environment to ensure that all requirements identified during the Discovery phase have been met by the system design. System Testing during the validation phase includes functional, performance and regression testing (as required).

- Functional Testing - Ensures that all functional requirements of the system are met. A requirements matrix should be used to ensure that requirements are tested and acceptable.
- Performance Testing - Ensures that timing requirements such as throughput, response time, and latency are met through design or optimization.

- Regression Testing - As modifications are made to system design or configuration parameters during the validation phase, regression testing must be performed to ensue that other system functions and throughput capabilities have not been degraded.

Validation stage tasks include the following:
- Develop a requirements matrix containing all functional and performance requirements.
- Identify, designate and configure a system testing environment.
- Identify and develop a test data set to be use for system testing.
- Migrate the system implementation from the development environment to the test environment.
- Develop, coordinate, and obtain client concurrence for system testing plans and procedures. Assign specific tests for each test matrix item.
- Perform system, functional, and performance testing.
- Identify, control and resolve defects found while system testing.
- Perform regression testing (as required).
- Develop and publish a System Test Report.
- Obtain client acceptance of the developed system.

Deploying the business integration system

The objective of the deployment stage is to ensure that the developed and tested business integration system is implemented in the client production environment, optimized as required, and is production ready at the client site.

Deployment stage tasks include the following:
- Develop and obtain client acceptance for an overall cutover and deployment plan, ensuring uninterrupted continuing operations with the existing systems and processes.
- Migrate the developed and tested system from the test environment to the production environment.
- Obtain and document client unique business process and data conversion requirements for systems integration.
- Develop, implement and test modifications required to support unique production environment (i.e., addresses, passwords, etc.), requirements.
- Install and test business integration system software on client production system.
- Optimize business integration system for the production environment.
- Obtain client acceptance of the deployed system.

Development tools

WebSphere Business Integration adapters include several tools to assist you in creating and modifying connectors and business objects. These are listed and described in Table 6.

Table 6. WebSphere Business Adapters development tools.

Tool	Description	For more information
Adapter Development Kit	A separately-available kit that provides a set of class libraries and utilities with which you can develop connectors.	See the *Connector Development Guide for Java* and the *Connector Development Guide for C++*.

Table 6. WebSphere Business Adapters development tools. (continued)

Tool	Description	For more information
Connector Configurator	A graphical user interface used to configure connectors.	See the *Connector Development Guide for Java* and the *Connector Development Guide for C++*, or the adapter user guide for the connector you are deploying.
Business Object Designer	A graphical user interface used for creating business object definitions both manually and from Object Discovery Agents (ODAs).	See the *Business Object Development Guide.*
Object Discovery Agent Development Kit (ODK)	A set of APIs that let you create Object Discovery Agents (ODAs). ODAs identify business object requirements specific to a data source and generate definitions from those requirements.	See the *Business Object Development Guide.*
Visual Test Connector	A tool that simulates the activities of a connector to allow you to test whether you have developed your business integration interfaces correctly.	See Appendix G, "Using Visual Test Connector," on page 159

Note: If you attempt to launch one of the designer tools and experience an error about a class not being found, you must launch System Manager and then try to launch the designer tool again. System Manager does not have to remain running after the tool is initially launched, however.

Chapter 6. Installing WebSphere Business Integration adapters

This chapter describes how to install WebSphere Business Integration adapters and supporting software on a WebSphere message broker.

If you are upgrading WebSphere Business Integration adapters from a previous version, see Appendix H, "Upgrading WebSphere Business Integration adapters," on page 173 for instructions.

This chapter describes installation on both Windows and UNIX operating systems. You need only refer to the section that is appropriate for your operating system, as listed below:

- "Installing for Windows systems"
- "Installing for UNIX systems" on page 53

This chapter contains information specific to installing the supporting software (JDK and WebSphere MQ) on a WebSphere message broker. A separate document, the *Installation Guide for WebSphere Business Integration Adapters*, contains all the details necessary for installation of WebSphere Business Integration adapters. The sections of this chapter that describe installation for either Windows or UNIX systems refer you to the *Installation Guide for WebSphere Business Integration Adapters* at the appropriate point in the installation process.

Installing for Windows systems

This section includes the following topics:

- "Software Requirements"
- "Installing the JDK" on page 52
- "Installing WebSphere MQ" on page 52
- "Installing WebSphere Business Integration adapters" on page 53

Software Requirements

The WebSphere business integration system includes WebSphere Business Integration adapters and other components. WebSphere Business Integration adapters and some associated components are delivered on CD or through ESD (IBM Electronic Software Delivery). Other components used by the WebSphere business integration system must be obtained and installed separately.

Table 7 on page 52 lists the software requirements for the WebSphere business integration system.

Table 7. WebSphere business integration system software requirements for Windows systems

Software	Shipped with product
Windows 2000 (Professional, Server, or Advanced Server) with Service Pack 4. **Note:** Beginning with WebSphere Business Integration Adapter Framework 2.4.0, adapters are no longer supported on Windows NT.	No
One of the following message brokers:	
IBM WebSphere MQ Integrator Broker v 2.1.0 with the following: • CSD05 service pack. Download from `http://www.ibm.com/software/integration/mqfamily/support/summary/mqsib.html` IBM WebSphere MQ Integrator v. 2.1.0. Check with your IBM representative to determine if any service packs are required. IBM WebSphere Business Integration Message Broker v. 5.0 with the following: • CSD02 service pack. Download from `http://www.ibm.com/software/integration/mqfamily/support/summary/wbib.html`	No
Java Runtime Environment (JRE): IBM version 1.3.1 SR5.	Yes
Java Development Kit - only needed for developing custom Java connectors. IBM JDK version 1.3.1 SR5.	Yes
IBM WebSphere MQ version 5.3.0.2 with CSD05.	No
Browser: An HTML browser such as Microsoft Internet Explorer or Netscape Navigator is required for viewing the HTML documents. For the exact versions supported, refer to the instructions that can be downloaded from `http://www.ibm.com/integration/wbiadapters/library/infocenter`.	No

Installing the JDK

The JDK (Java Development Kit) is required only if you plan to develop custom Java connectors (the Java compiler is needed). For development on Windows systems, the JDK is included in the WebSphere Business Integration Adapter FrameWork. Refer to the *WebSphere Business Integration Adapters Installation Guide* for more information on installing the Adapter Framework.

Installing WebSphere MQ

IBM WebSphere MQ is the messaging software that enables communication between a WebSphere message broker and the adapters.

Refer to the following WebSphere MQ publications for installation and configuration information:

- *WebSphere MQ: Quick Beginnings*
- *WebSphere MQ: System Administration*
- *WebSphere MQ: Intercommunication*

Note: You can browse or download these documents from IBM's Web site at: http://www.ibm.com/software/mqseries.

For information about related WebSphere MQ books, see "WebSphere message broker publications" on page viii.

To configure WebSphere MQ to work with a WebSphere message broker, see "Configuring the message broker to work with the connector" on page 77..

Installing WebSphere Business Integration adapters

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

Installing for UNIX systems

This section includes the following topics:
- "Software Requirements"
- "Installing the JDK" on page 54
- "Installing WebSphere MQ" on page 56
- "Installing WebSphere Business Integration adapters" on page 56

Software Requirements

The WebSphere business integration system includes WebSphere Business Integration adapters and other components. WebSphere Business Integration adapters and some associated components are delivered on CD or through ESD (IBM Electronic Software Delivery). Other components used by the WebSphere business integration system must be obtained and installed separately.

Table 8 on page 54 lists the software requirements for the WebSphere business integration system.

Table 8. WebSphere business integration system software requirements for UNIX systems.

Software	**Shipped with product**
Operating system: • **AIX** - AIX 5L version 5.1 maintenance level 1 or 5.2 with maintenance level 4 • **Solaris** - Solaris 7.0 or 8.0 , with Patch Cluster released 7/23/03 • **HP-UX**- HP-UX 11.11 (11i,r=B.11.110306.4). June 2003 GOLDBASE 11i,r=B.11.110306.4 and June 2003 GOLDAPPS11i 11i,r=B.11.110306.4 bundles.	No
One of the following message brokers:	
IBM WebSphere MQ Integrator Broker v 2.1.0 with the following: • CSD05 service pack. Download from http://www.ibm.com/software/integration/mqfamily/support/summary/mqsib.html IBM WebSphere MQ Integrator v. 2.1.0. Check with your IBM representative to determine if any service packs are required. IBM WebSphere Business Integration Message Broker v. 5.0 with the following: • CSD02 service pack. Download from http://www.ibm.com/software/integration/mqfamily/support/summary/wbib.html	No
Java Runtime Environment: • **AIX** - IBM JRE version 1.3.1 SR5. • **Solaris** - Sun JRE version 1.3.1.7. • **HP-UX** - Sun JRE version 1.3.1.6.	Yes
Java Development Kit - only needed for developing custom Java connectors. • **AIX:** IBM JDK version 1.3.1 SR5. • **Solaris:** Sun JDK version 1.3.1.7. • **HP-UX:** Sun JDK version 1.3.1.6.	No
WebSphere MQ version 5.3.0.2 with CSD05.	No
Browser: An HTML browser such as Microsoft Internet Explorer or Netscape Navigator is required for viewing the HTML documents. For the exact versions supported, refer to the instructions that can be downloaded from `http://www.ibm.com/integration/wbiadapters/library/infocenter`.	No

Installing the JDK

The JRE includes the Java Virtual Machine (JVM), which is needed to run WebSphere Business Integration adapters. However, it does *not* include development tools, such as JavaC (the Java compiler). If you do *not* plan to create custom connectors, use the JRE.

The JRE contains the runtime component of the Java software, which the WebSphere business integration system requires to execute. The JRE is included in the WebSphere Business Integration Adapter Framework. Therefore, there is no need to install it separately.

If you plan to create custom connectors, install the full JDK, which you can download from the Sun or IBM Web site. For Solaris and HP-UX, you can download the JDK from http://java.sun.com/products/jdk/1.3. For AIX, you can download the JDK from http://www.ibm.com/developerworks/java/jdk.

To install the JDK on your system:

1. Navigate to the directory in which you want to install the JDK.

   ```
   cd /install_dir
   ```

   where the *install_dir* is the path of the directory that will contain the JDK software. You can install this software anywhere:

   - The JDK is normally installed in a subdirectory of the `/usr` directory. If the `/usr` filesystem has enough space, install the JDK there, in a directory such as `/usr/jdk1.3`.
   - You can create an extract directory for the JDK (such as /*home_dir*/jdk1.3) and move into this directory. You must create a symbolic link from the `/usr/jdk1.3` directory to this extract directory.
2. On either the Sun or the IBM Web site, locate the Web page to download the JDK.

   The JDK is usually located under the Java 2 SDK Standard Edition for UNIX.
3. Download and read the accompanying README file for how to download the JDK.
4. Download the JDK into your installation directory.

   IBM and Sun provide a self-extracting `tar` file that allows you to extract its contents in any directory you choose. This file usually has the format *JDK_version*.bin (where *JDK_version* is the version of the JDK to download). Consult the README file for how to extract the JDK software from this file.
5. Download any operating-system patches listed for the JDK that are appropriate for your version of the UNIX operating system.

   Before installing these patches, verify that they are not already installed on your system.
6. Create a link from your download directory (for example, the `/usr/jdk1.3` directory) to the Java directory.
   a. `rm /usr/java`
   b. `ln -s /usr/jdk1.3 /usr/java`
7. After the installation, make sure the `bin` directory of the JDK is included in the PATH environment variable.

   You can do this in either of the following ways:

   - Edit `/etc/profile`, which is read at every user's startup. Shells such as `ksh`, `bash`, and `sh` obtain the settings in the `/etc/profile` file.

     For example, if you installed JDK in the `/usr/jdk1.3` directory, edit the PATH entry of `/etc/profile` so that it appears as follows:

     ```
     PATH=/usr/jdk1.3/bin:$PATH
     export PATH
     ```
   - Edit the personal profile file in the home directory of the WebSphere business integration administrator.

     The name of this personal profile file depends on the particular shell that the WebSphere business integration administrator account uses. Changes made in this personal profile affect *only* the user logged in as the WebSphere business integration administrator.

     Use the syntax appropriate for the particular shell to include the JDK `bin` directory in the PATH environment variable. For example, if the WebSphere business integration administrator account uses the `sh` shell, you can edit the lines that add the JRE path (`/usr/java/bin`) to include the JDK path, as follows:

     ```
     PATH=/usr/jdk1.3/bin:/usr/java/bin:$PATH
     export PATH
     ```

Add these lines *after* the line that sources the `CWSharedEnv.sh` file in the personal profile file for your WebSphere business integration administrator account.

Installing WebSphere MQ

IBM WebSphere MQ is the messaging software that enables communication between a WebSphere message broker and the adapters.

Refer to the following WebSphere MQ publications for installation and configuration information:

- *WebSphere MQ: Quick Beginnings*
- *WebSphere MQ: System Administration*
- *WebSphere MQ: Intercommunication*

Note: You can browse or download these documents from IBM's Web site at: http://www.ibm.com/software/mqseries.

To configure WebSphere MQ to work with a WebSphere message broker, see "Configuring the message broker to work with the connector" on page 77..

Installing WebSphere Business Integration adapters

To install WebSphere Business Integration Adapters, first obtain a supported version of the product. Then download the documentation for that release from the WebSphere Business Integration Adapters InfoCenter at the following Web address: http://www.ibm.com/software/websphere/wbiadapters/infocenter. Follow the instructions in the *Installation Guide for WebSphere Business Integration Adapters* to install the product in your business integration system.

Chapter 7. Administering the business integration system

This chapter provides information about the administrative tasks you need to perform for WebSphere Business Integration Adapters. Administrative tasks that relate exclusively to the WebSphere message broker you are using are covered in the administrative documentation for that message broker (Refer to "Related documents" on page vii for more information.)

The following sections are included in this chapter:

Starting a connector

The method for starting a connector depends on whether it is running on a UNIX or a Windows system. Settings in the connector configuration file are the default source for connector property values. However, some connector properties specified in the connector's configuration file can be overridden at runtime. Appendix D, "Connector startup options," on page 125 lists the options you can use to override properties set in the connector's configuration file.

Note: Be sure you have followed the instructions in "Configuring the connector startup files, shortcuts, and environment variables" on page 95 before you start a connector.

From Windows

You can start the connector in several ways:

- Double-click the desktop shortcut, if you have one created.
- Select the connector to start from **Programs > IBM WebSphere Business Integration Adapters > Adapters > Connectors.** The program name is "IBM WebSphere Business Integration Adapters" by default, but it can be customized.
- You can start the connector from the Adapter Monitor perspective of System Manager. Refer to "Change the state of an adapter" on page 64 for details.
- In a DOS window:
  - For most connectors, type the command:

    ```
    start_connName connectorName brokerName -cConfigFile
    ```

    Where:

    connName and *connectorName* are the name of the connector, *brokerName* is the name of the WebSphere message broker, and *configFile* specifies the full-path name of the connector's configuration file.

    You can modify the startup options for a connector by editing the connector's shortcut properties or by editing directly the connector's `start_connName.bat` file.

Note: Most connectors, including all Java-based connectors and most connectors provided by IBM can use the syntax described above. You can determine if your connector uses this syntax by looking in the directory *ProductDir*/`connectors`. If a script named `start_connName` is present, use the syntax above . Some connectors written in C++ must use the syntax described in the following section instead. If a `start_connName` script associated with your connector is not present in this directory, use the syntax described below.

– For a C++ connector that cannot use the `start_connName` syntax described above (some connectors from IBM, and some custom connectors) navigate to the directory where the connector is installed. By default, this directory is *ProductDir*. Type the command:

```
start_connector connectorName brokerName -cConfigFile
```

Where:

connectorName is the name of the connector, *brokerName* is the name of the WebSphere message broker, and *configFile* specifies the full-path name of the connector's configuration file.

You can modify the startup options for a Java connector by editing the connector's shortcut properties or by editing the connector's `start_connector.bat` file directly.

From UNIX

To start a connector:

1. Navigate to the *ProductDir*/`bin` directory
2. Run the connector manager script by issuing the command:
   `connector_manager_`*connName* `-start`

   Where:

   connName is the name of the connector.

   The case and spelling of this connector name must match the name of the connector's subdirectory under *ProductDir*/`bin/connectors`.

Stopping a connector

A connector can be stopped from either the system on which the connector is installed or the system on which the integration broker is installed. Several methods are available to stop a connector, as follows:

From the connector system

From Windows

- On Windows, you can stop the connector from the Adapter Monitor perspective of System Manager. Refer to "Change the state of an adapter" on page 64 for details.
- If you cannot stop the connector from Adapter Monitor, in the console window for that connector type "Q", and press Enter. This terminates the connector's process.

From UNIX

From a UNIX system, use the following command.

1. Navigate to the *ProductDir*/`bin` directory,
2. Type: `connector_manager_`*connName* `-` *option*

where `option` is one of the following options:

Table 9. connector_manager command stopping options

Option	Description
`- stopgraceful`	Invokes a java utility from the connector-agent side to stop the connector agent. However, if the automatic restart feature is on, the connector agent will restart itself again.
`-stop`	Invokes a java utility that stops the connector agent, so even if the automatic restart feature is on, the connector agent will be forced to shut down.
`-kill`	Kills the operating-system process for the connector agent. Use this if other methods fail to stop the connector.

From the broker system

From either Windows or UNIX systems, you can issue the `mqsiremotestopadapter` command at the command line on the integration broker from the system on which the integration broker is installed if the WebSphere Business Integration Adapters Utility has been installed.

Note: The WebSphere MQ Integrator Broker SupportPac, IO02: WebSphere Business Integration Adapters Utility must have been installed to make this command available.

The syntax for the command is:

```
mqsiremotestopadapter brokerName -c adapterName -n adapterQueue
[-h hostName] [-p port] [-q adapterQueueManager]
```

For details refer to the following section, "Mqsiremotestopadapter command line parameters."

Mqsiremotestopadapter command line parameters

Required Parameters

brokerName
The name of the WebSphere message broker with which the adapter is communicating.

-c *adapterName*
Identifies the name of the adapter to shutdown.

-n *adapterQueue*
The name of the WebSphere MQ queue monitored by the adapter for a shutdown message.

Optional Parameters

-h *hostName*
The name or IP address of the machine where the WebSphere MQ Queue Manager, specified with -q parameter, is running (defaults to the local machine).

-p *port* The port on which the WebSphere MQ Queue Manager, specified with the -q parameter, is listening for incoming requests. The default value is 1414.

-q *adapterQueueManager*
Specifies the WebSphere MQ Queue Manager that is being used by the adapter. The queue identified by the -n parameter must be managed by this Queue Manager. This Queue Manager is located on the same machine as the adapter.

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 60.
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*.

Using Adapter Monitor and Fault Queue Manager

Adapter Monitor is a perspective of the System Manager that enables you to change the state of an adapter, and, through Fault Queue Manager, to handle events that have failed and been received by the fault queue. For more information on using System Manager, refer to the Appendix E, "System Manager and the Eclipse Workbench," on page 129.

Adapter Monitor perspective

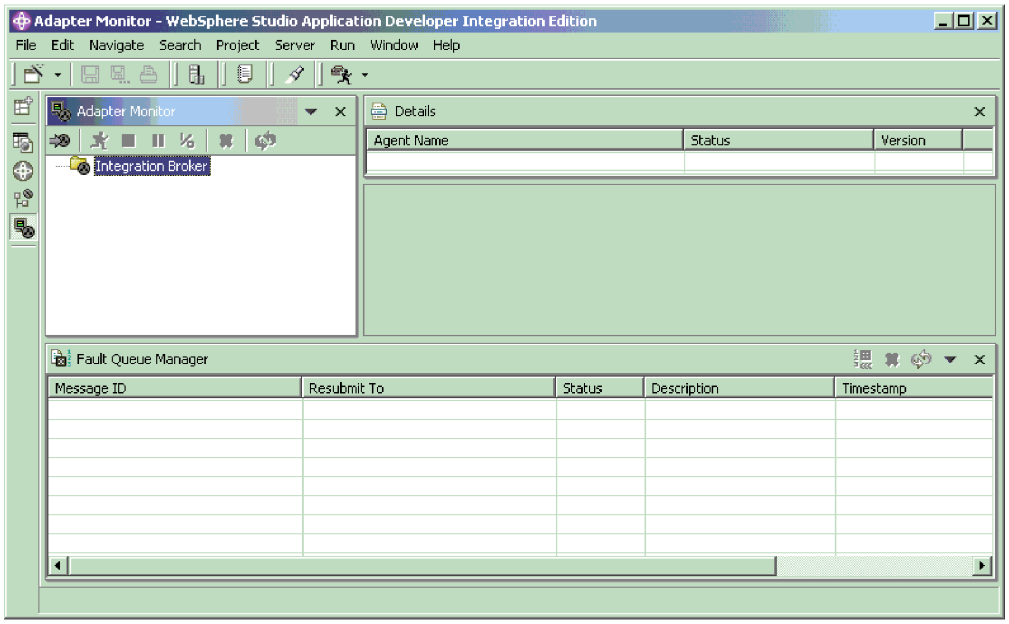
The Adapter Monitor perspective enables you to administer adapters and, through the Fault Queue Manager panel, to resubmit messages when errors occur in the processing of submitted events.

Adapter Monitor is used only with adapters that have been configured for use with JMS.

Opening Adapter Monitor

You can open Adapter Monitor as follows: In the System Manager perspective, expand a project folder, right-click on a connector definition icon, and choose

Adapter Monitor from the pop-up dialog. The Adapter Monitor window displays:


Adapter Monitor - WebSphere Studio Application Developer Integration Edition
File Edit Navigate Search Project Server Run Window Help
Adapter Monitor
Integration Broker
Details
Agent Name
Status
Version
Fault Queue Manager
Message ID
Resubmit To
Status
Description
Timestamp

Setting Adapter Monitor preferences

You can set preferences that determine the intervals at which the Adapter Monitor will poll the state of the adapter and the number of messages that will be displayed in the Fault Queue Manager. To set the preferences, from the WSADIE menu bar, choose **Window>Preferences>Adapter Monitor Preferences**, and set values for the following:

- Adapter Monitor View

  Enter a numeric value for the number of seconds that will elapse between each poll of the adapter status.
- Fault Queue Manager

  Enter a numeric value for the maximum number of messages that will be displayed by Fault Queue Manager.

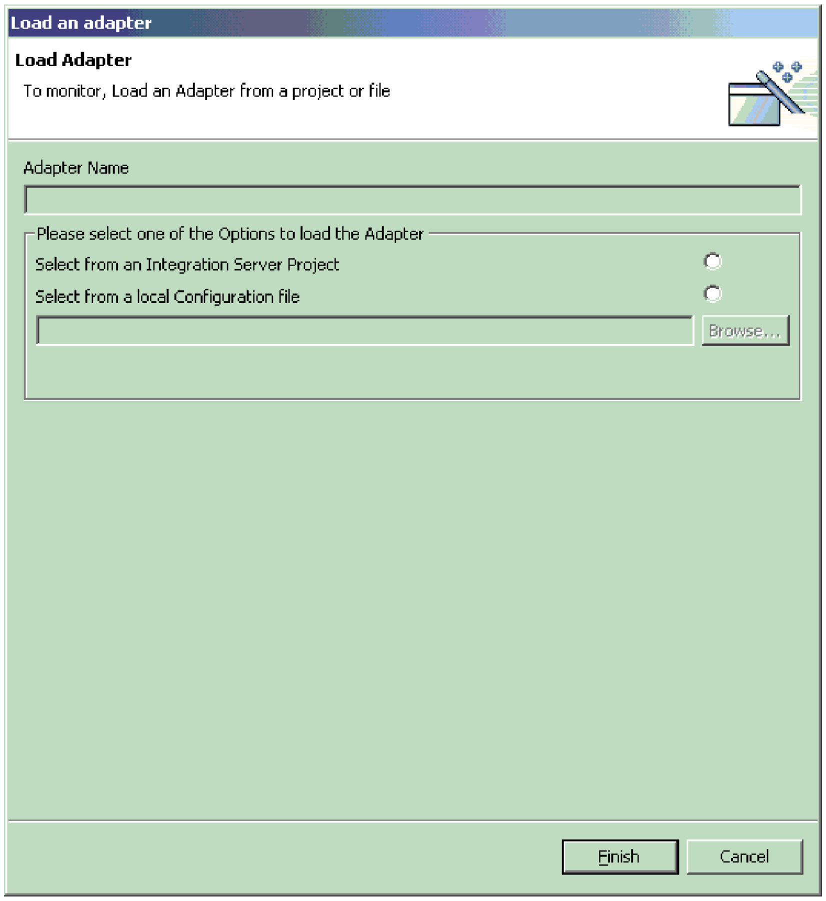
Choose Apply or OK.

Loading an adapter

To use Adapter Monitor, first start the queue manager. Then start the adapter and load the adapter into the Adapter Monitor view (you can also start the adapter after loading it into the Adapter Monitor view.

To start the adapter, use the connector startup script, shortcut, Visual Test Connector or other mechanism that you have set up for starting the specific adapter.

To load the adapter in Adapter Monitor, right click Integration Broker and choose Load Adapter. The following dialog appears:


Load an adapter

Load Adapter

To monitor, Load an Adapter from a project or file

Adapter Name

Please select one of the Options to load the Adapter

Select from an Integration Server Project

Select from a local Configuration file

You can choose to load an adapter either from the adapter's configuration file, or from an integration project.

- If you choose to load from project, the dialog displays the User Projects currently available in System Manager. Select the connector within a project and click Finish. (If no User Projects are available, the display will be empty, and you will need to either create a User Project, or load from the configuration file.)
- If you choose to load from file, a browse box opens; navigate to the configuration file for the adapter (typically, the configuration file has an file extension of *.cfg, but other extensions are possible) and choose Save. The name of your selected configuration file appears in the box. Choose Finish. The adapter is loaded into Adapter Monitor, and the current data for the adapter is displayed.

Adapter Monitor displays

When you load an adapter, an icon for the adapter appears under the Integration Broker folder in the top left panel. The icon indicates the current state of the adapter. The act of loading an adapter into Adapter Monitor does not by itself change the state of the adapter. After you have loaded the adapter, you can then perform actions from Adapter Monitor to change the state of the adapter.

The Adapter Monitor displays are refreshed periodically, according to the polling time interval that you set for the Adapter Monitor View in its preferences.

However, you can perform an immediate refresh at any time by choosing the Refresh button from the toolbar or menu bar.

The top right panel of Adapter Monitor is the Details panel. The Details panel shows the name of the adapter, its status, and version.

The bottom panel of Adapter Monitor comprises the Fault Queue Manager view. This view shows the messages that have been routed to the queue manager's fault queue. You can use the Fault Queue Manager display to either resubmit or delete messages that were placed in the fault queue as the result of a failed event flow.

Change the state of an adapter

The Adapter Monitor enables you to monitor and change the state of an adapter. The state of an adapter refers to the processing that is (or is not) being performed by an adapter.

Note: The existence of an adapter ″state″ presumes that the adapter has been started. An adapter that has not yet been started, or that was shutdown and not restarted, has no state, and is not affected by any actions in the Adapter Monitor. For information about starting an adapter, see "Starting a connector" on page 57

. Generally, the adapter performs two types of processing:

- Polling for event notification

  The adapter polls the event store of its application for events and sends the events as business object messages to the integration broker.
- Request processing

  The adapter receives request business objects sent from the integration broker to the application.

Adapter State	**Request Processing**	**Polling**
Active	yes	yes
Paused	yes	no
Inactive	no	no

To change the state of an adapter, right-click on the icon for the adapter, and choose one of the following:

- Activate

  Changes the adapter from the Paused or Inactive state to Active
- Deactivate

  Changes the adapter from the Paused or Active state to Inactive
- Pause

  Changes the adapter from Active state to Paused
- Shutdown

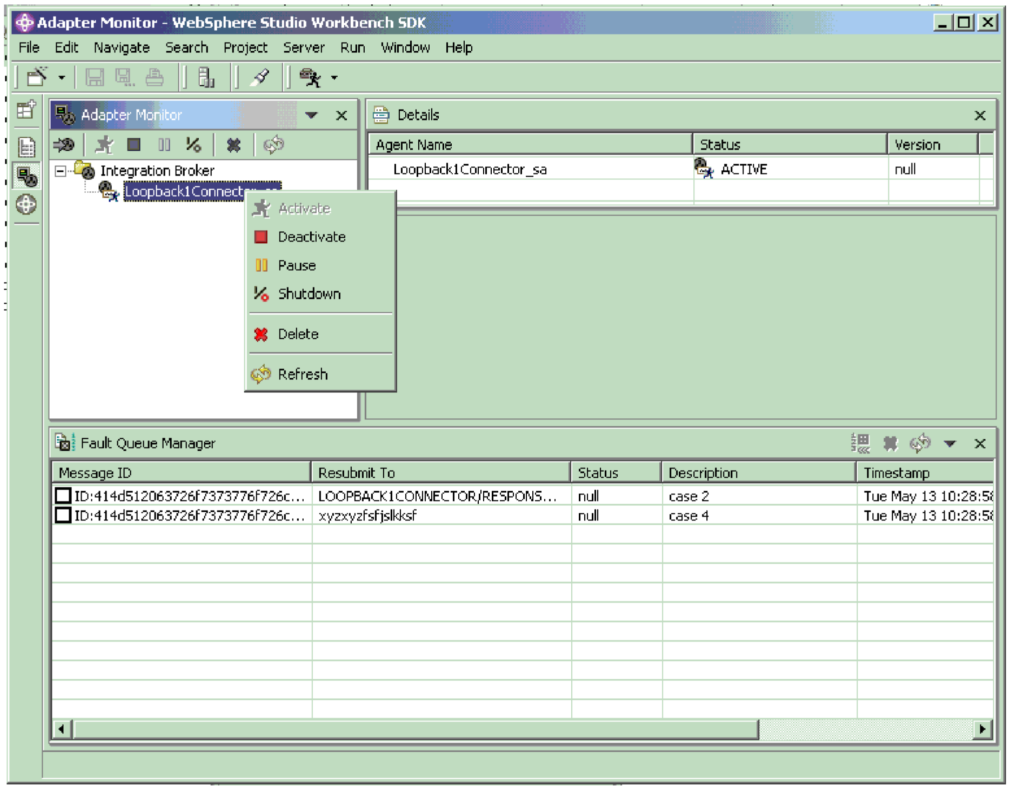
  Stops the adapter. This action terminates the connector startup script, closing the connection with the application and freeing any allocated resources. The connector remains shut down until restarted. The connector cannot be restarted from Adapter Monitor; see ″Starting an Adapter″ for instructions on starting and restarting.

The following two commands do not directly affect the processing of the connector, but do affect what the Adapter Monitor displays about the connector:

- Delete

  Deletes this adapter configuration from the Adapter Monitor. This does not change the adapter or alter its state; it only makes it unavailable from the Adapter Monitor.
- Refresh

  Triggers a GETSTATUS command which gets the current status of the connector agent



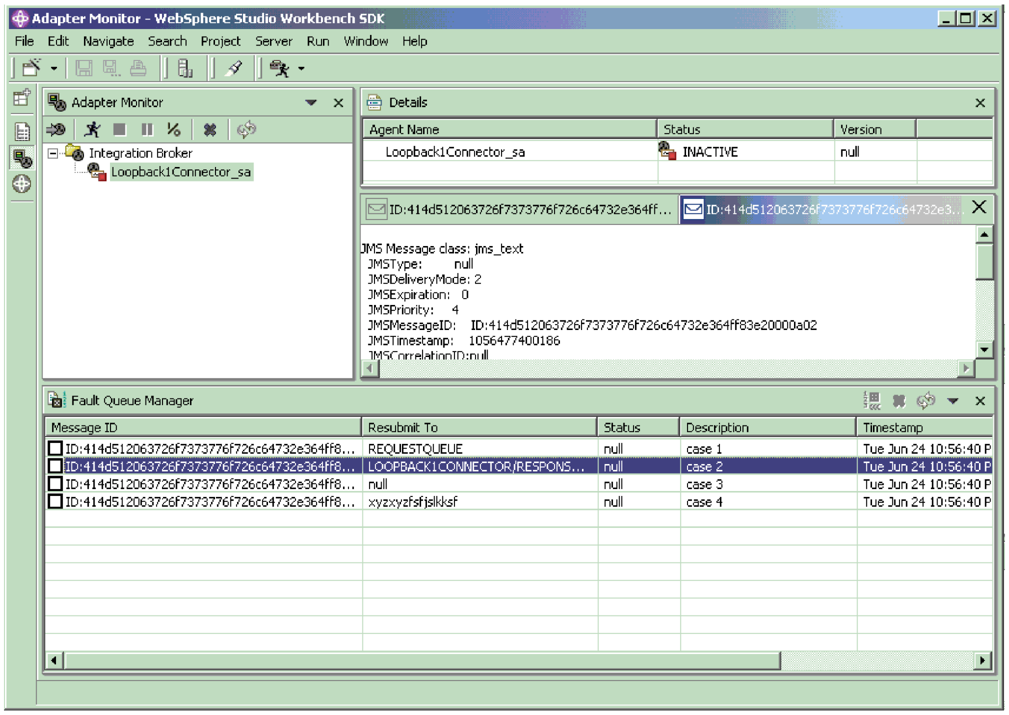
Agent Name	Status	Version
Loopback1Connector_sa	ACTIVE	null

Message ID	Resubmit To	Status	Description	Timestamp
ID:414d512063726f7373776f726c…	LOOPBACK1CONNECTOR/RESPONS…	null	case 2	Tue May 13 10:28:5
ID:414d512063726f7373776f726c…	xyzxyzfsfjslkksf	null	case 4	Tue May 13 10:28:5

Using the Fault Queue Manager Display

The Fault Queue Manager display shows events that have failed and been received by the fault queue. The display shows as many failed messages as you set in the preferences, listed in the order in which they were received.

To display information from the queue manager about any listed fault queue message, double-click the message field. Information for that message is displayed in the editor view panel (read only):


Adapter Monitor - WebSphere Studio Workbench SDK
File Edit Navigate Search Project Server Run Window Help
Adapter Monitor
Integration Broker
Loopback1Connector_sa
Details
ID:414d512063726f7373776f726c64732e364ff...
ID:414d512063726f7373776f726c64732e3...
Fault Queue Manager

Agent Name	Status	Version
Loopback1Connector_sa	INACTIVE	null

```
JMS Message class: jms_text
 JMSType:       null
 JMSDeliveryMode: 2
 JMSExpiration:   0
 JMSPriority:     4
 JMSMessageID:    ID:414d512063726f7373776f726c64732e364ff83e20000a02
 JMSTimestamp:    1056477400186
 JMSCorrelationID:null
```

Message ID	Resubmit To	Status	Description	Timestamp
☐ ID:414d512063726f7373776f726c64732e364ff8...	REQUESTQUEUE	null	case 1	Tue Jun 24 10:56:40 P
☐ ID:414d512063726f7373776f726c64732e364ff8...	LOOPBACK1CONNECTOR/RESPONS...	null	case 2	Tue Jun 24 10:56:40 P
☐ ID:414d512063726f7373776f726c64732e364ff8...	null	null	case 3	Tue Jun 24 10:56:40 P
☐ ID:414d512063726f7373776f726c64732e364ff8...	xyzxyzfsfjslkksf	null	case 4	Tue Jun 24 10:56:40 P

Handling failed events

The Fault Queue Manager lists and enables you to handle the failed event messages from two types of interaction patterns: the HubRequest (request/response) and the Agent Delivery for container managed events.

You can either delete event messages from the queue, or attempt to resubmit them.

To resubmit an event, mark the check box in the Message ID field for the event, right click, and choose Resubmit.

Adapter Monitor attempts to resubmit the event. If the event is successfully resubmitted, it is removed from the Fault Queue Manager display.

Messages with null value for the ResubmitTo field can not be resubmitted. An attempt to resubmit an event can fail, either because the message itself is invalid, or the ResubmitTo queue is not valid or available.

A message can be invalid because it contains an invalid JMS type or because it cannot be converted to a business object. In either case, when the attempt to resubmit an invalid message fails, Fault Queue Manager displays an error for the invalid message. When you choose OK, the message is not resubmitted, and it is removed from the fault queue display. If you close the dialog without choosing OK, the message will remain in the fault queue display.

If a message is valid but the resubmit attempt fails because the message ResubmitTo queue is null, invalid or unavailable , a Resubmit dialog appears, showing the values of the message. You can choose to either retain (by choosing Cancel) or delete (by choosing OK) the message from the queue.

Clearing messages from WebSphere MQ queues

WebSphere Business Integration Adapters provides a sample batch file you can use to clear messages from the WebSphere MQ queues in the business integration system. Clearing the queues might be necessary if a problem with the business integration system prevents messages from being removed for processing.

To clear messages from WebSphere MQ queues, run the batch file, `clear_mq.bat` (Windows) or `clear_mq` (UNIX) located in the *`ProductDir`*`\templates` directory. This batch file clears messages from the queues specified in the file, `crossworlds_mq.tst`. For more information about editing `crossworlds_mq.tst`, see"Using WebSphere Business Integration Adapters batch files to configure WebSphere MQ queues" on page 78.

Managing log and trace files

The following tools provide graphical user interfaces for configuring and viewing message logging and tracing. Use the:

- Connector Configurator to set up or change connector logging, and tracing
- Log Viewer to display log and trace files.

In addition to using Log Viewer to view logs, you can open the log with a text editor.

During startup, the connector generates a temporary log file. This file contains all messages that are logged during startup, including connector properties and business object definitions that are passed to the connector framework. The file name is `broker_name_connector_name_tmp.log`, and it is written to the *`ProductDir`* directory. Once the connector is running, it handles logging and tracing as configured in the standard connector configuration properties.

While logging and tracing messages are written using UTF-8 encoding in the locale specified in the connector's configuration file, time information is written in a locale-independent format.

At connector startup, a log file is also created, if one does note already exist, in the location specified in the connector's configuration file. If a log file already exists, new entries are simply added to it. Unless a limit has been placed on the size of the connector log file, its size depends on the amount of time since it was last managed and the volume of transactions passing through the system. If the connector log file is configured with no size limit, it can continue to grow until it cannot be opened or it exhausts its disk space.

If tracing is enabled, a trace file is created at startup if one does not already exist. The size of the trace file must be managed in the same manner as the connector log file to avoid the problems caused by excessively large files.

Table 13 on page 93 lists the files used by the connector to store logging and tracing information.

To manage log and trace files, you can use the Connector Configurator to:

- Specify a size limit for log and trace files
- Have the files automatically archived once they reach their size limit.
- Specify the number of archive files to maintain.

For more information about using the Connector Configurator to set these options, see “Configuring logging and tracing options” on page 93.

Archival logging of log and trace files

If archival logging is enabled, each time the connector’s log or trace file reaches its maximum size, it is renamed as a new archive file. The archive file’s name is derived from the original log or trace file name, with the following inserted into the name: `_Arc_`*`number`*.

For example, if five archive files are to be used and the log file has the name `Connector.log`, then:

- The first archive created is named `Connector_Arc_01.log`.
- When the new log file fills up, `Connector_Arc_01.log` is renamed *`Connector`*`_Arc_02.log`,
- New log information is again saved to `Connector_Arc_01.log` and so on in a circular fashion, until there are five archive files.
- If there are already five archive files when a new log file is created, the oldest one, number five, is deleted. Then the remaining archive files are renamed and their numbers incremented so the number of archives matches the number you configured. Figure 21 shows the progression of files using this configuration.

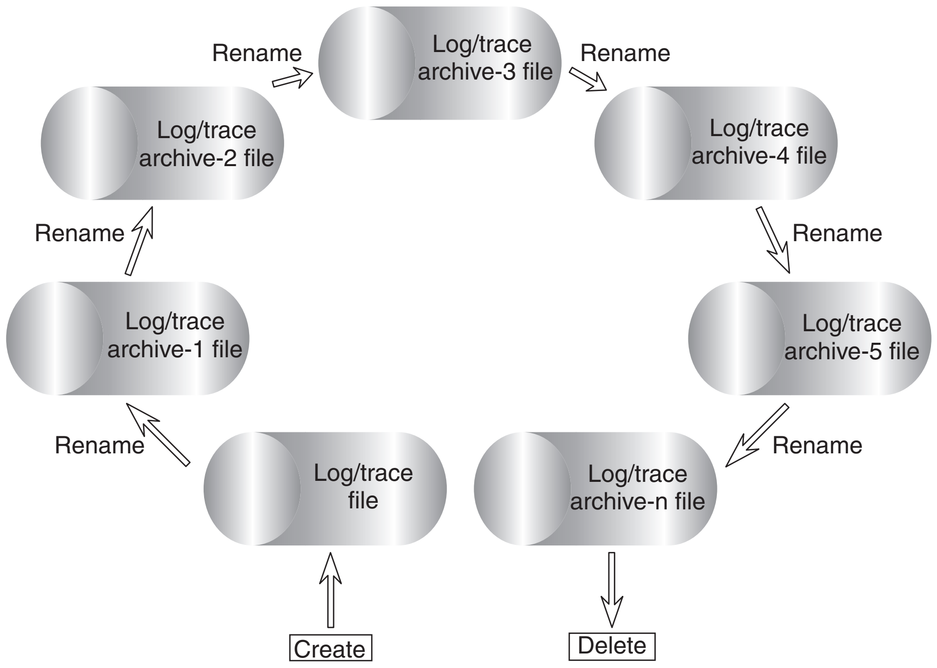


Figure 21. Circular archival logging.

See the configuration task “Configuring logging and tracing options” on page 93 for details.

Managing other files

While log and trace files can be managed by means of archival logging, other log files specific to each application need to be managed manually. Most of these files are created during runtime if they do not already exist. New information is appended to any existing file.

Any file management procedure can be used, but IBM suggests the following periodic log file management:

- Rename the files by appending a date to the file.
- Move the files to an archive directory.

Using Log Viewer to view connector messages

Log Viewer allows you to see messages contained in the log file and the trace file for the connector. You can sort and filter the output display as well as print, save, and email copies of the file. Logging and tracing options, as well as the location of the generated files, are specified as a properties in the connector's configuration file.

Note: Log Viewer runs only on a Windows 2000 machine. To configure or view a UNIX log file using Log Viewer, copy the log file from the UNIX machine to a Windows machine and view it from there.

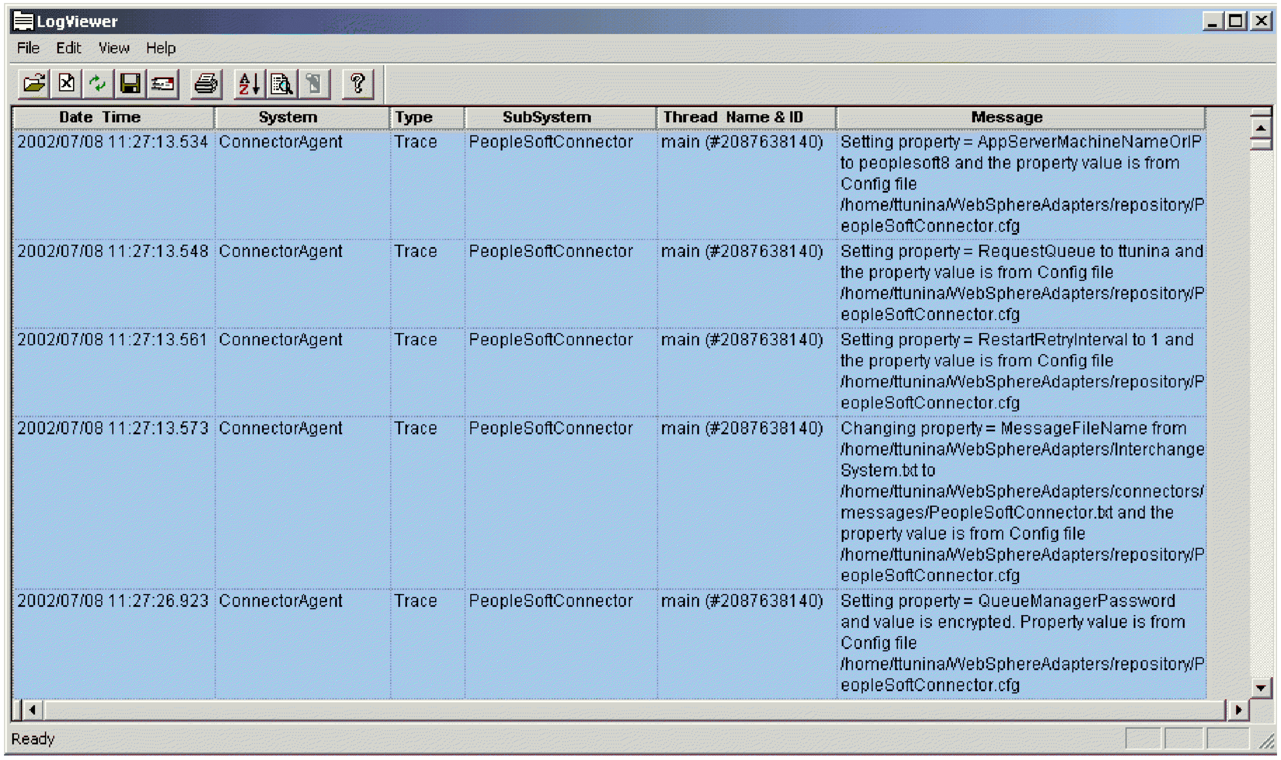
To start Log Viewer, you can do either of the following:

- From the IBM WebSphere Business Integration Adapters Start menu shortcut, select Log Viewer from the Tools submenu. Use the Open option of the File menu to browse for the log file.
- Use the Run command from the Start menu and browse for the `LogViewer.exe` file. Use the Open option of the File menu to browse for the log file.

Using the Log Viewer menu options, you can perform the following tasks:

- "Setting Log Viewer preferences" on page 70
- "Changing how messages are viewed" on page 72
- "Controlling the Log Viewer display output" on page 74

Log Viewer, displaying a sample log file is shown in Figure 22 on page 70.



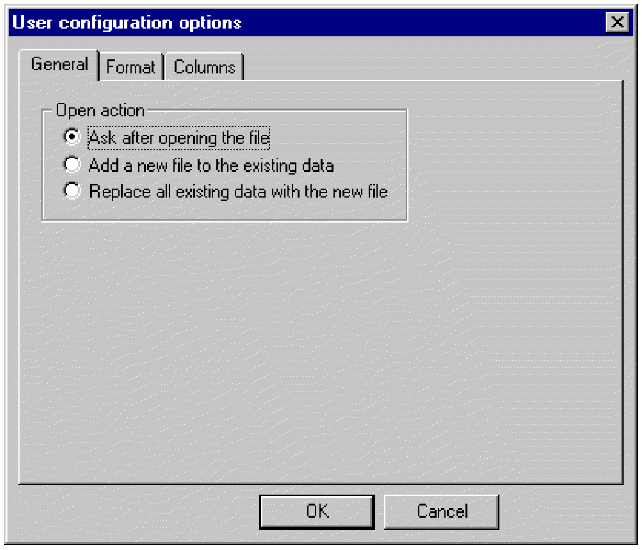
Date Time	System	Type	SubSystem	Thread Name & ID	Message
2002/07/08 11:27:13.534	ConnectorAgent	Trace	PeopleSoftConnector	main (#2087638140)	Setting property = AppServerMachineNameOrIP to peoplesoft8 and the property value is from Config file /home/ttunina/WebSphereAdapters/repository/PeopleSoftConnector.cfg
2002/07/08 11:27:13.548	ConnectorAgent	Trace	PeopleSoftConnector	main (#2087638140)	Setting property = RequestQueue to ttunina and the property value is from Config file /home/ttunina/WebSphereAdapters/repository/PeopleSoftConnector.cfg
2002/07/08 11:27:13.561	ConnectorAgent	Trace	PeopleSoftConnector	main (#2087638140)	Setting property = RestartRetryInterval to 1 and the property value is from Config file /home/ttunina/WebSphereAdapters/repository/PeopleSoftConnector.cfg
2002/07/08 11:27:13.573	ConnectorAgent	Trace	PeopleSoftConnector	main (#2087638140)	Changing property = MessageFileName from /home/ttunina/WebSphereAdapters/InterchangeSystem.txt to /home/ttunina/WebSphereAdapters/connectors/messages/PeopleSoftConnector.txt and the property value is from Config file /home/ttunina/WebSphereAdapters/repository/PeopleSoftConnector.cfg
2002/07/08 11:27:26.923	ConnectorAgent	Trace	PeopleSoftConnector	main (#2087638140)	Setting property = QueueManagerPassword and value is encrypted. Property value is from Config file /home/ttunina/WebSphereAdapters/repository/PeopleSoftConnector.cfg

Figure 22. Log Viewer with sample log.

Setting Log Viewer preferences

1. To set Log Viewer preferences, select **Edit >Preferences** from the menu bar.

   The User Configuration Options, General properties dialog box displays (see Figure 23).



Open action
- [✓ Ask after opening the file]
- Add a new file to the existing data
- Replace all existing data with the new file

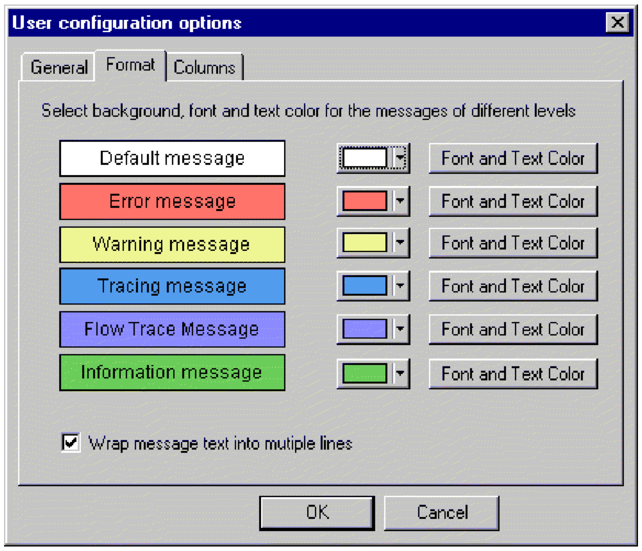
Figure 23. Log Viewer User Configuration Options, General Properties screen.

This dialog lets you specify how to display the log file when you open it. The available choices are:

- Query your preferences each time you open a log file.
- Merge the log file you are opening with the log file that is currently displayed.
- Replace the log file that is currently displayed with the contents of the one you are opening.

2. To change the background color and font of the Log Viewer messages, click the **Format** tab.

   The User Configuration Options, Format properties dialog box displays (see Figure 24).



User configuration options

General | Format | Columns

Select background, font and text color for the messages of different levels

Default message	⬜	Font and Text Color
Error message	🟥	Font and Text Color
Warning message	🟨	Font and Text Color
Tracing message	🟦	Font and Text Color
Flow Trace Message	🟪	Font and Text Color
Information message	🟩	Font and Text Color

☑ Wrap message text into mutiple lines

OK | Cancel

Figure 24. Log Viewer User Configuration Options, Format Properties screen.

This dialog lets you specify how to display the log messages. The available choices are:

- Assign different background colors and fonts for each of the types of messages that display so you can easily recognize their severity (for example, red background with larger font allows for Warning messages).
- Wrap the text of messages if the text is wider than the column.

Note: Flow trace messages are not generated for connectors using a WebSphere message broker.

3. To change the Log Viewer columns that are displayed, click the **Columns** tab.

   The User Configuration Options, Columns properties dialog box displays (see Figure 25 on page 72).

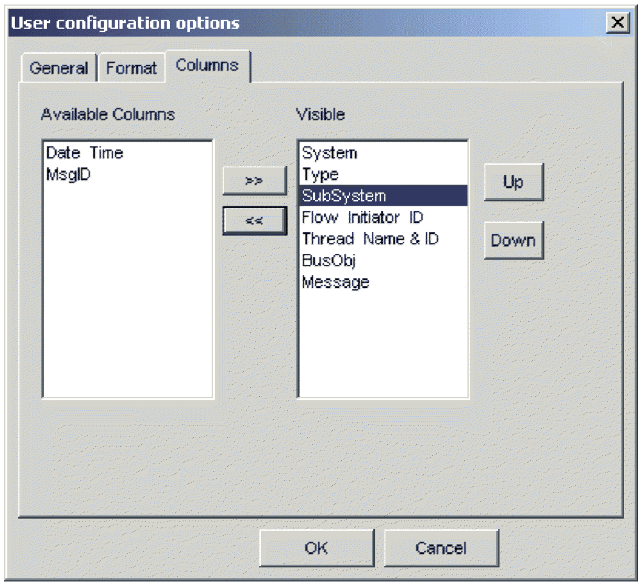


Figure 25. Log Viewer User Configuration Options, Columns Properties screen.

This dialog lets you specify which columns display in Log Viewer as follows:

- To display a column, highlight a column name in the Available Columns pane and click the >> button to move it to the Visible pane.
- To hide a column, highlight a column name in the Visible pane and click the << button to move it to the Available Columns pane.
- Click any of the column names in the Available Columns pane and click the **Up** or **Down** button to change its ordering from left to right in the Log Viewer display. Up moves columns to the left and Down moves columns to the right.
- Click the checkbox next to **Automatically hide empty columns** to keep the Log Viewer display compact.

Note: The column, Flow Initiator ID, is not relevant for connectors using a WebSphere message broker.

Changing how messages are viewed

The View menu contains additional options to change Log Viewer displays. From that menu, you can:

- Display/hide the Log Viewer toolbar.
- Display/hide the Log Viewer status bar.
- Split the window into two or more views
- Filter or show all messages by checking filtering options in the filter tabs, such as time range or by type of message (see Figure 26 on page 73 and Table 10 on page 74). To set filter options:
  1. From the menu bar, select **View > Filter > Use Filter** . The Filter Settings dialog box displays.
  2. In the Activate Filters area, click the box that is associated with the tab containing the filter options you want to apply.
  3. Click **OK** to enable filtering.

The filtered output can be toggled on or off with the Filter Toggle button on the toolbar.

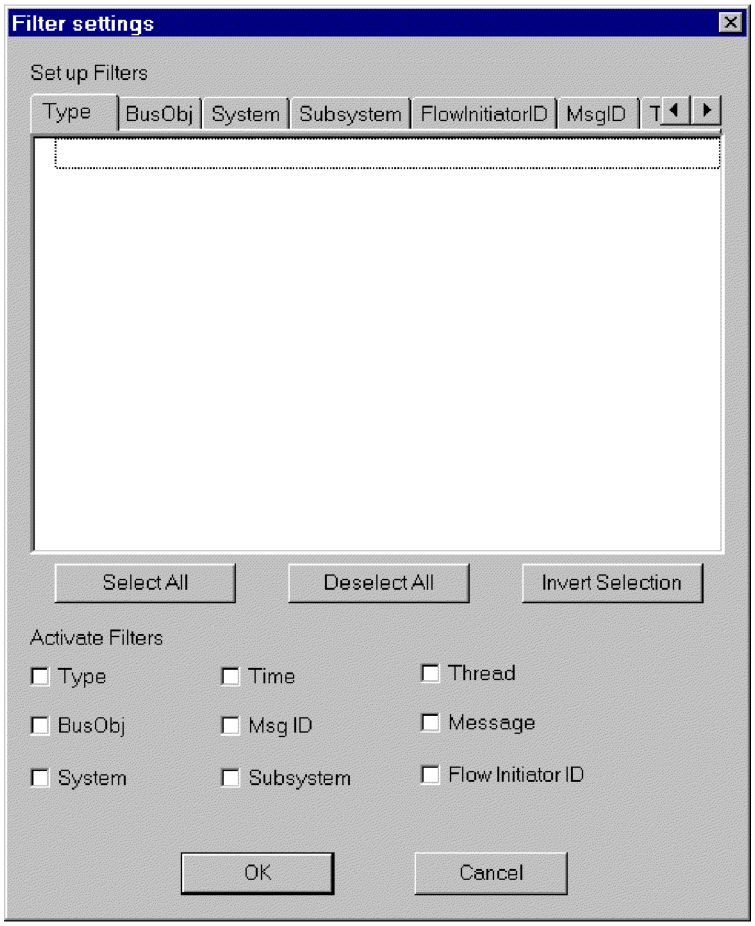
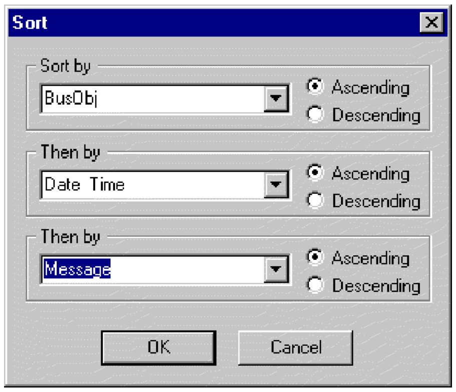


Figure 26. Log Viewer Filter screen.

Note: The tab, FlowInitiator ID, is not relevant for connectors using a WebSphere message broker.

- Sort the messages; Figure 27 shows the Sort options. Click the down arrow in each sort field to select Date/Time or EventID. You can also sort in ascending or descending order.



Sort

Sort by: BusObj [✓ Ascending] Descending

Then by: Date Time [✓ Ascending] Descending

Then by: Message [✓ Ascending] Descending

Figure 27. Log Viewer Sort Properties screen.

Controlling the Log Viewer display output

Several options are available for controlling Log Viewer output. In the File menu, there are options for print previewing, printing, saving, refreshing the display, sending to an email recipient, and determining the style for page setup, headers and footers. The variables for header and footers are:

Variable name	Description
$F	Name of file
$A	Application name
$P	Page number
$N	Total number of pages
$D	Date (can be followed by additional parameters (for example $D{%y:%h:%m})

Filtering messages

To filter the messages that will be displayed in Log Viewer, choose **View->Filter->Use Filter**from the Log Viewer menu bar. The Filter Settings dialog displays categories that correspond to the parameters of the logging message format. Message format parameters are listed in Table 10.

Table 10. Message format parameters for log file.

Variable	Description
Time	Timestamp: the date of logging in the format *year/month/date time.*
System	For connectors using the WMQI integration broker, *system* is the application-specific component of the connector.
Thread	Thread name and thread ID.
Name	The name of the component, such as ClarifyConnector.
MsgType	Indicates the severity of the message. See Table 11 on page 75.
MsgID	The message number.
SubSystem	The connector name.
BO	The business object name.
MsgText	The associated text for the message number.
BOD	Business object dump. The data contained in the business object.

In the Filter Settings dialog, you first choose the filtering categories that you want to use, then select the specific items that you want to display from each category, and then choose which filters you want to activate for your current Log Viewer display.

Follow these steps:

1. In the Filter Settings dialog, choose a tab under Set up Filters to display the items that you want to use for filtering messages. For example, choose Time if you want to filter according to the timestamp of the message. You can set up multiple filters, and use them either separately or together.

2. In the displayed list of items, select each item for which you want to view messages in Log Viewer. For example, if you want to view only messages that are timestamped between 5 March 2002 at 9:00 AM and 6 March 2002 at 5:00 PM, select the range for those times under the Time tab.

   You can use the buttons below the list box to select all the displayed items, or to deselect all the displayed items, or to invert your current selection choices.
3. Under Activate Filters, check the box for each filter type that you want to activate. For example, if you want to see only those messages with a particular message ID that have a particular timestamp, activate both the MsgID filter and the Time filter.
4. Click **OK**. The Filter Settings dialog closes, and the Log Viewer display refreshes to show only those messages that you have allowed through the filters.

Note that in addition to filtering according to the categories, you can also display only those messages that contain a specific text string. To do so, select Messages under Set up Filters, enter the specific text for which you want to show messages, and check the box for **Message under Activate Filters**.

Message types

Table 11 describes the types of messages issued by WebSphere Business Integration Adapters.

Table 11. Message types.

Message type	Description
Info	Informational only. You do not need to take action.
Warning	A default condition chosen by InterChange Server.
Error	A serious problem that you should investigate.
Fatal Error	An error that stops operation and should be reported.
Trace	Tracing information for the trace level specified.
Flow Trace	Flow tracing information for business objects.
Internal Error	A serious internal problem that should be investigated.
Internal Fatal Error	An internal error that stops operation. It should be reported.

Note: If a message type of Internal Error or Internal Fatal Error appears, record the circumstances surrounding the problem, and then contact IBM Technical Support.

Chapter 8. Configuring the WebSphere business integration system

This chapter explains how to configure the components of the WebSphere business integration system: the integration broker, the business objects, and the connectors. It includes the following sections:

Overview of configuration tasks

To configure the business integration system, you need to perform the following tasks:

1. Configure the message broker to support the connector by defining the necessary queues.
2. Generate the business object definitions to be used by the connector.
3. Create a message broker project and deploy it to the message broker.
4. Configure standard and application-specific configuration properties for the connector.
5. Configure tracing, logging, and messaging options for the connector.
6. Create message flows to define how the message broker is to process the business object messages it received.

Each of these tasks is covered in more detail below.

Configuring the message broker to work with the connector

To enable a WebSphere message broker to work with a connector, you need to configure the WebSphere MQ queues that carry messages between the connector and the integration broker, and define appropriate queue configurations. You also need to ensure that the connector's configuration file contains correctly specified queue and queue manager information.

"Message queues" on page 41 provides information about how WebSphere MQ queues are used in the WebSphere business integration system. "Setting the connection mode with the queue manager" on page 92 explains how to specify the connection mode in the connector's configuration file. For detailed information about WebSphere MQ queues, queue managers, and queue configurations, see *WebSphere MQ: Intercommunication*.

Configuring the WebSphere MQ queues

The business integration system requires that you configure queues with the properties listed below.

Note: When you configure the connector, under "Specifying the queues to be used by the connector" on page 92, you will need to specify the name of each of these queues as a standard property in the connector's configuration file.

- **DeliveryQueue:** Delivers event delivery messages from the connector framework to a WebSphere message broker.
- **RequestQueue:** Delivers request messages from a WebSphere message broker to the connector framework.
- **ResponseQueue:** This queue is not used with WebSphere message brokers but must be defined for consistency with WebSphere InterChange Server.
- **FaultQueue:** Delivers fault messages from the connector framework to the message broker. The connector framework places a message on this queue when it is unable to place the message on the reply-to queue.
- **SynchronousRequestQueue:** Delivers request messages from the connector framework to the message broker that require a synchronous response. This queue is necessary only if the connector uses synchronous execution. With synchronous execution, the connector framework sends the message to SynchronousRequestQueue and waits for a response back from the message broker on the SynchronousResponseQueue. The response message sent to the connector bears a correlation ID that matches the ID of the original message.
- **SynchronousResponseQueue:** Delivers response messages from the message broker to the connector framework sent in reply to a synchronous request. This queue is necessary only if the connector uses synchronous execution.
- **AdminInQueue:** Delivers administrative messages from the message broker to the connector framework.
- **AdminOutQueue:** Delivers administrative messages from the connector framework to the message broker.

Ways to define queues

You can configure the WebSphere MQ queues needed for your adapter, using any of the following methods:

- Customize and run a batch file provided with WebSphere Business Integration Adapters.
- Use WebSphere MQ Explorer.
- Issue WebSphere MQ commands.

Tip

To make it easy to identify the connector with which a queue is associated, use the name of the connector as a prefix in the queue name. For example, name the Clarify connector's event delivery queue: `clarifyconnector/deliveryqueue`.

Using WebSphere Business Integration Adapters batch files to configure WebSphere MQ queues: WebSphere Business Integration Adapters provides a set of batch files that you can run to configure the WebSphere MQ queues needed for the adapters you are deploying. The batch files, located in `*ProductDir*\templates`, consist of:

- **configure_mq.bat** (Windows)
  configure_mq (UNIX)

  Run this batch file to configure the WebSphere MQ queues specified in `crossworlds_mq.tst`
- **crossworlds_mq.tst**
  Edit this file to specify the WebSphere MQ queues in the business integration system. This file is read as input by `configure_mq.bat` and `clear_mq.bat`, a batch file WebSphere Business Integration Adapters provided to clear messages from WebSphere MQ queues.

For more information about using `clear_mq.bat`, see "Clearing messages from WebSphere MQ queues" on page 67..

The contents of the `crossworlds_mq.tst` file are shown below. You can use this one file to specify the queues needed by each adapter you are configuring. Edit the file as follows:

1. Delete the statements:
   ```
   DEFINE QLOCAL(IC/SERVER_NAME/DestinationAdapter)
   DEFINE QLOCAL(AP/DestinationAdapter/SERVER_NAME)
   ```
   These apply only to business integration systems that use WebSphere InterChange Server.
2. For each adapter you are deploying, create a separate set of queue definition statements using as a template the statements beginning with `DEFINE QLOCAL(`*`AdapterName`*`/AdminInQueue)`.
3. If you are using bindings mode with remote queue definitions, customize the statement, `DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP),` with the requested information for each queue manager you need to configure. If you are using client mode for your queue configuration, leave the statement as is. For more information about supported queue configurations, see "Defining the queue configuration" on page 80.

```
*****************************************************************/
*                                                                */
*   Define the local queues for all Server/Adapter pairs.        */
*   For MQ queues, they must have the following definition:      */
*       Application = DEFINE QLOCAL (AP/AdapterName/ServerName)    */
*                                                                 */
*   Example:                                                      */
*   DEFINE QLOCAL(AP/ClarifyConnector/CrossWorlds)                */
*                                                                 */
*   DEFINE QLOCAL(AP/SAPConnector/CrossWorlds)                    */
*                                                                 */
*   If your server is named something different than 'CrossWorlds' */
*   make sure to change the entries to reflect that.              */
******************************************************************/
    DEFINE QLOCAL(IC/SERVER_NAME/DestinationAdapter)
    DEFINE QLOCAL(AP/DestinationAdapter/SERVER_NAME)
******************************************************************/
*   For each JMS queue (delivery Transport is JMS),
*   default values follow the convention:
*           AdapterName/QueueName
******************************************************************/
    DEFINE QLOCAL(AdapterName/AdminInQueue)
    DEFINE QLOCAL(AdapterName/AdminOutQueue)
    DEFINE QLOCAL(AdapterName/DeliveryQueue)
    DEFINE QLOCAL(AdapterName/RequestQueue)
    DEFINE QLOCAL(AdapterName/ResponseQueue)
    DEFINE QLOCAL(AdapterName/FaultQueue)
    DEFINE QLOCAL(AdapterName/SynchronousRequestQueue)
```

```
    DEFINE QLOCAL(AdapterName/SynchronousResponseQueue)
*********************************************************************/
*   Define the default CrossWorlds channel type                      */
*********************************************************************/
    DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP)
*********************************************************************/
*   End of CrossWorlds MQSeries Object Definitions                   */
*********************************************************************/
```

Using WebSphere MQ Explorer to configure WebSphere MQ queues: For information about configuring queues using WebSphere MQ Explorer, open WebSphere MQ Explorer and refer to its online help.

Using WebSphere MQ commands to configure WebSphere MQ queues: For information about configuring queues using WebSphere MQ commands, see *WebSphere MQ: System Administration Guide* and *WebSphere MQ: Script (MQSC) Command Reference*.

Defining the queue configuration

The WebSphere business integration system supports several queue managers and queue configurations. The connector can communicate with the queue manager in any of the following modes.

Bindings mode

With bindings mode, the WebSphere message broker and the connector can communicate directly with the queue manager, without using a TCP/IP connection. The integration broker and the connector need to be installed on the same machine so that they can use the same queue manager. This is the default mode.

Bindings mode with remote queue definitions

If the WebSphere message broker and the connector are installed on separate machines, with each machine running its own queue manager, the connector and the integration broker can still communicate with their respective queue managers using bindings mode. However, you need to specify remote queue definitions as explained in the example below.

Suppose brokerQM is the queue manager used by the integration broker and connQM is the queue manager used by the connector. To enable communication between the two queue managers, you need to set up the following channel definitions:

- For each queue that transmits messages from the connector to the message broker, a remote queue definition must be created on connQM that points to a local queue on brokerQM. This requirement applies to the following queues:
  - DeliveryQueue
  - SynchronousRequestQueue
  - FaultQueue
  - AdminOutQueue
- For each queue that transmits messages from the integration broker to the connector, a remote queue definition must be created on brokerQM that points to a local queue on connQM. This requirement applies to the following queues:
  - RequestQueue
  - AdminInQueue
- For reply-to queues: when the integration broker sends a request message to the connector framework, it specifies in the message header the queue manager and

queue to which the response is to be sent.This is also true for requests sent from the connector framework to the integration broker. Appendix A, “WebSphere MQ message formats,” on page 99, provides more information about these header fields. You must perform certain administrative tasks to have response messages routed to the correct reply-to queue. These are described in *WebSphere MQ: Intercommunication*.

Client mode

If the message broker and the connector must use TCP/IP to communicate with their respective queue managers, then they must use a client mode connection. Communication occurs through a client connection that uses TCP/IP as its underlying transport.

Creating business object definitions

To create the business object definitions to be used by the connector, you have several options, all of which are covered in detail in the *Business Object Development Guide*.

Note: WebSphere Business Integration Adapters uses only application-specific business objects, not generic business objects. All references to business objects throughout this book refer to application-specific business objects. Generic business objects are used in business integration systems based on the WebSphere InterChange Server integration broker. Some of the books in the WebSphere Business Integration Adapters library, such as *Business Object Development Guide* are also part of the WebSphere InterChange Server library and refer to both types of business objects.

The options for creating business object definitions are as follows:

- Use an ODA (Object Discovery Agent) to generate application-specific business object definitions. The ODA examines specified objects in the application, “discovers” the elements of those objects that correspond to business object attributes and their 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. Refer to the adapter user guide for the connector you are configuring to determine whether an ODA (Object Discovery Agent) is provided.
- If no ODA is included with the business integration adapter, you can use the Object Discovery Agent Development Kit (ODK) to develop an ODA and then run it against the application.
- Create business object definitions manually using the Business Object Designer tool.

In addition, many adapters come with sample business objects. If they are included, the samples are located in the following product directory:

```
ProductDir\connectors\ConnName\Samples
```

Once you create the business object definitions for application-specific business objects, read Appendix G, “Using Visual Test Connector,” on page 159, which explains how to test the business object definitions once you have created them.

Creating a message broker project

Once you have created the business object definitions the connector is to support, you must deploy them into the message broker workspace (for WebSphere Business Integration Message Broker) or deploy them to the message broker (for WebSphere MQ Integrator or WebSphere MQ Integrator Broker.) You do this using the System Manager. System Manager is installed with the Adapter Framework and provides a graphical user interface from which to configure and administer adapters. (For more information about System Manager, refer to the Appendix E, "System Manager and the Eclipse Workbench," on page 129.)

Note: The connector must be installed before you can deploy the project to the message broker. Refer to the *WebSphere Business Integration Adapters Installation Guide* for installation instructions.

To start System Manager, from the Windows Start menu select **IBM WebSphere Business Integration Adapters** > **Tools** > **System Manager**. Make sure you are viewing the System Manager perspective. The first step you should complete before creating a new project is to specify the importer paths.

Specifying importer and workspace paths

You must specify the paths for the broker importer and message broker workspace directory before you deploy a project to a message broker if they are not in the same workspace.

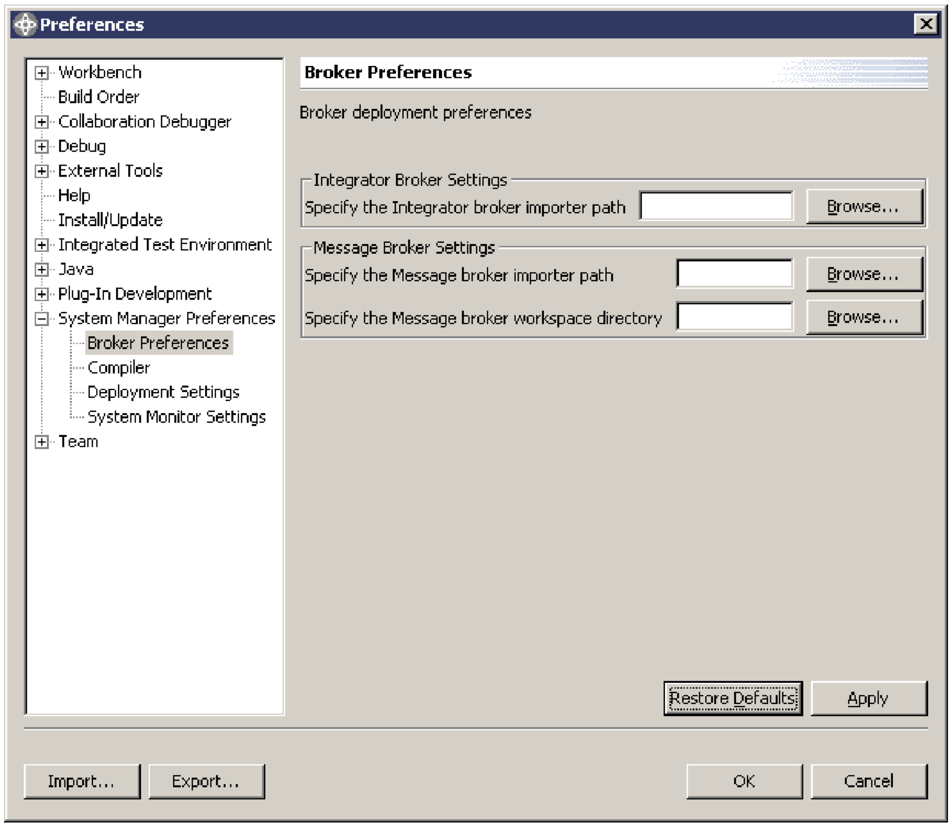
Note: If you are deploying business objects to WebSphere Business Integration Message Broker, and the System Manager and Message Broker Toolkit are in the same plugin directory you do *not* have to perform this step. During deployment, the specified project will be created in the current System Manager workspace.

Tip

After deployment, the Broker Application Development perspective in the System Manager workbench will display an entry for the destination message set project. If plugins for both System Manager and the Message Broker Toolkit are launched when you start the workbench, then they are in the same workspace. If you can see the Broker Application Development perspective in the same workbench as System Manager, then the Message Broker Toolkit plugins were also launched.

If the broker importer and message broker workspace are in different directories, specify their paths as follows:

1. From System Manager select **Window** > **Preferences** > **System Manager Preferences** >**Broker Preferences**. The following window appears:



Broker Preferences

Broker deployment preferences

Integrator Broker Settings
Specify the Integrator broker importer path Browse...

Message Broker Settings
Specify the Message broker importer path Browse...
Specify the Message broker workspace directory Browse...

2. Enter (or click Browse to locate) the pathnames as follows:

   For WebSphere MQ Integrator Broker or WebSphere MQ Integrator, enter the integrator importer path (under ″Integrator Broker Settings″). The importer is called `mqsiImpXMLSchema.exe` and should be found in the `bin` directory of your broker installation.

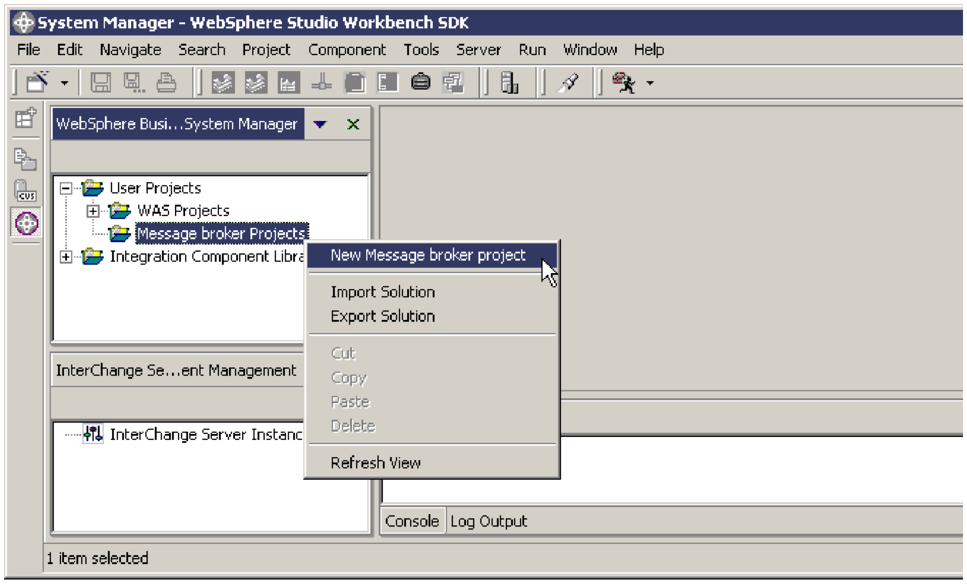
   For WebSphere Business Integration Message Broker, enter the message broker importer path and the message broker workspace directory (under ″Message Broker Settings″). The importer is called `mqsicreatemsgdefs.exe` and should be found in the `eclipse` directory of your broker installation. Make sure to select the message broker workspace, not the System Manager workspace. The default path for the message broker workspace is `eclipse\workspace`. (The message broker workspace is the workspace used to store the message set and message flow projects, and any other projects of WebSphere Business Integration Message Broker tooling.)

   Note: Pathnames entered should be absolute.

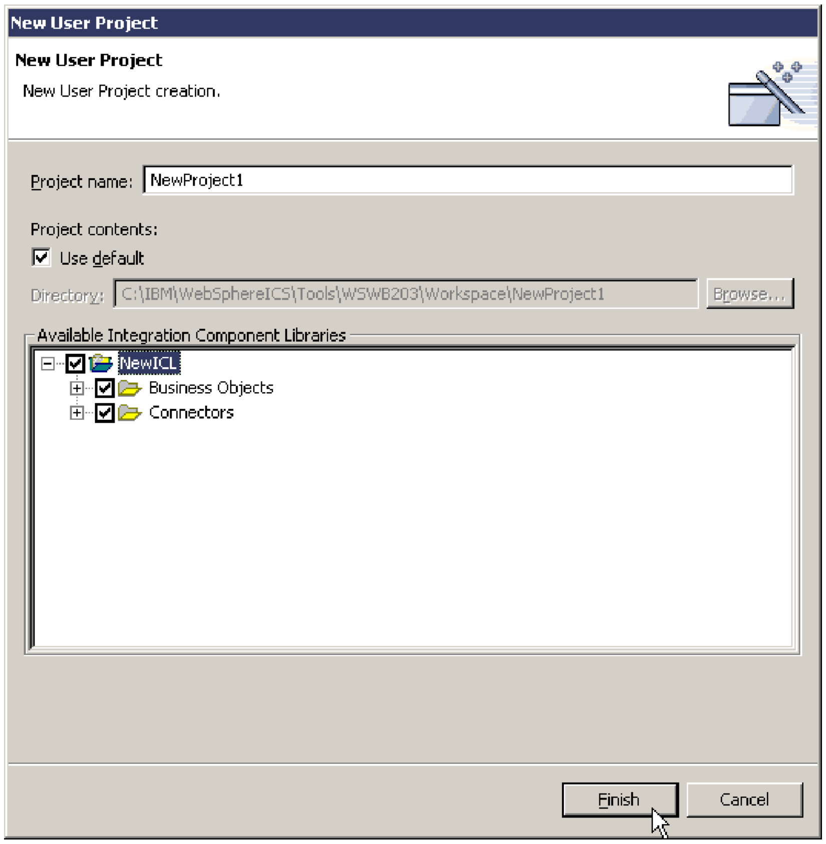
Creating a new user project

To create a new user project, follow these steps:

1. From the User Projects menu, select **Message broker Projects** and then **New Message broker project** as in the following illustration:


System Manager - WebSphere Studio Workbench SDK
File Edit Navigate Search Project Component Tools Server Run Window Help
WebSphere Busi...System Manager
User Projects
WAS Projects
Message broker Projects
Integration Component Libra
New Message broker project
Import Solution
Export Solution
Cut
Copy
Paste
Delete
Refresh View
InterChange Se...ent Management
InterChange Server Instanc
Console
Log Output
1 item selected

2. A New User Project window will be displayed. Enter the name of the project. For example:


New User Project

New User Project

New User Project creation.

Project name: NewProject1

Project contents:

☑ Use default

Directory: C:\IBM\WebSphereICS\Tools\WSWB203\Workspace\NewProject1

3. In the Available Integration Component Libraries window you will see a list of the integration component libraries that have been detected. (Note: A path to the libraries must be entered when you install System Manager). Select the plus

signs (+) to expand the checkboxes in the Available Integration Component Libraries window and select the business object definitions and connectors you wish to include in this project.

Note: Selecting the checkbox beside ″Business Objects″ (as in the example) will select all available business object definitions. Selecting the checkbox beside ″Connectors″ (as in the example) will select all available connectors.

4. Click Finish. The name of the new project will appear in the Broker Projects list in System Manager.
5. In the User Projects panel, expand the name of the new project and the ″Business Objects″ and ″Connectors″ items that appear and you will see the names of the Business Objects and Connectors that you selected.

The remaining steps in the deployment process differ depending upon which message broker you are using as an integration broker. In the next step you will right-click on the new project name and choose one of two types of deployment, as follows:

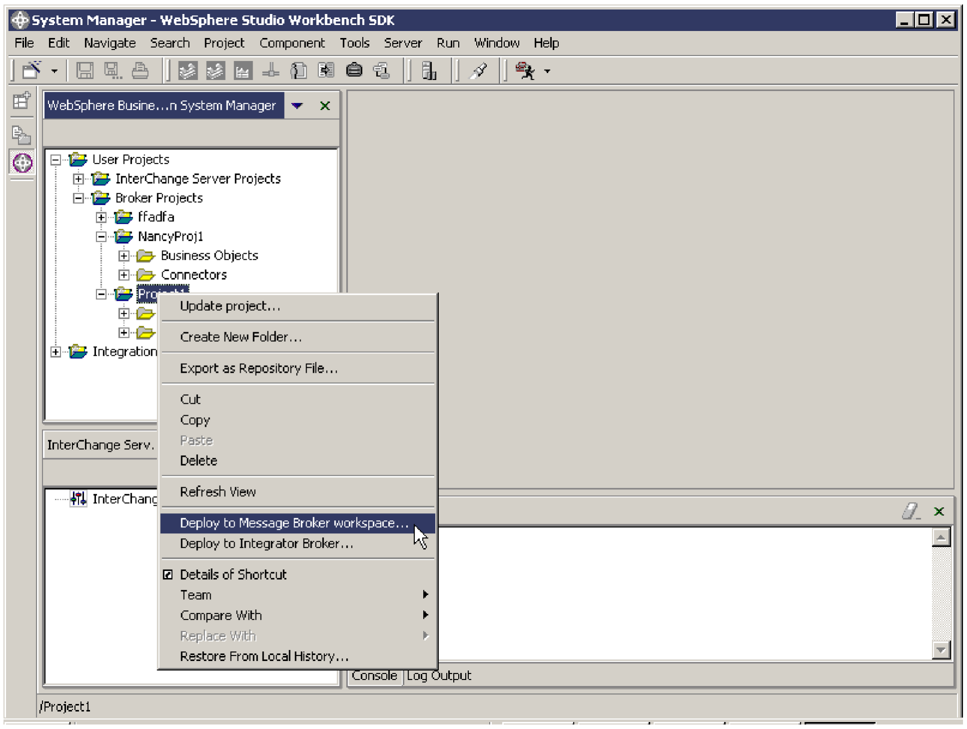
If you are using...	Select...	Refer to...
WebSphere Business Integration Message Broker	Deploy to message broker workspace	"Deploying to a message broker workspace" on page 85
WebSphere MQ Integrator	Deploy to integrator broker	"Deploying to an integrator broker" on page 88
WebSphere MQ Integrator Broker		

Deploying to a message broker workspace

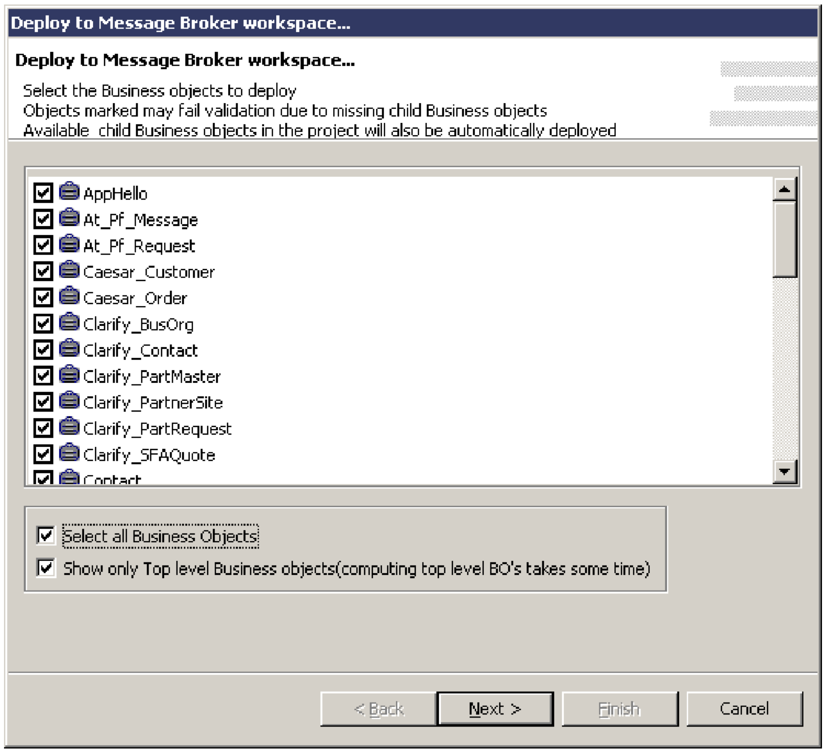
To deploy the project to a message broker workspace (for WebSphere Business Integration Message Broker used as the integration broker), do the following from System Manager:

1. Right-click on the name of the new project in the User Projects panel, then select ″Deploy to message broker workspace″ as shown in the following

illustration:


System Manager - WebSphere Studio Workbench SDK
File Edit Navigate Search Project Component Tools Server Run Window Help
WebSphere Busine...n System Manager
User Projects
InterChange Server Projects
Broker Projects
ffadfa
NancyProj1
Business Objects
Connectors
Integration
InterChange Serv.
InterChang
Update project...
Create New Folder...
Export as Repository File...
Cut
Copy
Paste
Delete
Refresh View
Deploy to Message Broker workspace...
Deploy to Integrator Broker...
Details of Shortcut
Team
Compare With
Replace With
Restore From Local History...
Console
Log Output
/Project1

The following window appears, showing the available business objects:



Deploy to Message Broker workspace...

Deploy to Message Broker workspace...

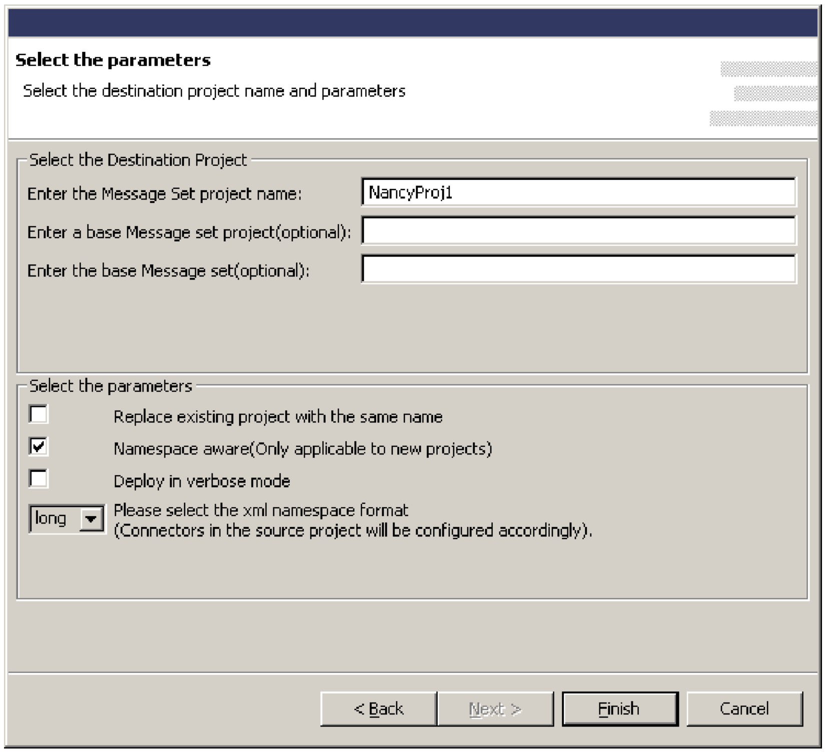
Select the Business objects to deploy
Objects marked may fail validation due to missing child Business objects
Available child Business objects in the project will also be automatically deployed

- [✓] AppHello
- [✓] At_Pf_Message
- [✓] At_Pf_Request
- [✓] Caesar_Customer
- [✓] Caesar_Order
- [✓] Clarify_BusOrg
- [✓] Clarify_Contact
- [✓] Clarify_PartMaster
- [✓] Clarify_PartnerSite
- [✓] Clarify_PartRequest
- [✓] Clarify_SFAQuote
- [✓] Contact

[✓] Select all Business Objects
[✓] Show only Top level Business objects(computing top level BO's takes some time)

2. Choose the desired business objects in the list (or "Select all business objects" or "Show only Top level Business objects").

3. Click Next. The following window appears:



Select the parameters

Select the destination project name and parameters

Select the Destination Project

Enter the Message Set project name: NancyProj1

Enter a base Message set project(optional):

Enter the base Message set(optional):

Select the parameters

- [ ] Replace existing project with the same name
- [x] Namespace aware(Only applicable to new projects)
- [ ] Deploy in verbose mode

long — Please select the xml namespace format (Connectors in the source project will be configured accordingly).

4. Enter a name for the Message Set project (required) and any other parameters (optional). The parameters are described in more detail below:

   Message set project name. Any text string designating the name of this project. This name will appear in the System Manager window when deployment is complete.

   Base message set project and base message set. If you enter names of an existing base message set project and base message set, then the contents of the base message set will be used as the basis of the new destination project. All the message set definitions in the base project will be copied to the new destination project. In addition, the destination project will also include any business objects you selected from the previous window during this deployment. You can view the names of existing message set projects and base message sets from the Broker Application Development perspective of Message Broker Tooling.

   Note: You must enter both a base message set project and a base message set; if only one of these parameters is specified, it will be ignored, and only those business objects selected during this deployment will be included in the project.

   Replace existing project with the same name. This option is applicable if the name entered in the `Enter message set project name` field is identical to the name of an existing project. If so, selecting this option replaces an existing project's content with content based on your selections during this deployment (only the name remains the same) . If this option is not checked and the specified project name matches that of an existing project, then the project retains whatever business objects it already contained and those selected during this deployment will be added. All existing business objects with the same names will be overwritten. If this option is checked but the project name does not match an existing project, this option is ignored and a new project is created.

Namespace aware and xml namespace format. By default, XML namespace format is set to short. *It is important to select the correct combination of choices for* `Namespace aware` *and* `XML namespace format`*, or your deployment might fail.* Refer to “Choosing XML Namespace length” on page 90 for information on whether to select long or short for this parameter.

Deploy in verbose mode. Selecting this option will cause more details of the deployment process to be displayed or logged during deployment.

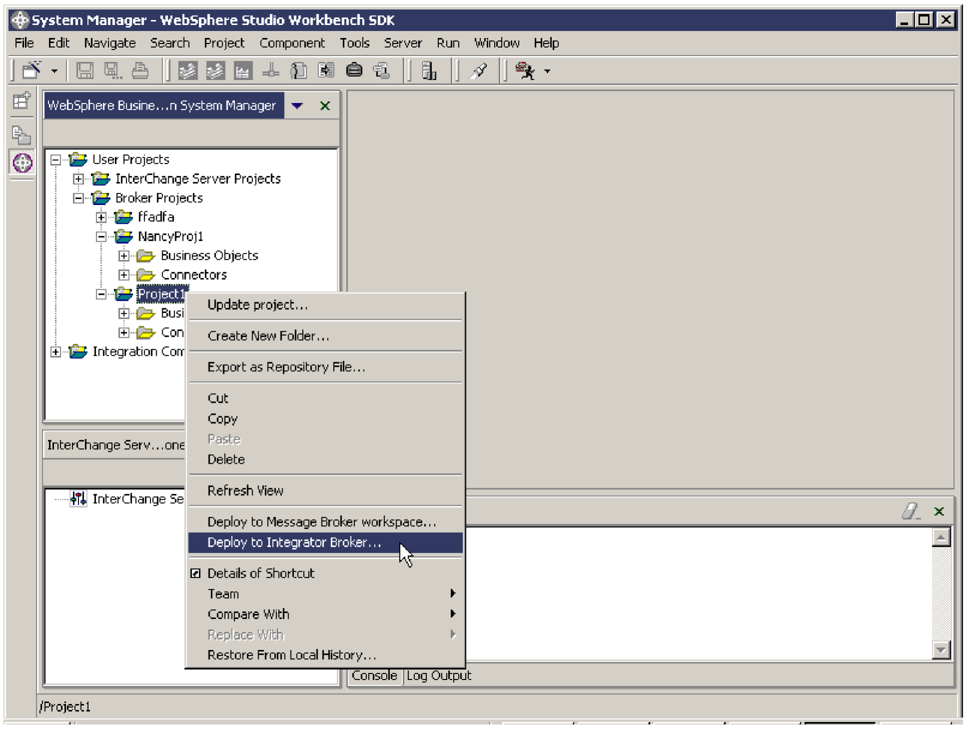
5. Click Finish. If the project deploys successfully, a window will display a message indicating successful deployment. In addition, The results of the deployment will be recorded in the default log file `mqsicreatemsgdefs.report.txt`,located in the importer directory (default directory `eclipse`). Also refer to the console panel of System Manager; it will display which business objects have successfully deployed.

   Note: The Message Broker Toolkit must closed when you click Finish for deployment to succeed. If the Toolkit was open at this time, you will receive a message stating that deployment of some or all of the business objects and message sets failed. You must redeploy from the beginning of the process, making sure that the Toolkit is closed.

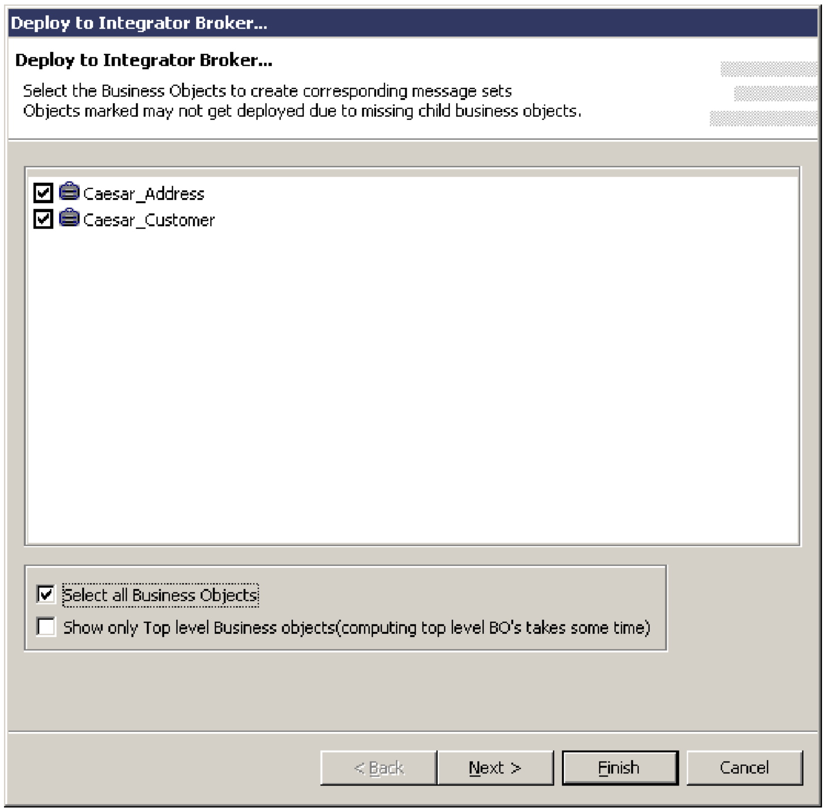
Deploying to an integrator broker

To deploy the project to an integrator broker (for WebSphere MQ Integrator or WebSphere MQ Integrator Broker used as the integration broker), do the following from System Manager:

1. Right-click on the name of the new project in the User Projects panel, then select ″Deploy to integrator broker″ as shown in the following illustration:


System Manager - WebSphere Studio Workbench SDK
File Edit Navigate Search Project Component Tools Server Run Window Help
WebSphere Busine...n System Manager
User Projects
InterChange Server Projects
Broker Projects
ffadfa
NancyProj1
Business Objects
Connectors
Project1
Busi
Con
Integration Com
InterChange Serv...one
InterChange Se
Update project...
Create New Folder...
Export as Repository File...
Cut
Copy
Paste
Delete
Refresh View
Deploy to Message Broker workspace...
Deploy to Integrator Broker...
Details of Shortcut
Team
Compare With
Replace With
Restore From Local History...
Console
Log Output
/Project1

The following window appears, showing the available business objects:



Deploy to Integrator Broker...

Deploy to Integrator Broker...

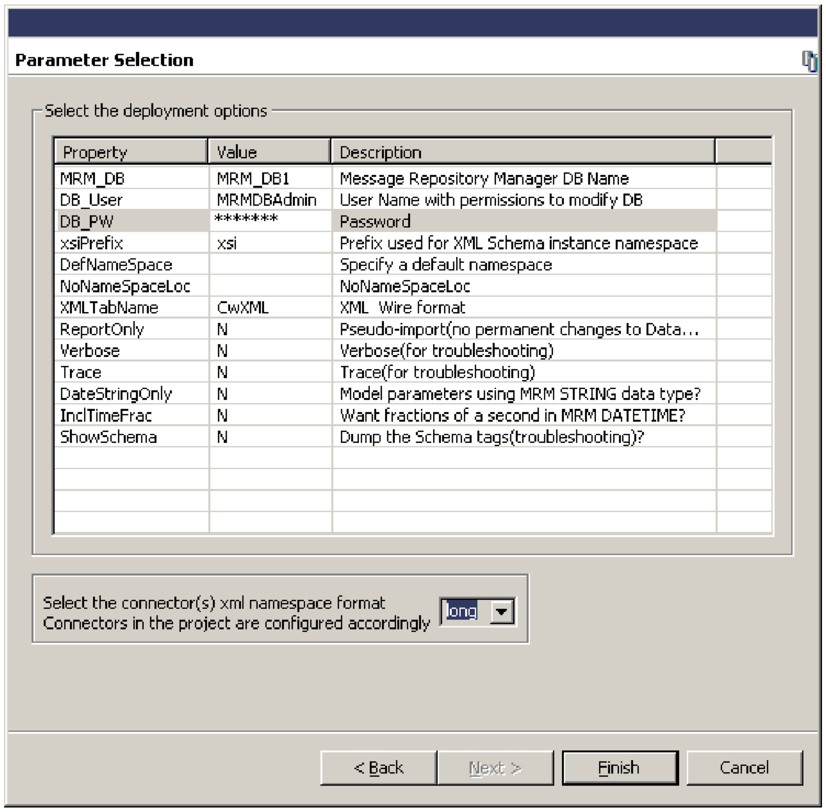
Select the Business Objects to create corresponding message sets
Objects marked may not get deployed due to missing child business objects.

[✓] Caesar_Address
[✓] Caesar_Customer

[✓] Select all Business Objects
[ ] Show only Top level Business objects(computing top level BO's takes some time)

2. Choose the desired business objects in the list (or″Select all business objects″ or ″Show only Top level Business objects″.
3. Click Next. The Parameter Selection window appears. This window allows you to change WebSphere MQ Integrator values and to select the XML namespace

format for the connector. For example:



Select the deployment options

Property	Value	Description
MRM_DB	MRM_DB1	Message Repository Manager DB Name
DB_User	MRMDBAdmin	User Name with permissions to modify DB
DB_PW	*******	Password
xsiPrefix	xsi	Prefix used for XML Schema instance namespace
DefNameSpace		Specify a default namespace
NoNameSpaceLoc		NoNameSpaceLoc
XMLTabName	CwXML	XML Wire format
ReportOnly	N	Pseudo-import(no permanent changes to Data...
Verbose	N	Verbose(for troubleshooting)
Trace	N	Trace(for troubleshooting)
DateStringOnly	N	Model parameters using MRM STRING data type?
InclTimeFrac	N	Want fractions of a second in MRM DATETIME?
ShowSchema	N	Dump the Schema tags(troubleshooting)?

Select the connector(s) xml namespace format
Connectors in the project are configured accordingly: long

4. Enter the Password (DB_PW property) and any other values to change from the defaults. (Values entered from any previous use of this window will have been saved and will be presented as the defaults.) Note that by default, XML namespace format is set to short. It is important to select the correct combination of choices for `Namespace aware` and `XML namespace format`, or your deployment might fail. Refer to "Choosing XML Namespace length" for information on whether to select long or short for this parameter.
5. Click Finish. If deployment is successful, a window will appear with a message indicating sucessful deployment. Also refer to the console panel of System Manager; it will display which business objects have successfully deployed.

Choosing XML Namespace length

When you deploy your project to a message broker, you have the option of selecting`long` or `short` `XMLNameSpace` format. The default is `short`. `XMLNameSpaceFormat` is a standard configuration connection property for adapters. The choice you make within System Manager overrides whatever might already exist in the adapter's configuration file. If you are deploying to WebSphere Business Integration Message Broker, the `Select the parameters` window allows you to specify that the project is namespace-aware. If the project is namespace-aware, either `short` or `long` formats are allowed. If the project is not namespace-aware, then `long` must be used. Selecting `short` with a project that is not namespace-aware will cause deployment to fail.

Note: IBM recommends that any new projects be set to namespace-aware mode for ongoing compatibility with industry standards. IBM recommends setting `XMLNameSpace` format to `short`, if possible, for better performance.

If you are deploying to WebSphere MQ Integrator or WebSphere MQ Integrator Broker, then `long` or `short` are allowed. However, if `short` is selected, then the

`RFH2messagedomain` property of the connector must be set to **xml**. If `RFH2messagedomain` is set to **mrm**, only XML messages in `long` namespace format can be processed.

Note: WebSphere MQ Integrator and WebSphere MQ Integrator Broker do not support namespaces. Therefore you cannot specify namespace-aware for projects deployed to these brokers.

Table 12 shows the XMLNameSpace format alternatives available:

Table 12. XMLNameSpace format alternatives

Message Broker	**Namespace aware?**	
	Yes	**No**
WebSphere Business Integration Message Broker	long or short	long
WebSphere MQ Integrator or WebSphere MQ Integrator Broker	N/A	long or short (for short, RFH2messagedomain must be **xml**)

Enabling the application for use with the connector

To allow the connector's application-specific component to deliver business data to and from the application, you must establish a dedicated user account for the connector on the application. You will need to specify the user ID and password of this account when you create the configuration file for the connector.

For most connectors, the application must be configured to implement the event detection mechanism. Once the application has been configured, it can detect entity changes and write event records to the event store. The information is then picked up by the connector and processed. You should create triggers only for business objects and operations that are to be processed by a WebSphere MQ message flow. Otherwise, the message queues will fill up with messages that are never removed for processing.

For detailed information about these tasks and others necessary to enable the application to work with the connector, refer to the adapter user guide for the connector you are configuring.

Configuring the connector

The Connector Configurator tool provides a graphical user interface for configuring the connector. When you are finished specifying values for the connector's configuration properties, the Connector Configurator generates a configuration file for the connector and places it in the connector's local repository.

Important: If the business integration adapter is running on UNIX, you must create the configuration file using Connector Configurator on Windows and then copy the file to your UNIX machine. When you create the configuration file, make sure that you observer UNIX path and file name conventions when setting properties.

When you ran the IBM WebSphere Business Integration adapter installer, it loaded a connector definition file for the connector in *`ProductDir`*`\connectors\repository\`*`ConnName`*. The connector definition file provides initial values for some configuration file properties. The next step is to use Connector Configurator to create a configuration file for the connector. While the connector definition file provides some starting values for the configuration file, the configuration file contains all the standard and application-specific properties for the connector, and specifies its supported business objects. Appendix C, "Standard configuration properties for connectors," on page 109, describes these properties in detail.

If you manually installed a new connector (one not pre-built by IBM) you must create a new configuration file for that connector, using Connector Configurator.

Tip

Use the information provided below together with the adapter user guide for the connector you are configuring, which contains:
- Complete instructions for using Connector Configurator
- Detailed information about the supported settings for the connector's standard properties, application-specific properties, and logging and tracing options.

The sections below discuss Connector Configurator settings that apply to every connector working with a WebSphere message broker.

Specifying the location of the connector's local repository

You can specify the location you want to use for the local repository using the standard property, RepositoryDirectory. The default location is *ProductDir*`\repository`.

Specifying the queues to be used by the connector

In "Configuring the WebSphere MQ queues" on page 78, you defined a set of queues to be used by the connector to communicate with the message broker. In the Connector Configurator, click the **Standard Properties** tab and assign these queues to the connector using the following standard properties:
- DeliveryQueue
- RequestQueue
- ResponseQueue (this queue is used only with WebSphere InterChange Server but must be defined for compatibility)
- FaultQueue
- SynchronousRequestQueue (needed only if the connector is using synchronous execution)
- SynchronousResponseQueue (needed only if the connector is using synchronous execution)
- AdminInQueue
- AdminOutQueue

Setting the connection mode with the queue manager

The default connection mode is bindings mode. Specify client mode as follows:
1. In the Connector Configurator, click the **Standard Properties** tab.
2. Assign to the standard property, jms.MessageBrokerName, the following value: *QueueMgrName*`:[`*Channel*`]:[`*HostName*`]:[`*PortNumber*`]`, where the variables represent the following:

QueueMgrName
The name of the queue manager.

Channel
The channel used by the client.

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

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

For example:

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

Setting configuration properties for synchronous execution

If your connector uses synchronous execution, click the Standard Properties tab and set the following properties:

- SynchronousRequestQueue = *SynchronousRequestQueue*
- SynchronousResponseQueue = *SynchronousResponseQueue*
- SynchronousRequestTimeout = 10000

Configuring logging and tracing options

Logging is used to communicate system messages, component state changes, failures, and tracing information. The following files are generated:

Table 13. Connector logging and tracing files.

Default file name and path	Description
Temporary log file: *ProductDir*\broker_name_connector_name_tmp.log.	During startup, the connector generates a temporary log file. This file contains all messages that are logged during startup, including connector properties and business object definitions that are passed to the connector framework. and the file is written to the directory in which the product is installed.
Connector log file: **UNIX:** A connector logs messages to STDOUT by default, then those messages are rerouted to *ProductDir*/logs/connector_manager_*ConnName*.log. **WINDOWS:** A connector logs messages to `STDOUT` by default, but can be configured to send to a local destination log file in the *ProductDir* directory.	The connector's log file is used to store messages issued by the connector. It also contains messages regarding WebSphere MQ communication errors.
Connector message file: *ProductDir*\connectors\messages\*ConnName_LocaleName*.txt	This file contains the full text for each message issued by the connector. You can use this file to look up the text of message IDs you see in the log file. If the locale specified in the connector configuration file is not supported, the file *ConnName*.txt is used.
Trace file: Defaults to STDOUT for both UNIX and Windows.	Contains trace messages as specified by the selected trace level.

The logging system is always active and provides an accurate monitor of the connector.

To troubleshoot a problem, you can turn on tracing. Trace messages help you monitor actions taken in components of the business integration system. Trace levels define the amount of detail written to the trace file. The higher the trace level, the more detail you receive. Tracing differs from logging in the following ways:

- Logging always occurs, but tracing can be turned on and off as needed.
- Tracing contains more detailed information than logging about the state of components and the actions taken by them.
- Logging and tracing settings are persistent after reboots.

Tracing is off by default because it produces messages that are more detailed than you normally need.

For information about viewing logging and tracing messages using LogViewer, see "Using Log Viewer to view connector messages" on page 69.

Configuring connector logging

To configure connector logging options, click the **Trace/Log Files** tab and specify the following:

1. To have log messages routed to STDOUT, click the **To console (STDOUT)** check box.
2. To have log messages routed to a file, click the **To file** check box and specify the full-path name of the log file you want to use. You can have log messages routed to the console and to a file by specifying both the "To console" and "To file" options.
3. If you designated the use of a log file, also specify the following options:
   a. To limit the size of the log file, set **Log file size** to a numeric value and unit of measure.
   b. To permit the log file to grow with no limit, click the **Unlimited** check box.
   c. If you have set a maximum size for the log file and you want to use file archiving, set **Number of archives** to the number of archive files you want to maintain.

For more information about managing log files, see "Managing log and trace files" on page 67.

Configuring connector tracing

To configure connector tracing options:

1. Click the **Trace/Log Files** tab.
2. To have trace messages routed to STDOUT, check **To console (STDOUT)**.
3. To have trace messages routed to a file, check **To file** and specify the full-path name of the trace file you want to use. You can have trace messages routed to the console and to a file by specifying both the **To console** and **To file** options.
4. If you designated the use of a trace file, also specify the following options:
   a. To limit the size of the trace file, set **Trace file size** to a numeric value and unit of measure.
   b. To permit the trace file to grow with no limit, check **Unlimited**.

c. If you have set a maximum size for the trace file and you want to use file archiving, set **Number of archives**to the number of archive files you want to maintain.

5. To set the tracing level:
   a. Click the **Standard Properties** tab.
   b. Set the **AgentTraceLevel** property to one of the values listed in Table 14.

Refer to the adapter user guide for the connector you are configuring for more details about the information generated by the different trace levels for that connector. You can set connector tracing to one of the following levels:

Table 14. Connector trace levels.

Trace level	Description
1	Traces initialization and the sending and receipt of business objects.
2	Prints messages for level 1. In addition, provides more details than Level 1 for the same types of events.
3	Prints messages for levels 1 and 2. In addition, traces the exchange of messages between the connector agent and the messaging driver.
4	Prints messages for levels 1 through 3. In addition, traces the passing of business objects between internal levels of the connector.
5	Prints messages for levels 1 through 4. In addition, traces the passing of administrative messages between internal levels of the connector.

A new or changed tracing level takes effect when you restart the connector.

For more information about archiving trace files, see "Managing log and trace files" on page 67.

Configuring the connector startup files, shortcuts, and environment variables

The procedure for starting a connector and the set-up tasks required both depend on the platform where the connector is running.

For Windows

When you install WebSphere Business Integration Adapters on Windows, a shortcut is created for each installed connector on the WebSphere Business Integration Adapters program menu (`Start > Programs > IBM WebSphere Business Integration Adapters > Adapters > Connectors`). You must use the startup options listed in Appendix D, "Connector startup options," on page 125 to customize the following:

- The connector's shortcut properties
- The connector's startup file, `start_`*`connName`*`.bat` (for connectors written in Java)
- The connector's startup file, `start_connector.bat` file (for connectors written in C++).

For UNIX

In the UNIX environment, you start a connector by running connector_manager_`connName` script, which is a wrapper for the generic connector manager script (*`ProductDir`*`/bin/connector_manager`). This wrapper includes the following information:

- The name of the connector to start or stop

- Appropriate command-line options of the generic connector manager. For example:
  - The SAP connector requires the -t command-line option. Therefore, its startup script already includes the -t option.
  - All UNIX connectors run with the -b option. Therefore, all connector startup scripts already include the -b option. To have a connector run in the foreground, remove the -b option from the generic connector manager script (`connector_manager`).
- The name of the configuration file.

If you have created a custom adapter or if you have installed an adapter using electronic software delivery (ESD), you need to do the following before you start up the connector for the first time:

1. Run the Connector Script Generator tool to update the `connector_manager_`*`connNam`*e script with the name of the connector's configuration file. See Appendix F, "Using the Connector Script Generator tool," on page 157, for more information about running this tool.

   Alternatively, you can navigate to the *`ProductDir`*`/bin` directory and edit the `connector_manager_`*`connName`* file to specify the name of the connector's configuration file. In the file, locate the AGENTCONFIG_FILE variable and set it to the full-path name of the configuration file as follows:

   ```
   AGENTCONFIG_FILE=ConfigFile
   ```
2. If desired, update your PATH environment variable to include the *`ProductDir`*`/bin` directory.
3. Ensure that the `CWSharedEnv.sh` file is sourced from the shell startup script (such as `.cshrc`) for your account.

Customizing the startup script: The generic connector manager script calls the appropriate `start_`*`connector`*`.sh` script, which is the actual script that manages the particular connector. Each WebSphere Business Integration adapter includes a `start_`*`connector`*`.sh` script. You can modify the `start_`*`connector`*`.sh` script to include any of the supported startup options listed in Appendix D, "Connector startup options," on page 125.

Note: For information about creating a startup files for connectors, see *Connector Development Guide for C++* or *Connector Development Guide for Java*.

Defining message flows

When a WebSphere Business Integration adapter uses a message broker as its integration broker, it uses WebSphere MQ message flows to process and route business object messages representing data or requests being sent by business applications to one another. A single message flow, defined for each queue, processes all messages placed on that queue. Using the MQ Integrator Broker Control Center (for WebSphere MQ Integrator Broker or WebSphere MQ Integrator) or the Message Brokers Toolkit (for WebSphere Business Integration Message Broker), you can build message flows from message-processing primitives to allow processing decisions to be made on either the message header or the message content. That is, the message flow can specify different processing steps for each type of message it is expected to handle.

Each business message (sent by the connector framework to the integration broker or from the integration broker to the connector framework) includes message header and message descriptor information that identifies the business object the message represents.

Before you define the message flows for the business integration system, you need to identify the business objects that will be processed from each queue. You also need to familiarize yourself with the message descriptor, message header, and message format for each type of message the message flows will process. Refer to "Interfaces for message exchange" on page 40 and Appendix A, "WebSphere MQ message formats," on page 99 for more information.

For detailed information on creating message flows, see your message broker documentation.

Transaction management

WebSphere message brokers offer distributed transaction support. Message flows can execute within a globally coordinated transaction, in which messages received from and sent to WebSphere MQ queues can be coordinated with any database updates performed during the processing of a message. Message flows can use the features of WebSphere MQ to act as a Transaction Manager to coordinate database updates within the WebSphere MQ unit of work. See your message broker's administration documentation for further information about implementing transaction management in your business integration system.

Using Visual Test Connector to verify your interfaces

Visual Test Connector simulates the activities of a connector to allow you to test your integration components without the complexity of running an actual connector. You can use Visual Test Connector to verify that you have configured your source and destination connectors correctly and that you have properly specified their supported business object definitions. See Appendix G, "Using Visual Test Connector," on page 159 for detailed information on how to use Visual Test Connector.

Appendix A. WebSphere MQ message formats

Each message contains three components: a message descriptor (MQMD), a message header (MQRFH2), and a message body.

Message descriptor

The WebSphere MQ message descriptor (MQMD) contains the message ID and includes information needed for processing the message.

Message header

The MQRFH2 message header carries JMS-specific data that is associated with the message content. It can also carry additional information that is not directly associated with JMS. The message header contains the following folders:

- The <mcd> folders contains properties that describe the “shape” or “format” of the message.
- The <jms> folder is used to transport JMS header fields, and JMS properties that cannot be fully expressed in the MQMD. This folder is always present in the messages implemented using JMS, which are sent by the connector framework. However, in the business integration system, this folder is irrelevant and is omitted by the WebSphere message broker when sending messages to the connector framework.
- The <usr> folder is used to transport any application-defined properties associated with the message. This folder is only present if the application has set some application-defined properties. In the business integration system, this folder is used to send return status information in a response message. The tables below identify the types of messages that require this folder.

Message body

The message body is formatted as specified by the XML schema specified for the message. In order for the data handler to find and use the correct XML schema for formatting a message, the following three names must be the same:

- The name of the XML schema stored in the connector’s repository
- The name of the XML schema imported into a WebSphere message broker’s message repository and saved as a message set definition.
- The value of messagetype in the message’s MQRFH2 message header.

The message formats and the settings for particular properties for the different types of messages exchanged by the connector framework and WebSphere MQ Integrator are listed in the tables below.

Table 15. Format and property settings of event delivery messages from the connector framework to a WebSphere message broker.

MQMD	Contains no relevant information.
RFH2 message header	In the <mcd> folder, the following fields contain information that identifies the message, its format, and how it needs to be parsed: • **Message domain** is set to **mrm**by default. This specifies that the integration broker should use the parser specifically for MRM-managed messages (those that are fully modelled in the message repository). This setting can be changed to **xml** by changing the RHF2MessageDomain standard connector configuration property. Refer to ″Appendix C, “Standard configuration properties for connectors”″ for more information. • **Message type** identifies the name of the highest-level business object represented by the message. • **Message set** identifies the message set this specific message is associated with. A separate message set is created for each type of business object. • **Message format** is set to **CwXML**.
Message body	Contains an XML instance document that conforms to the XML schema and the imported message broker message set definition for the business object specified by the message type in the RFH2 header.

Table 16. Format and property settings of request messages from the connector framework to a WebSphere message broker.

MQMD	Reply-to information is located in two fields: **ReplyToQ** and **ReplyToQMgr**. They contain the queue name and queue manager name to which the integration broker needs to direct the response message. In JMS messages, these fields specify the JMSReplyTo destination on the request message. MessageID (JMSMessageID) contains a unique value, which is copied to the CorrelID property field on the response message.
RFH2 message header	In the <mcd> folder, the following fields contain information that identifies the message, its format, and how it needs to be parsed: • **Message domain** is set to **mrm** by default. This specifies that the integration broker should use the parser specifically for MRM-managed messages (those that are fully modelled in the message repository).This setting can be changed to **xml** by changing the RHF2MessageDomain standard connector configuration property. Refer to ″Appendix C, "Standard configuration properties for connectors"″ for more information. • **Message type** identifies the name of the highest-level business object represented by the message. • **Message set** identifies the message set this specific message is associated with. A separate message set is created for each type of business object. • **Message format** is set to **CwXML**. In the <jms> folder, the **Rto** (JMSReplyTo) field contains a URI that encodes the queue name and queue manager name to which the integration broker needs to direct the response message. See *WebSphere MQ: Using Java* for information about how this URI is specified. The reply-to information in **Rto** and in **ReplyToQ/ReplyToQMgr** in the MQMD are the same.
Message body	Contains an XML instance document that conforms to the XML schema and the imported message broker message set definition for the business object specified by the message type in the RFH2 header.

Table 17. Format and property settings of response messages from a WebSphere message broker to the connector framework.

MQMD	The **CorrelID** field contains the message ID of the request to which the integration broker is responding. For JMS messages, this field is used to define JMSCorrelationID.
RFH2 message header	In the <mcd> folder, the following fields contain information that identifies the message, its format, and how it needs to be parsed: • **Message domain** is set to **mrm** by default. This specifies that the integration broker should use the parser specifically for MRM-managed messages (those that are fully modelled in the message repository). This setting can be changed to **xml** by changing the RHF2MessageDomain standard connector configuration property. Refer to ″Appendix C, "Standard configuration properties for connectors"″ for more information. • **Message type** identifies the name of the highest-level business object represented by the message. • **Message set** identifies the message set this specific message is associated with. A separate message set is created for each type of business object. • **Message format** is set to **CwXML**. In the <usr> folder, the following fields contain return status information: • **Status** field contains a string with return status information. Possible string values are: **-1:** The requested operation failed. **0:** The requested operation succeeded. **1:** The requested operation succeeded. The application has returned a changed business object. • **Description** field - When status is set to -1, it contains an extended error string with the message sent by the integration broker.
Message body	Contains an XML instance document that conforms to the XML schema and the imported message broker message set definition for the business object specified by the message type in the RFH2 header.

Table 18. Format and property settings for request messages sent from a WebSphere message broker to the connector framework.

MQMD	Reply-to information is located in two fields: **ReplyToQ** and **ReplyToQMgr**. They contains the queue name and queue manager name to which the integration broker needs to direct the response message. In JMS messages, these fields specify the JMSReplyTo destination on the request message. If the ReplyToQ and ReplyToQMgr fields are left blank, the connector framework is not expected to provide a response. If a response is required, messages can also specify reply-to information in the **Rto** property field of the message header. MessageID (JMSMessageID) contains a unique value, which is copied to the CorrelID property field on the response message.
RFH2 message header	In the <mcd> folder, the following fields contain information that identifies the message, its format, and how it needs to be parsed: • **Message domain** is set to **mrm**. This specifies that the integration broker should use the parser specifically for MRM-managed messages (those that are fully modelled in the message repository).This setting can be changed to **xml** by changing the RHF2MessageDomain standard connector configuration property. Refer to ″Appendix C, “Standard configuration properties for connectors”″ for more information. • **Message type** identifies the name of the highest-level business object represented by the message. • **Message set** identifies the message set this specific message is associated with. A separate message set is created for each type of business object. • **Message format** is set to **CwXML**. In the <jms> folder, the **Rto** (JMSReplyTo) property field can optionally contain the queue name and queue manager name to which the connector framework needs to direct the response message.
Message body	Contains an XML instance document that conforms to the XML schema and the imported message broker message set definition for the business object specified by the message type in the RFH2 header.

Table 19. Format and property settings for response messages sent from the connector framework to a WebSphere message broker.

MQMD	The **CorrelID** property field contains the message ID of the request to which the connector framework is responding.
RFH2 message header	In the <mcd> folder, the following fields contain information that identifies the message, its format, and how it needs to be parsed: • **Message domain** is set to **mrm**. This specifies that the integration broker should use the parser specifically for MRM-managed messages (those that are fully modelled in the message repository).This setting can be changed to **xml** by changing the RHF2MessageDomain standard connector configuration property. Refer to ″Appendix C, “Standard configuration properties for connectors”″ for more information. • **Message type** identifies the name of the highest-level business object represented by the message. • **Message set** identifies the message set this specific message is associated with. A separate message set is created for each type of business object. • **Message format** is set to **CwXML**. In the <usr> folder, the following fields contain return status information: • **Status** field contains a string with a return status indicator. Possible string values are: **-1:** The requested operation failed. **0:** The requested operation succeeded. **1:** The requested operation succeeded. The application has returned a changed business object. • Description property field - When status is set to -1, it contains an extended error string with the message sent by the connector framework.
Message body	Contains an XML instance document that conforms to the XML schema and the imported message broker message set definition for the business object specified by the message type in the RFH2 header.

Table 20. Format and property settings for administrative messages sent from the connector framework to a WebSphere message broker.

MQMD	Contains no relevant information.
RFH2 message header	In the <mcd> folder, the following fields contain information that identifies the message, its format, and how it needs to be parsed: • **Message domain** is set to **xml** to indicate that the message should be parsed by the WebSphere message broker’s generic XML parser.
Message body	See Appendix D, “Connector startup options,” on page 125 for information about message body content.

Table 21. Format and property settings for administrative messages sent from a WebSphere message broker to the connector framework

MQMD	If the administrative message is Stop Connector, the Format property is set to: MQC.MQFMT_STRING and the Expiry (JMSExpiration) property field is set to one minute.
Message body	See Appendix D, "Connector startup options," on page 125 for information about message body content.

Appendix B. WebSphere MQ message body formats for administrative messages

This appendix includes sample message body contents for administrative messages exchanged between the connector framework and WebSphere message brokers.

Messages from the connector framework to WebSphere message brokers

The connector framework sends the integration broker a message with the following message body contents if an error requires the connector to shut down:

```
<?xml version="1.0" encoding="UTF-8"?>
<CwConnectorCommand
<CwConnectorCommand
   xmlns="http://www.ibm.com/websphere/crossworlds/2002/CwConnectorCommand"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.ibm.com/websphere/crossworlds/2002/
             CwConnectorCommand file:CwConnectorCommand.xsd">
   <Source>Connector1</Source>
   <Destination>IntegrationBroker</Destination>
   <Command>Shutdown</Command>
</CwConnectorCommand>
```

Messages from WebSphere message brokers to the connector framework

The integration broker sends the connector framework a message with the following message body contents to initiate a shutdown of the connector:

```
<?xml version="1.0" encoding="UTF-8"?>
<CwConnectorCommand
   xmlns="http://www.ibm.com/websphere/crossworlds/2002/CwConnectorCommand"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.ibm.com/websphere/crossworlds/2002/
             CwConnectorCommand file:CwConnectorCommand.xsd">
   <Source>IntegrationBroker</Source>
   <Destination>Adapter</Destination>
   <Command>Shutdown</Command>
</CwConnectorCommand>
```

Appendix C. 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

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 22 on page 111 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 22. 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 22. 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 22. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
OADMaxNumRetry	A positive number	1000	Dynamic	Repository Directory must be <REMOTE>
OADRetryTimeInterval	A positive number in minutes	10	Dynamic	Repository Directory must be <REMOTE>
PollEndTime	HH:MM	HH:MM	Component restart	
PollFrequency	A positive integer in milliseconds no (to disable polling) key (to poll only when the letter p is entered in the connector's Command Prompt window)	10000	Dynamic	
PollQuantity	1-500	1	Agent restart	JMS transport only: Container Managed Events is specified
PollStartTime	HH:MM(HH is 0-23, MM is 0-59)	HH:MM	Component restart	
RepositoryDirectory	Location of metadata repository		Agent restart	For ICS: set to <REMOTE> For WebSphere MQ message brokers and WAS: set to C:\crossworlds\repository
RequestQueue	Valid JMS queue name	*CONNECTORNAME*/REQUESTQUEUE	Component restart	Delivery Transport is JMS
ResponseQueue	Valid JMS queue name	*CONNECTORNAME*/RESPONSEQUEUE	Component restart	Delivery Transport is JMS: required only if Repository directory is <REMOTE>
RestartRetryCount	0-99	3	Dynamic	
RestartRetryInterval	A sensible positive value in minutes: 1 - 2147483547	1	Dynamic	
RHF2MessageDomain	mrm, xml	mrm	Component restart	Only if Delivery Transport is JMS and WireFormat is CwXML.

Table 22. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
SourceQueue	Valid WebSphere MQ name	*CONNECTORNAME*/SOURCEQUEUE	Agent restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	Component restart	Delivery Transport is JMS
SynchronousRequestTimeout	0 - any number (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 trueto 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.

RHF2MessageDomain

WebSphere message brokers and WAS only.

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

A sample header would look like this:

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

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

SourceQueue

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

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see "ContainerManagedEvents" on page 116.

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 isCwBO.

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 D. Connector startup options

The tables below list the command-line options you can specify when starting a connector from Windows or UNIX. Some of these options override selected property settings in the connector's configuration file.

Note: Information about the structure and development of startup scripts is available in the *Connector Development Guide for C++* or *Connector Development Guide for Java*.

Windows

You can modify the startup for a connector by editing the following to use the connector startup options listed in Table 23:

- The connector's shortcut properties.
- The connector's startup file, usually `start_`*`connName`*`.bat`.

  Note: You can determine if your connector uses this file by looking in the directory *`ProductDir`*`/connectors`. If a script named `start_`*`connName`* (where *`connName`* is the name of your connector) is present, this is the file to edit.
- The connector's startup file, `start_connector.bat` file (only for connectors written in C++ that do not use a `start_`*`connName`*`.bat` startup file.).

Note: The -c option is required to start the connector. If you use the shortcut to start the connector, the shortcut properties must include this option in the target field. Similarly, if you start the connector using `start_connector.bat` or `start_`*`connName`*`.bat`, the command line must include this option.

Table 23. Command-line connector startup options for Windows.

Option	Description
`-c` *`configFile`*	The full path name of the configuration file to be used during startup. This option is required.
`-f` *`pollFrequency`*	The amount of time between polling actions. Possible values for `pollFrequency` are: • The number of milliseconds between polling actions. • `key`: Causes the connector to poll only when you type the letter p in the connector's Command Prompt window. Enter the word in lowercase. • `no`: Causes the connector not to poll. Enter the word in lowercase. The default is 10000.
`-j`	Specifies that the connector is written in Java. This option is optional if you specify `-l` *className.*

Table 23. Command-line connector startup options for Windows. (continued)

Option	Description
`-t`	**Note:** The installer has included (or omitted) the -t option in the shortcut as required for the connector. If you start up a connector from the command line, the value you specify for -t must be the same as the -t value specified in the shortcut. Turns on the connector property, `SingleThreadAppCalls`. This property guarantees that all calls the connector framework makes to the application-specific connector code are with one event-triggered flow. The default value is `false`. This property should not be changed from its shipped value. Each connector has the appropriate setting for this option, depending on its architecture.
`-x` *`connectorProps`*	Passes application-specific connector properties to the connector. Use the format `prop_name=`*`value`* for each value you enter.

UNIX

In the UNIX environment, you start a connector by running the `connector_manager_`*`connName`* script, which is a wrapper for the generic connector manager script (*`ProductDir`*`/bin/connector_manager`). The generic connector manager script calls the appropriate `start_`*`connName`*`.sh` script, which handles the actual connector management for the connector.

Each WebSphere Business Integration adapter includes a `start_`*`connName`*`.sh` script. You can modify this script to also include any of the supported startup options listed in Table 24..

Table 24. Options for the start_connector.sh script.

Option	Description
`-f`*`poll_freq`*	The amount of time between polling actions. Possible values for `poll_freq` are: • The number of milliseconds between polling actions. • `key`: Causes the connector to poll only when you type the letter p in the connector's Command Prompt window. Enter the word in lowercase. • `no`: Causes the connector not to poll. Enter the word in lowercase. The default is 10000. The `-f` option is valid on the command-line invocation of `connector_manager_`*`connector`*. The connector manager script can pass this option to its associated `start_`*`connector`*`.sh` script. This option overrides the poll frequency specified in the connector's configuration file.

Table 24. Options for the start_connector.sh script. (continued)

Option	Description
-t*threading_type*	The threading_type option specifies the threading model. **Note:** Only use the -t option when you start a custom-developed connector. Connectors that are installed using the WebSphere Business Integration Adapters installer already have a connector_manager_*connector* startup script that specifies (or omits) the required -t option, as required by the application, on the line that starts up the connector. Possible values for threading_type are: • SINGLE_THREADED: only a single thread accesses the application • MAIN_SINGLE_THREADED: only the main thread accesses the application • MULTI_THREADED: multiple threads can access the application. The -t option is *not* valid on the command-line invocation of connector_manager_*connName*. Specify it inside the generic connector_manager script, in the invocation of the start_*connector*.sh script.

Appendix E. System Manager and the Eclipse Workbench

This Appendix provides a general guide to using features of System Manager and the Eclipse Workbench of which it is part.

Note: More detailed instructions for using some System Manager features are included elsewhere in this manual. For information about using Adapter Monitor and Fault Queue Manager, refer to "Using Adapter Monitor and Fault Queue Manager" on page 61. For information on creating a new project, refer to "Creating a message broker project" on page 82.

This chapter contains the following sections:

- "About System Manager"
- "Using System Manager" on page 131
- "Working with integration component libraries" on page 135
- "Working with user projects" on page 136
- "Working with components in integration component libraries" on page 141
- "Working with solutions" on page 144
- "Exporting components to a package using System Manager" on page 147
- "Dependencies and references" on page 148
- "Standard operations available for multiple workbench resources" on page 149
- "Using Eclipse-based workbenches" on page 150
- "Troubleshooting problems connecting to the integration broker in System Manager" on page 156

About System Manager

System Manager is a plug-in that runs within the Eclipse-based tooling frameworks named WebSphere Studio Workbench and WebSphere Studio Application Developer Integration Edition. This section provides an overview of the Eclipse framework, WSWB and WSADIE, and of the IBM WebSphere Adapter tools plug-ins.

About the Eclipse Platform

The Eclipse Platform is an open-source integrated development environment (IDE) for the creation of tools. It provides tools developers with a development kit and runtime that enables the developer to write plug-ins that allow the user to work with a particular type of resource.

IBM has two branded versions of the Eclipse platform—WebSphere Studio WorkBench (WSWB) and WebSphere Studio Application Developer Integration Edition (WSADIE).

Plug-ins

Plug-ins are the modular extensions that software vendors develop to add functionality to an Eclipse-based workbench. Plug-ins encapsulate the perspectives, editors, and views that enable users of the workbench to work with particular types of resources.

For instance, one plug-in might provide the features of a text editor. Another plug-in might provide the features of an HTML editor. The System Manager tool plug-ins provide the features to work with integration components. The benefit of this plug-in model is that the user has a single tool in which they can work with many types of resources, rather than using dedicated tools for each type of resource.

To install a plug-in, you extract one or more compressed archives that represents the plug-in into the `plugins` directory within the product directory of your workbench. The System Manager plug-ins are extracted into the `plugins` directory by the installer.

IBM provides the System Manager plug-ins with the WebSphere Business Integration Adapter framework to work with business integration resources. These plug-ins are embodied by a number of uncompressed directories in the `plugins` directory of your workbench within the `com.ibm.btools` namespace. Much of the primary interface you will use in creating integration components—the System Manager—is encompassed in the contents of the `com.ibm.btools.csm` plug-in directory, for instance.

Workbench

The workbench is the collection of perspectives, editors, and views that are active in your Eclipse-based tooling framework, which are in turn affected by the collection of plug-ins you have installed and enabled. It is a general term used to refer to the Eclipse-based interface in which you are working, independent of the fact that the interface changes depending on how you use it.

Workspace

A workspace is a container for projects. The workspace is a directory in the file system where, by default, you are prompted to store your projects.

Projects

Projects are user-defined groups of resources, and are ultimately directories in the file system.

One of your first tasks when developing a business process interface is to define an integration component library, which is a project that contains the components you develop. When you create the integration component library you specify the location in the file system where it is stored (by default this is the workspace directory). A folder is created in that location with the name you specify for the integration component library and within the library folder a number of folders is created for each type of integration component (for instance, there are folders named `BusinessObjects` and `Connectors`).

You also create projects named **user projects**. User projects are collections of shortcuts that reference integration components. You must add integration components to a user project from integration component libraries in order to deploy components to an integration broker. Besides being required to deploy components to a server, user projects are designed to allow you to functionally group components together. An integration component library is a collection of all components you might need to work with, but a user project is designed to let you group together the components you are working on for a specific interface.

Resources

Resources are projects, files, and folders that you work with in the workbench.

When you create an integration component, it is stored as a file in the appropriate folder within the integration component library project. The different types of integration components are stored with different extensions (for instance, connectors are stored with extension `.con`), but they are all stored in XML format.

Perspectives

A perspective is a grouping of editors and views designed to provide a particular user role with what it requires. For instance, the System Manager perspective provides views for working with integration component libraries.

Editors

Editors allow you to open, save, and close resources in the workbench.

Views

Views provide information about the resources with which you are working in the workbench.

System Manager, for example, has the WebSphere Business Integration System Manager view, which is the view to integration component libraries and user projects.

About WSWB and WSADIE

WebSphere Studio Workbench (WSWB) is an IBM-branded release of the Eclipse platform. IBM delivers WSWB with its integration brokers and you can install it along with the core infrastructure. WSWB is capable of running all the plug-ins necessary to develop WebSphere Business Integration Adapters integration components.

WebSphere Studio Application Developer Integration Edition (WSADIE) is an IBM-branded release of the Eclipse platform, like WSWB, but can also be used to develop new plug-ins. WSADIE is not delivered with the integration broker because the ability to create new plug-ins is not required to develop integration components. If you have it installed, however, then you can use it to run the required System Manager and Integrated Test Environment plug-ins.

About System Manager

System Manager is the perspective in which you work with the integration components and server instances in a WebSphere Business Integration Adapters business integration system. You use System Manager primarily for the following tasks:

- Launching other tools in the WebSphere Business Integration Adapters toolset
- Developing and configuring some integration components
- Deploying integration components to a repository or broker.

Using System Manager

This section describes how to start and use the System Manager perspective.

Starting System Manager

When you install an BM WebSphere integration broker, you have the option to install the WebSphere Studio Workbench with support for the plug-ins required to interact with the broker, or to install the plug-ins into an existing installation of WebSphere Studio Application Developer Integration Edition.

To start System Manager, do the following:

1. Select **Start > Programs > WebSphere Business Integration Adapters > Tools > System Manager**.
2. Select **Windows > Open Perspective > Other** from the menu bar.
3. Select `System Manager` from the list of perspectives and then click **OK**.

   WebSphere Studio Workbench starts and appears. Figure 28 shows the System Manager perspective and "System Manager interface" describes the interface and its elements.

System Manager interface

The System Manager perspective has several views and editors in the default configuration it opens with. Figure 28 shows the System Manager perspective:

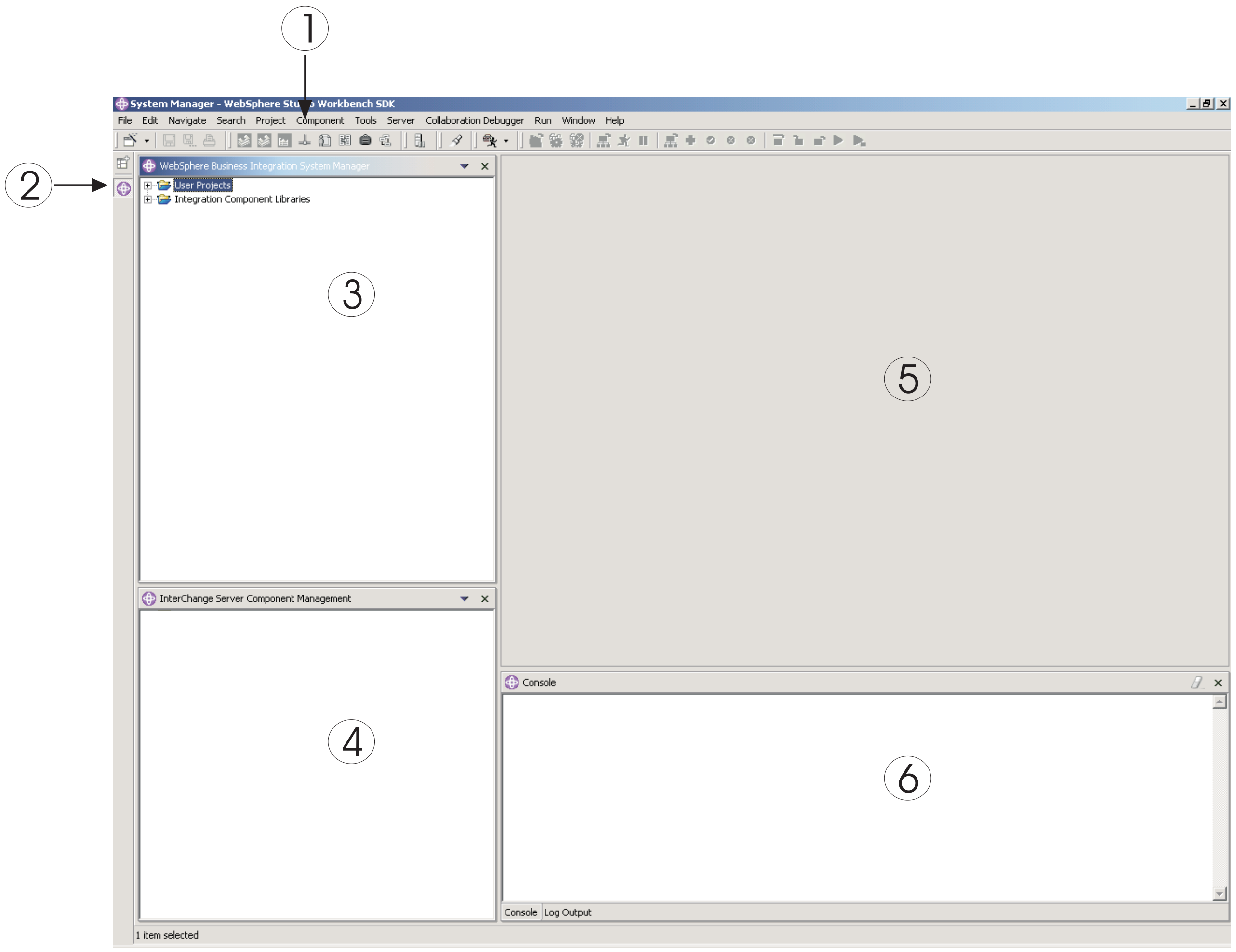

1
2
3
4
5
6
System Manager - WebSphere Studio Workbench SDK
File Edit Navigate Search Project Component Tools Server Collaboration Debugger Run Window Help
WebSphere Business Integration System Manager
User Projects
Integration Component Libraries
InterChange Server Component Management
Console
Console Log Output
1 item selected

Figure 28. System Manager perspective

Table 25 describes the interface elements of the System Manager perspective, identified by the numbers in Figure 28 on page 132:

Table 25. System Manager perspective interface elements

Interface element number	Interface element name
1	"Menu bar and toolbar" on page 133
2	"Perspective shortcut bar" on page 134

Table 25. System Manager perspective interface elements (continued)

Interface element number	Interface element name
3	"WebSphere Business Integration System Manager view" on page 134
4	Interchange Server Component Management view. This section is relevant only when running WebSphere Interchange Server as the integration broker. You can ignore it if you are using a different broker.
5	"Editor view" on page 134
6	"Console view" on page 135

The following sections describe the interface elements of System Manager in greater detail.

Menu bar and toolbar

You use the menu bar and toolbar to work with the Eclipse-based tooling framework and to work with integration broker components. Many of the menu bar items have toolbar equivalents, so the following sections only describe menu bars and their items.

File menu: This is an Eclipse-standard menu that is used to work with resources. You will use it primarily to create new integration component libraries and user projects.

For more information on creating integration component libraries and user projects, see "Working with integration component libraries" on page 135 and "Working with user projects" on page 136.

Edit menu: This is an Eclipse-standard menu that has many standard items such as Cut, Copy, and Paste.

For more information on cutting, copying, pasting, and deleting the components you create, see "Working with components in integration component libraries" on page 141.

Navigate menu: This is an Eclipse-standard menu that allows you to navigate among the resources in the workbench. For more information on this menu, see the workbench documentation.

Search menu: This is an Eclipse-standard menu that allows you to search for and search within resources. For more information on this menu, see the workbench documentation.

Project menu: This is an Eclipse-standard menu that has menu items for manipulating Project resources in the workbench; this menu item is not used when working with the System Manager perspective. For more information on this menu, see the workbench documentation.

Component menu: This menu is provided with the System Manager perspective and is useful for working with integration components you create. The items in this menu are documented throughout this guide and others in sections that describe specific tasks.

Tools menu: This menu is provided with the System Manager perspective and is used to launch tools used to create integration components.

For more information, see "Working with components in integration component libraries" on page 141.

Server menu: This menu has the **Register Servers** item, which you can use to register an InterChange Server instance. This item is not relevant if your integration broker is not Interchange Server.

Run menu: This menu has an item to configure external tools to run external programs, batch files, and build scripts. For more information, see the workbench documentation.

Window menu: This menu has items for working with perspectives, views, editors, and preferences.

These items are covered in the workbench documentation and in various sections of this guide.

Help menu: This menu has items that launch the workbench documentation and that provide version information about the workbench and perspectives.

Perspective shortcut bar

Use the perspective shortcut bar to navigate conveniently between different perspectives. You might have the System Manager and Java perspectives open; you can click on their workspace icons in the perspective shortcut toolbar to switch between them.

You can also navigate to other perspectives by using the **Window** menu:

- To navigate to a perspective represented by an icon that is higher up in the perspective shortcut bar than the icon for the perspective you are currently viewing use the keyboard shortcut **Alt+Up Arrow**.
- To navigate to a perspective represented by an icon that is lower down in the perspective shortcut bar than the icon for the perspective you are currently viewing, select **Perspective > Next** from the menu bar, or use the keyboard shortcut **Alt+Down Arrow**.

WebSphere Business Integration System Manager view

This view has the **User Projects** and **Integration Component Libraries** nodes, which are types of integration broker projects.

For more information on working with these types of projects, see "Working with integration component libraries" on page 135 and "Working with user projects" on page 136.

InterChange Server Component Management view

This view is available for those using WebSphere InterChange Server as an integration broker. It is not relevant for other integration brokers.

Editor view

This is the view in which you work with different resources in the framework, such as files and integration component definitions. Different editors open to work with different types of resources. For instance, text files are opened in a text editor.

Console view

This view has two tabs: **Console** and **Log Output**. When you compile maps or collaboration templates in System Manager, the **Console** tab displays messages to indicate whether compilation for each component completed successfully and the **Log Output** tab displays any errors or warnings encountered.

Working with integration component libraries

You use integration component libraries to store the components that you develop. This section describes how to create a new integration component library.

Once you have created an integration component library you will typically want to perform the following tasks as well:

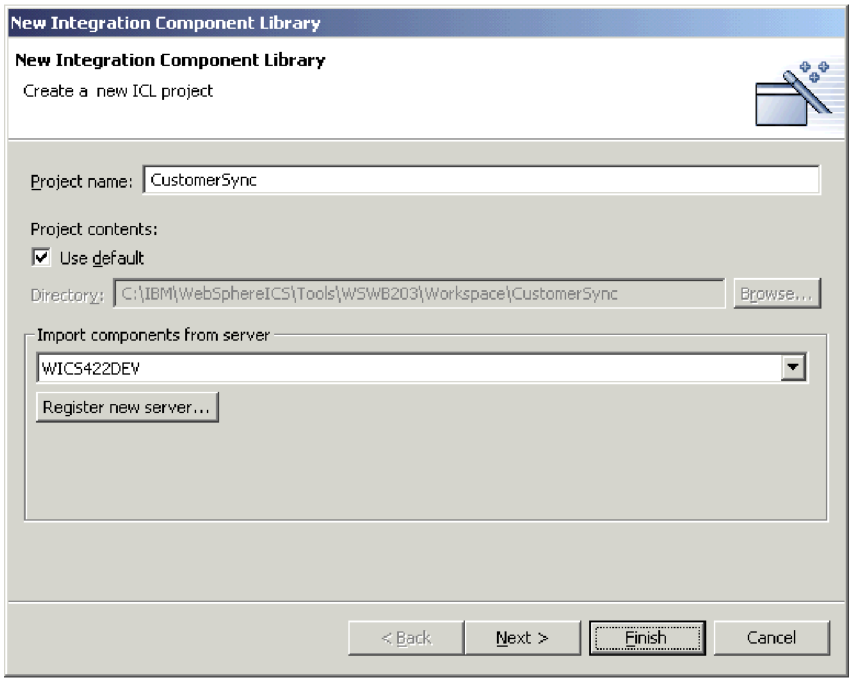
- You will want to import components into the library. For information on several ways of doing this, see “Working with components in integration component libraries” on page 141.
- You will want to create shortcuts to the components in user projects. For more information on this, see “Working with user projects” on page 136.
- You will want to deploy components to an integration broker. For more information, see “Deploying to a message broker workspace” on page 85.
- You will want to export components to a package, either to import them into servers or other libraries or to back up your development. For more information, see “Exporting components to a package using System Manager” on page 147.

Creating integration component libraries

Do the following to create a new integration component library in System Manager by using a wizard:

1. Do one of the following to start the “New Integration Component Library” wizard:
   - Select **File > New > Integration Component Library** from the menu bar.
   - In the WebSphere Business Integration System Manager view, right-click the **Integration Component Libraries** folder and select **New Integration Component Library** from the context menu.
   - Click **Open The New Wizard button** in the toolbar and select **New Integration Component Library** from the menu.

   Figure 29 on page 136 shows the “New Integration Component Library” wizard.


New Integration Component Library

New Integration Component Library

Create a new ICL project

Project name: CustomerSync

Project contents:

☑ Use default

Directory: C:\IBM\WebSphereICS\Tools\WSWB203\Workspace\CustomerSync

Import components from server

WICS422DEV

Figure 29. Creating an integration component library

2. Type a name for the integration component library in the **Project name** field.

   Project names can only contain alphanumeric characters and underscores, and must be specified in English.

   It is recommended that you name this type of library in such a way as to associate it with the server it corresponds to. For instance, if the server name is `SERV420DEV`, you might name the library `SERV420DEVICL`.
3. To have the folder for the library created in the default location (your workspace) and with a name identical to the name specified for the library, leave the **Use default location** checkbox enabled.

   If you want to specify the name and location of the library folder, do the following:

   a. Clear the **Use default location** checkbox.
   b. Type the full path and name of the directory that you want to use for the library in the **Location** field, or click **Browse** to select an existing directory.

   Note: There is no way to create the folder for a library in the path of the workspace other than to let System Manager do it by use of the **Use default location** checkbox.
4. Click **Finish** to complete the wizard.

   System Manager creates a folder with the name you specified under the **Integration Component Libraries** folder.

Working with user projects

You create shortcuts in user projects to the integration components you want to work with in one or more libraries. User projects provide a way for you to organize your view of components as an interface. You must add component shortcuts to a user project to deploy the components to a server from System Manager.

This section contains the following sections:

- "Configuring integration broker preferences for user projects"
- "Creating user projects"
- "Adding shortcuts to a user project" on page 139
- "Exporting a solution" on page 144

Configuring integration broker preferences for user projects

You can use System Manager to create user projects for multiple integration brokers. Do the following to enable System Manager for different integration brokers:

1. Open the file named `cwtools.cfg` in *ProductDir*`\bin` in a text editor.
2. To enable System Manager for WebSphere Application Server projects, set the `Installed` property in the `WAS_PROJECT` section to the value `true`.

   To enable System Manager for WebSphere MQ Integrator Broker projects, set the `Installed` property in the `WMQI_PROJECT` section to the value `true`.

   To enable System Manager for WebSphere InterChange Server projects, set the `Installed` property in the `ICS_PROJECT` section to the value `true`.
3. Save and close the file.

Creating user projects

Note: The following is a general description of how to create user projects. For specific instructions on creating a new message broker project, refer to "Creating a message broker project" on page 82.

Do the following to create a new user project in System Manager by using a wizard:

1. Do one of the following to start the "New User Project" wizard:
   - Select **File > New > User Project** from the menu bar.
   - In the WebSphere Business Integration System Manager view, right-click the **User Projects** folder, then select **New User Project**, then select **Message Broker Projects** from the context menu.
   - In the WebSphere Business Integration System Manager view, expand the **User Projects** folder, then right-click the **Message broker Projects** folder and select **New Message broker project** from the context menu.
   - Click **Open The New Wizard button** in the toolbar and select **New User Project** from the menu.
2. Type a name for the user project in the **Project name** field.

   Project names can only contain alphanumeric characters and underscores, and must be specified in English.

   It is recommended that you name this type of user project in such a way as to associate it with the server it corresponds to. For instance, if the server name is `SERV420DEV`, you might name the user project `SERV420DEVUP`.
3. To have the folder for the user project created in the default location (your workspace) and with a name identical to the name specified for the user project, leave the **Use default** checkbox enabled in the "Project contents" pane.

   If you want to specify the name and location of the user project folder, do the following:

   a. Clear the **Use default** checkbox in the "Project contents" pane.

   b. Type the full path and name of the directory that you want to use for the user project in the **Directory** field, or click **Browse** to select an existing directory.

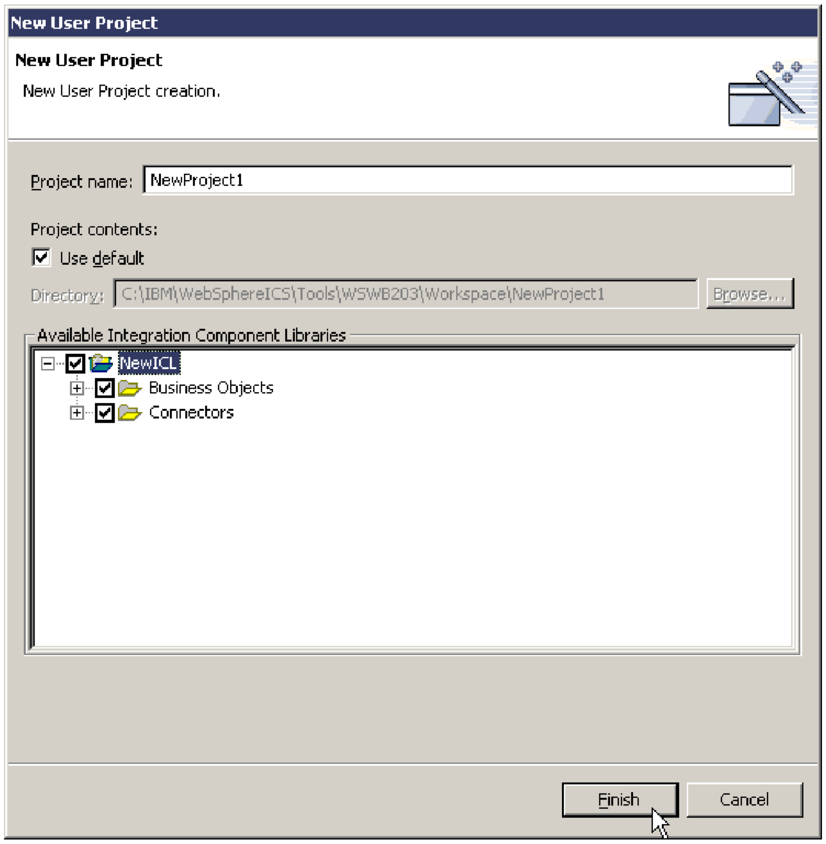
   Note: There is no way to create the folder for a user project in the path of the workspace other than to let System Manager do it by use of the **Use default location** checkbox.

4. If you do not want to create shortcuts to existing integration components at this time, proceed to step 5.

   If you do want to create shortcuts to existing integration components, enable the checkbox next to an integration component library to create shortcuts to all of the components within it or expand an integration component library folder and enable the checkboxes next to component groups, or expand the folders for groups and enable the checkboxes for individual components.

   Note: If you select components with the same names from multiple integration component libraries you do not receive a prompt to inform you that there are duplicate references in your selection. In the event that you do select duplicate components, shortcuts are created for the component in the integration component library that was furthest down in the list of libraries when you made your selections in the wizard.

   Figure 30 shows the “New User Project” wizard.



New User Project

New User Project

New User Project creation.

Project name: NewProject1

Project contents:

☑ Use default

Directory: C:\IBM\WebSphereICS\Tools\WSWB203\Workspace\NewProject1

Figure 30. Creating a user project

5. Click **Finish** to complete the wizard.

System Manager creates a folder with the name you specified in the folder corresponding to the integration broker (Message broker Projects) in the **User Projects** folder.

Adding shortcuts to a user project

You add shortcuts to a user project to give yourself a view to an interface that you are working on. There are several ways to add shortcuts to a user project, described in the following sections:

- "Using the Dependency Tree"
- "Using the Update Project wizard" on page 140
- "Dragging-and-dropping components" on page 141
- "Importing a solution" on page 146
- "Importing a solution" on page 146

Using the Dependency Tree

The "Dependency Tree" wizard is the most convenient interface for adding component shortcuts to a user project. User projects are primarily designed to represent interfaces, and interfaces are generally centered around a connector object. User projects, then, are generally centered around a connector object as well, and you can typically create the shortcuts you need for a user project by discovering the dependencies of the connector object.

For conceptual information about dependencies, see "Dependencies and references" on page 148.

Do the following to add shortcuts to a user project by using the "Dependency Tree" wizard:

1. Right-click an integration component, such as a connector object, in a library and choose **Show Dependencies** from the context menu.
2. Select the user project to which you would like shortcuts to be added from the **Add to the project** drop-down menu.
3. Select components for which you would like to create shortcuts in the left-hand pane of the wizard.

   You can use keyboard shortcuts to facilitate the process, such as holding down **Shift** to select ranges of objects and holding down **Ctrl** to select single non-contiguous objects.
4. Click the right-facing arrow to add the components to the right-hand pane of the wizard.

   Figure 31 on page 140 shows the "Dependency Tree" wizard:

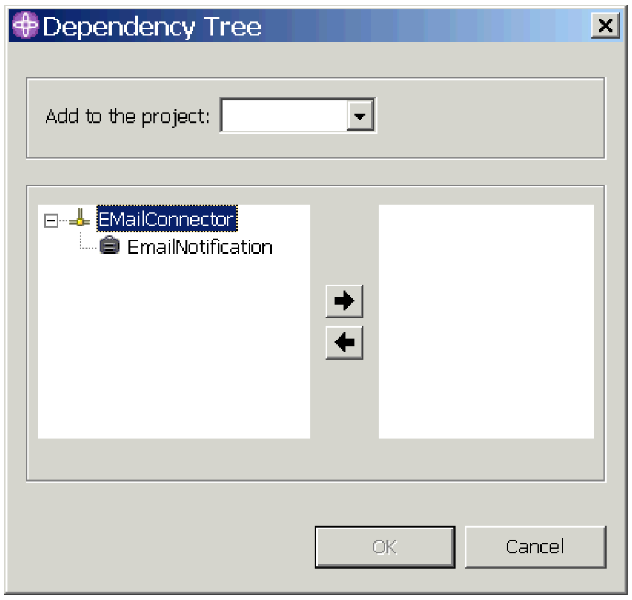


Figure 31. Adding shortcuts to a user project by using the Dependency Tree

5. Click **OK**.

   System Manager creates shortcuts to the selected components in the specified user project.

Using the Update Project wizard

The “Update Project” wizard provides an interface to add shortcuts to a user projects that is similar to the one displayed when you initially create a user project. Do the following to use the “Update Project” wizard:

1. Right-click any user project in the WebSphere Business Integration System Manager view and choose **Update project** from the context menu.
2. Enable the checkbox next to an integration component library to create shortcuts to all of the components within it or expand an integration component library folder and enable the checkboxes next to component groups, or expand the folders for groups and enable the checkboxes for individual components.
3. Click **Finish**.

   If there are shortcuts in the user project to components with the same name as those you selected in the wizard, then the wizard displays a prompt that allows you to do the following:

   - Overwrite the displayed component.
   - Overwrite all duplicate components.
   - Not overwrite the component.
   - Cancel the update operation.

   If you did not select any components with the same name as components for which shortcuts already exist in the user project, then shortcuts are added to the project and the wizard exits.

Dragging-and-dropping components

You can select components from integration component library folders and drag-and-drop them into a user project to add shortcuts to those components to the user project.

If there are no shortcuts in the folder of the user project currently then you must drag-and-drop the components onto the folder itself. Drag-and-drop the components onto the folder and release the mouse button when a square appears beneath the mouse pointer.

If there are already shortcuts in the folder of the user project then you cannot drag-and-drop components onto the folder. Drag-and-drop the components between existing shortcuts in the folder until a line appears and then release the mouse button.

Working with components in integration component libraries

The majority of your time is spent working with integration components when you implement a WebSphere Business Integration system. Although this guide does not go into detail about how to develop individual components, this section does cover how to launch the designer tools, how to start creating new components, how to start modifying existing components, and how to work with the few components that are developed in System Manager.

For more information on how to develop integration components, see the following guides:

- *Business Object Development Guide*
- *Connector Development Guide for Java or C++*

Launching designer tools

This section describes the different ways you can launch each of the designer tools. You can use the designer tools to create new components or to open and modify existing components.

Note: If you attempt to launch one of the designer tools and experience an error about a class not being found, you must launch System Manager and then try to launch the designer tool again. System Manager does not have to remain running after the tool is initially launched, however.

Business Object Designer

To launch Business Object Designer, do one of the following:

- Right-click the **Business Objects** folder in the WebSphere Business Integration System view and choose **Create New Business Object** from the context menu
- Select any folder in the WebSphere Business Integration System Manager view and do one of the following:
  - Select **Tools > Business Object Designer** from the menu bar
  - Click the **Business Object Designer** toolbar button
  - Use the keyboard shortcut **Ctrl+4**
- Select **Start > Programs > IBM WebSphere Business Integration Adapters> Tools > Business Object Designer**

For more information on Business Object Designer, see the *Business Object Development Guide*.

Connector Configurator

To launch Connector Configurator, do one of the following:

- Right-click the **Connectors** folder in the WebSphere Business Integration System view and choose **Create New Connector** from the context menu
- Select any folder in the WebSphere Business Integration System Manager view and do one of the following:
  - Select **Tools > Connector Configurator** from the menu bar
  - Click the **Connector Configurator** toolbar button
  - Use the keyboard shortcut **Ctrl+1**
- Select **Start > Programs >IBM WebSphere Business Integration Adapters> Tools > Connector Configurator**

For more information on Connector Configurator, see any WebSphere Business Integration adapter user guide or the *Connector Development Guide for Java or C++*.

Creating new components

Launching their respective designer tools as described in "Launching designer tools" on page 141 allows you to create new components. For example, for new Business Objects launch Business Object Designer and for new connectors launch Connector Configurator. Refer to the documentation for each of these tools for more information.

Modifying existing components

- Double-click the component in a library or the shortcut to a component in a user project.
- Select the component in a library or the shortcut to a component in a user project and do one of the following:
  - Launch its designer tool as described in "Launching designer tools" on page 141
  - Press **Enter**
  - Press **Ctrl+E**
  - Select **Component > Edit Definitions** from the menu bar
- Right-click the component in a library or the shortcut to a component in a user project and choose **Edit definition** from the context menu
- Launch the component's designer tool as described in "Launching designer tools" on page 141 and then open the component after the tool has started.

Importing components into a library from a package

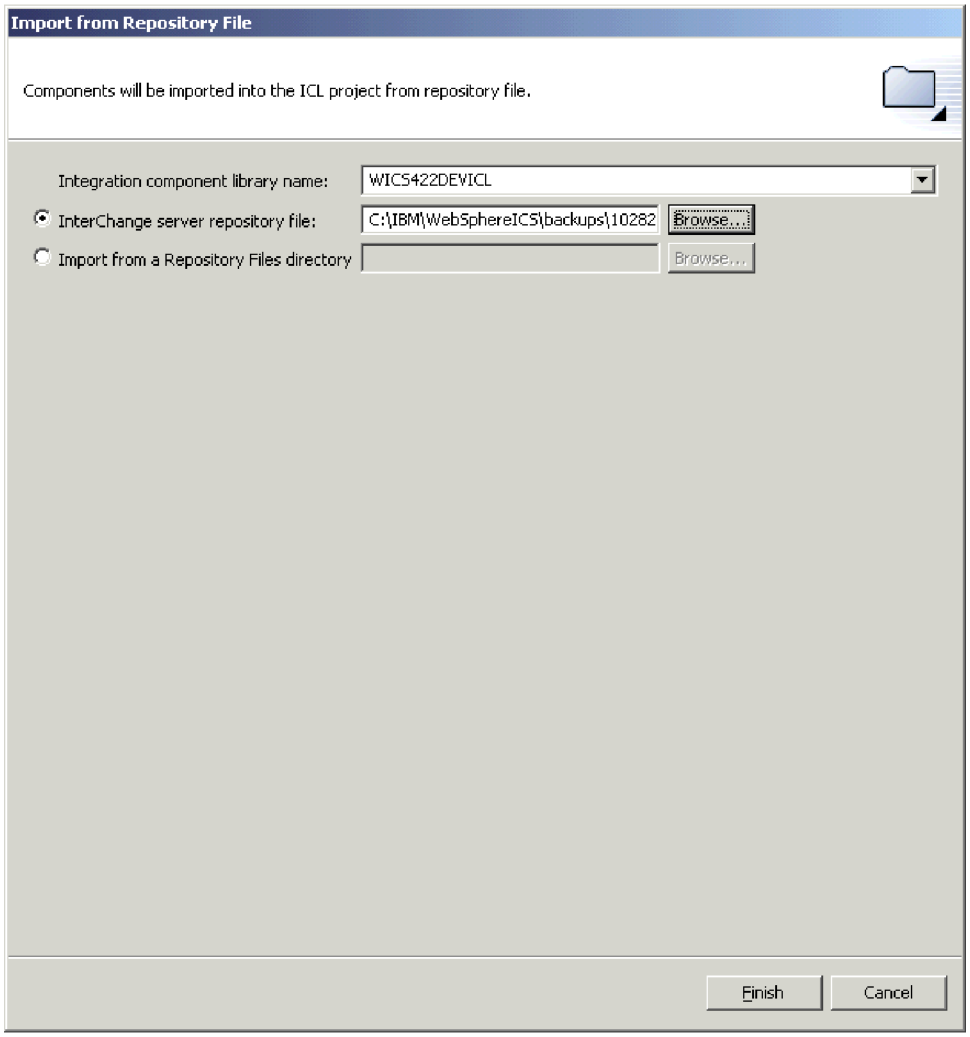
You can export integration components to a `.jar` file package, as described in "Exporting components to a package using System Manager" on page 147. This makes it easy to migrate components between environments, share them with other developers, and submit them to technical support.

Do the following to import components into an integration component library from a package:

Important: If there are components with the same name as those in the package you are importing, System Manager overwrites the existing components without a warning.

1. Right-click an integration component library and choose **Import from Repository File** from the context menu.

System Manager displays the “Import Repository File” wizard, as shown in Figure 32.


Import from Repository File

Components will be imported into the ICL project from repository file.

Integration component library name: WICS422DEVICL

(●) InterChange server repository file: C:\IBM\WebSphereICS\backups\10282

( ) Import from a Repository Files directory

Figure 32. Importing a package

2. At the “Import Repository File” screen, ensure that the Integration Component Library Name drop-down menu contains the name of the library into which you want to import the components.

   If you launched the “Import from Repository File” wizard from a library other than the one into which you want to import the components, you can change the destination this way instead of closing the wizard and launching it again.
3. Do one of the following to specify the components you want imported:
   - To import a single package file, either type the full path and name of the `.jar` file that you want to import in the **Integration repository file** field or click **Browse** to select the file.
   - To import an entire directory of package files, either type the full path to the directory in the **Import from a Repository Files directory** field or click **Browse** to select the file.
4. Click **Finish**.

Note: Do not use the **File > Import** menu item in the workbench to import a package file. Although the "Zip file" wizard works with archives with the `.jar` extension, and WebSphere Business Integration adapter package files have a `.jar` extension, the "Zip file" wizard does not work properly with package files.

Working with solutions

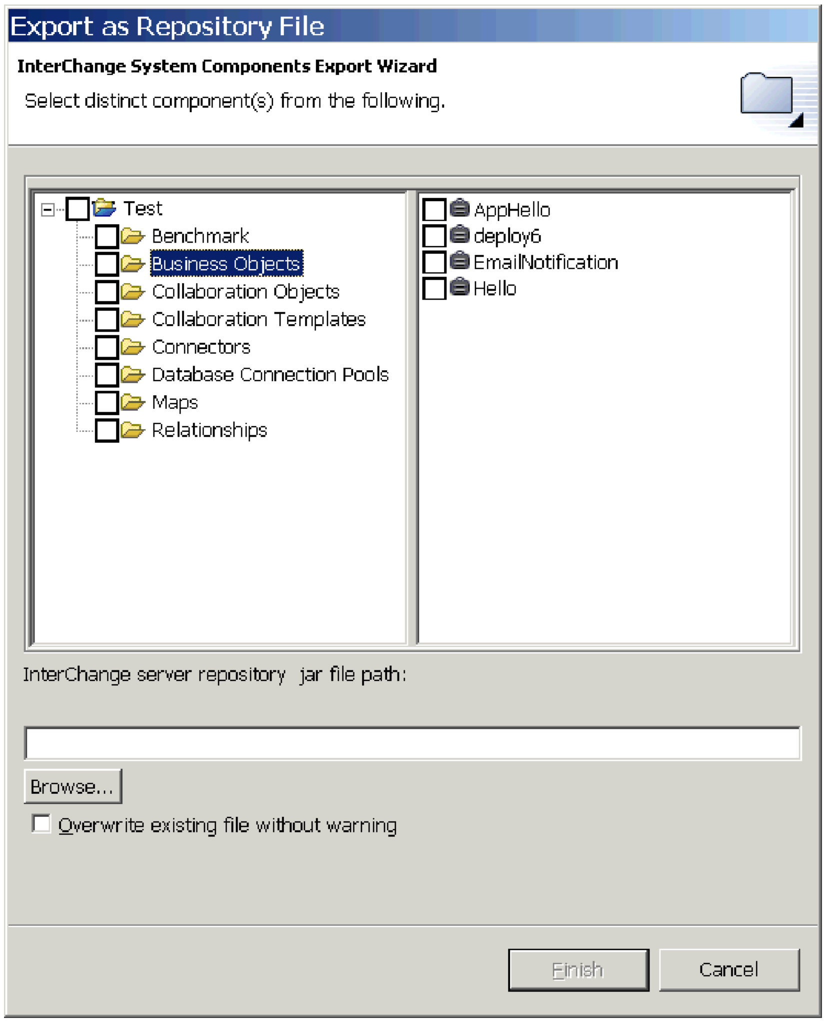
You can export a user project as a solution. This action copies the shortcuts from the user project as well as the component definitions that the shortcuts reference in the integration component libraries. This makes it easy to migrate an entire interface or business integration system from one environment to another.

Exporting a solution

Do the following to export a user project and the integration components it references as a solution:

1. In the WebSphere Business Integration System Manager view, expand the **User Projects** folder, then right-click the **Message Brokers** folder and choose **Export Solution** from the context menu.

   System Manager displays the "Export Solution" wizard, as shown in Figure 33 on page 145.


Export as Repository File

InterChange System Components Export Wizard

Select distinct component(s) from the following.

InterChange server repository jar file path:

☐ Overwrite existing file without warning

Figure 33. Exporting a solution

2. Use the following techniques to select the components you want to export:
   - Enable the checkbox next to user projects to select all of the components in the projects.
   - Enable the checkbox next to a component group to select all of the components in the group.
   - Highlight a component group and then enable checkboxes next to individual components in the right-hand pane to select those components.
3. Either type the full path and name of the directory into which the solution should be exported in the text field at the bottom of the wizard screen, or click **Browse** to navigate to the desired directory.
4. Click **Finish**.

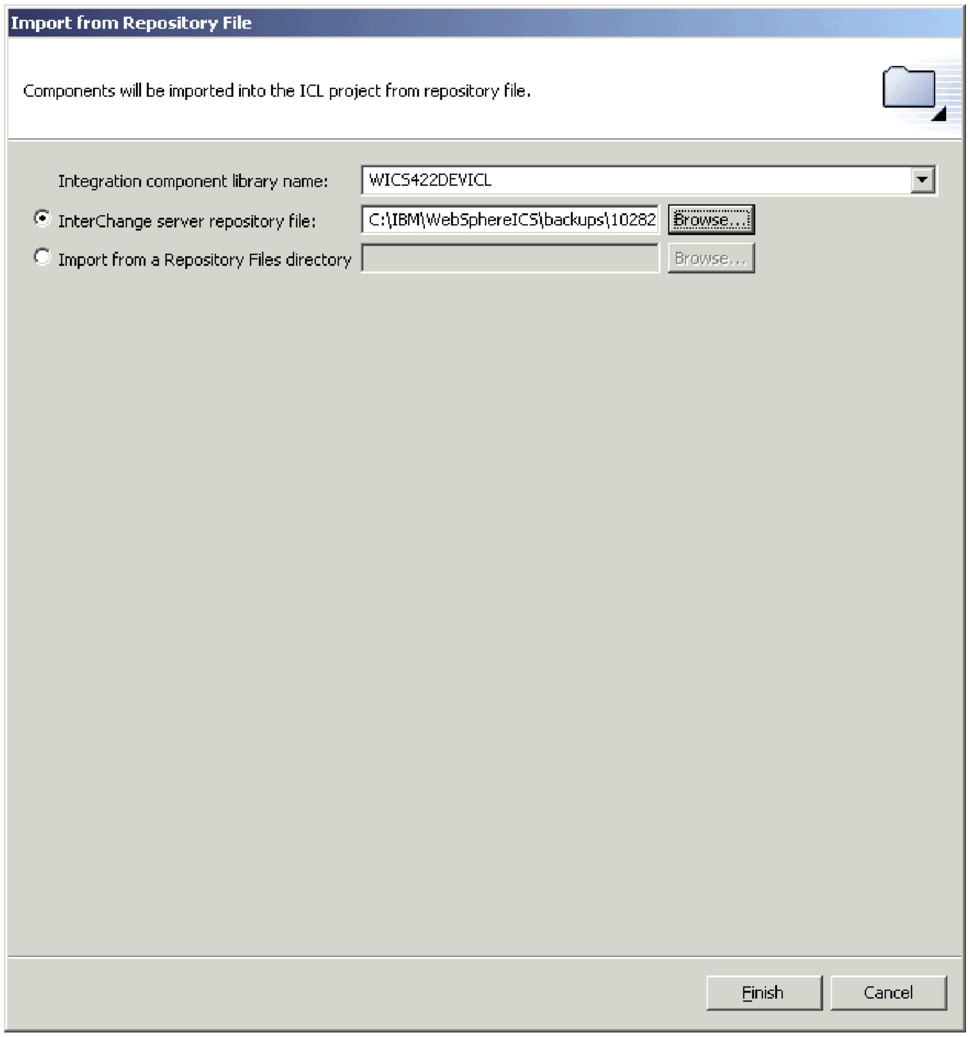
   System Manager does the following to export the solution in the directory specified in step 3:
   - Creates a `User` directory that contains the shortcuts in the user projects selected during the export of the solution.
   - Creates a `System` directory that contains the directories of the integration component library referenced by the shortcuts in the user projects selected during the export of the solution.
5. When prompted that the export operation completed successfully, click **OK**.

Importing a solution

Do the following to import a solution:

1. In the WebSphere Business Integration System Manager view, expand the **User Projects** folder, then right-click the **Message Brokers** and choose **Import Solution** from the context menu.

   System Manager displays the "Import Solution" wizard, as shown in Figure 34.



Import from Repository File

Components will be imported into the ICL project from repository file.

Integration component library name: WICS422DEVICL

[✓ InterChange server repository file:] C:\IBM\WebSphereICS\backups\10282

Import from a Repository Files directory

Figure 34. Importing a solution

2. Either type the full path and name of the directory in which the exported solution exists into the **Solution directory name** field, or click **Browse** to navigate to the desired directory.
3. Click **Finish**.

   System Manager creates the integration component libraries and user projects defined in the exported solution in your environment.

Exporting components to a package using System Manager

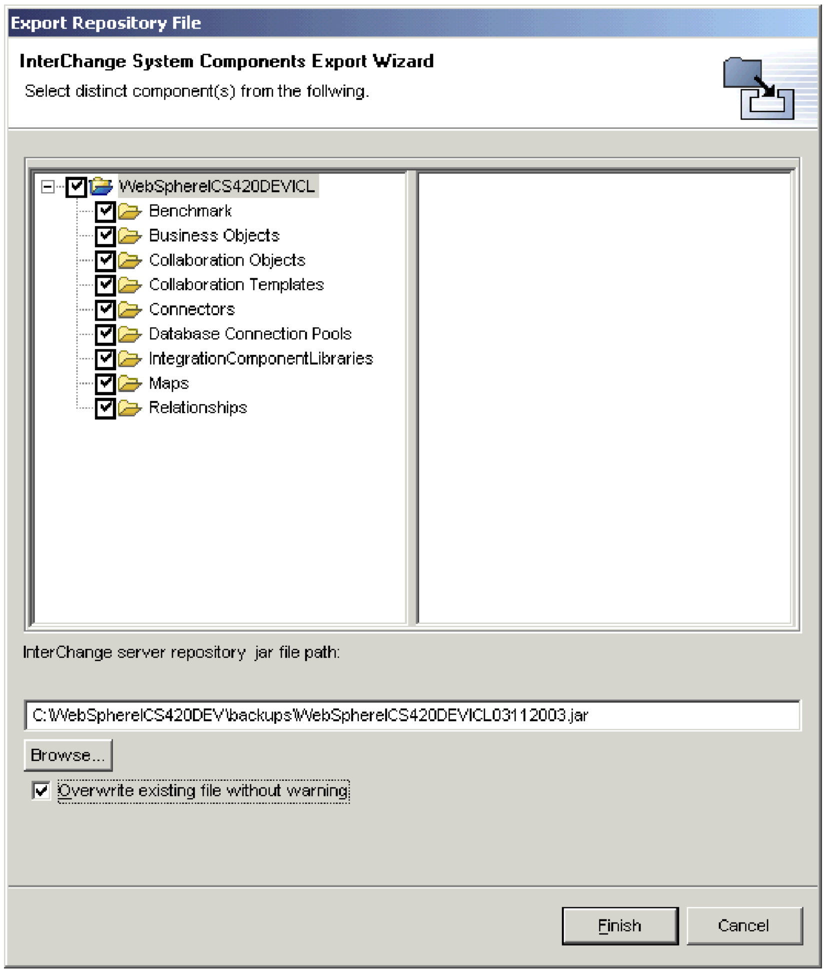
You can export integration components to a package file. Integration components are resources, which are ultimately files stored in the file system as described in “Resources” on page 130. When System Manager exports components to a package, it compresses the following resources into a `.jar` (Java archive) file:

- Definition files (stored in XML format, with different extensions depending on the component type)
- Java source files for maps and collaboration templates
- Message files

Do the following to export components to a package:

1. Right-click either an integration component library or a user project that contains the components you want to export and choose **Export as Repository File** from the context menu.

   System Manager displays the “Export Repository File” wizard, as shown in Figure 35.



Export Repository File
InterChange System Components Export Wizard
Select distinct component(s) from the follwing.
WebSphereICS420DEVICL
Benchmark
Business Objects
Collaboration Objects
Collaboration Templates
Connectors
Database Connection Pools
IntegrationComponentLibraries
Maps
Relationships
InterChange server repository jar file path:
C:\WebSphereICS420DEV\backups\WebSphereICS420DEVICL03112003.jar
Browse...
Overwrite existing file without warning
Finish
Cancel

Figure 35. Exporting a package

2. Use the following techniques to select the components you want to export:
   - Enable the checkbox next to the integration component library or user project to select all of the components in the library or project.

- Enable the checkbox next to a component group to select all of the components in the group.
- Highlight a component group and then enable checkboxes next to individual components in the right-hand pane to select those components.

3. If you plan to specify an existing `.jar` file to export the components to and you want to overwrite it without receiving a prompt, enable the **Overwrite existing file without warning** checkbox.

   Note: To benefit from the **Overwrite existing file without warning** checkbox you must enable it before specifying the file to use as described in step 4. System Manager prompts you to overwrite a file as soon as it detects that an existing file has been specified and does not wait until you have finished the wizard, so you must enable this option beforehand to benefit from it.
4. Type the name and path of the `.jar` file to which the components should be exported into the **Repository jar file path** field, or click **Browse** to select a file to overwrite, or navigate to a directory and specify a file name.

   If you specified the name and path to an existing file and did not enable the **Overwrite existing file without warning** checkbox and you want to overwrite the existing file then click **Yes** when prompted.

   Note: If you type the name and path of the file into the field you must include the `.jar` extension in order for the **Finish** button to be enabled.
5. Click **Finish** to complete the wizard.

Dependencies and references

Integration components depend upon one another to perform their roles in the business integration system. For instance, business object definitions can contain other business object definitions as children and connectors have maps associated with their supported business objects. These dependencies must be satisfied for the system to function properly.

The terms **dependencies** and **references** are used to describe the relationship between components, depending on the context. For example, a connector definition requires the business object definitions it supports so that it can exchange data with the integration broker. In this context, the business object definitions are **dependencies** of the connector definition. If you view the same relationship between the business object definition and the connector definition, but in the context of the business object, the connector is one of the **references** of the business object definition—along with any other business objects that contain it as a child, any maps that transform it, any collaboration templates that support it for their port definitions, and so forth.

Table 26 on page 149 specifies the components that can be dependencies and references for each component type.

Table 26. Integration component dependencies and references

Component	Dependencies	References
Business objects	• Business objects	• Business objects • Maps • Connectors • Collaboration templates • Collaboration objects
Connectors	• Business objects • Maps	• Collaboration objects
Maps	• Business objects • Maps	• Connectors • Collaboration objects
Collaboration templates	• Business objects	None
Collaboration objects	• Business objects • Maps • Connectors • Collaboration templates • Collaboration objects	• Collaboration objects

Showing dependencies and references

You can use System Manager to show the dependencies and references for an integration component.

To show the dependencies for a component, right-click it in System Manager and select **Show Dependencies** from the context menu. The “Dependency Tree” wizard appears. For more information, see “Using the Dependency Tree” on page 139.

To show the references for a component, right-click it in System Manager and select **Show References** from the context menu. The “Object References” window appears.

Standard operations available for multiple workbench resources

Although many of the tasks you perform in the workbench are sensitive to the particular resource you are working with or the context in which you are working, there are many operations that affect all resources the same way. This section describes the tasks you can perform in the workbench that work similarly for all resources.

Adding projects to the workspace from source code control

You can add integration component libraries and user projects to your workspace from a source code control system such as Rational ClearCase.

For more information about how to do so, see the documentation for the source code control system plug-in.

For an example of how to do so using ClearCase, see the *Implementation Guide for WebSphere Interchange Server*.

Cutting, copying, and pasting resources

You can cut, copy, and paste resources both in System Manager and in the file system.

To cut, copy, or paste a user project, integration component library, integration component, shortcut, or folder in System Manager, right-click the resource and choose the desired menu item. If you copy integration components, you cannot paste them into the same library (for instance, to copy a business object definition and paste it into the same library to specify a different name and use it as a template). You can, however, open the component definition in its designer tool and do a "Save as" operation to save it with a new name to the same library.

To cut, copy, and paste integration components or shortcuts in the file system, launch Windows Explorer and navigate to the appropriate subdirectory within the project directory, copy the file that shares the name of the component, and paste it into the appropriate subdirectory within the destination project directory.

If you use cut, copy, and paste operations in the file system, you must refresh the integration component library or user project in System Manager in order to see the newly added resources. For more information, see "Refreshing resources."

You cannot just cut, copy, and paste whole user projects or integration component libraries, because there are meta-data references maintained in the workbench that specify which user projects and libraries exist. Copying a folder into the workspace directory does not update those meta-data references. You can, however, create a new user project or library in System Manager to satisfy the meta-data entries, and then paste the folders of component definitions into the directory for the new library or user project.

Refreshing resources

If you add component definitions to a library or shortcuts to a user project by cutting, copying, and pasting files in the file system then you must refresh the library in System Manager so that the changes are reflected.

To refresh an integration component library or user project, right-click it in System Manager and choose **Refresh View** from the context menu.

Deleting resources

Do the following to delete a workbench resource:

1. Either right-click the resource in System Manager and choose **Delete** from the context menu or select the resource in System Manager and press the **Delete** key.
2. When presented with the "Delete Component" dialog, click **OK**.

Note: If a component has dependencies then you will not be allowed to delete it.

Using Eclipse-based workbenches

This section describes some of the optional tasks you may want to perform in your tooling framework to make your experience with the tools more efficient.

Opening and closing perspectives

This section describes how to open and close perspectives.

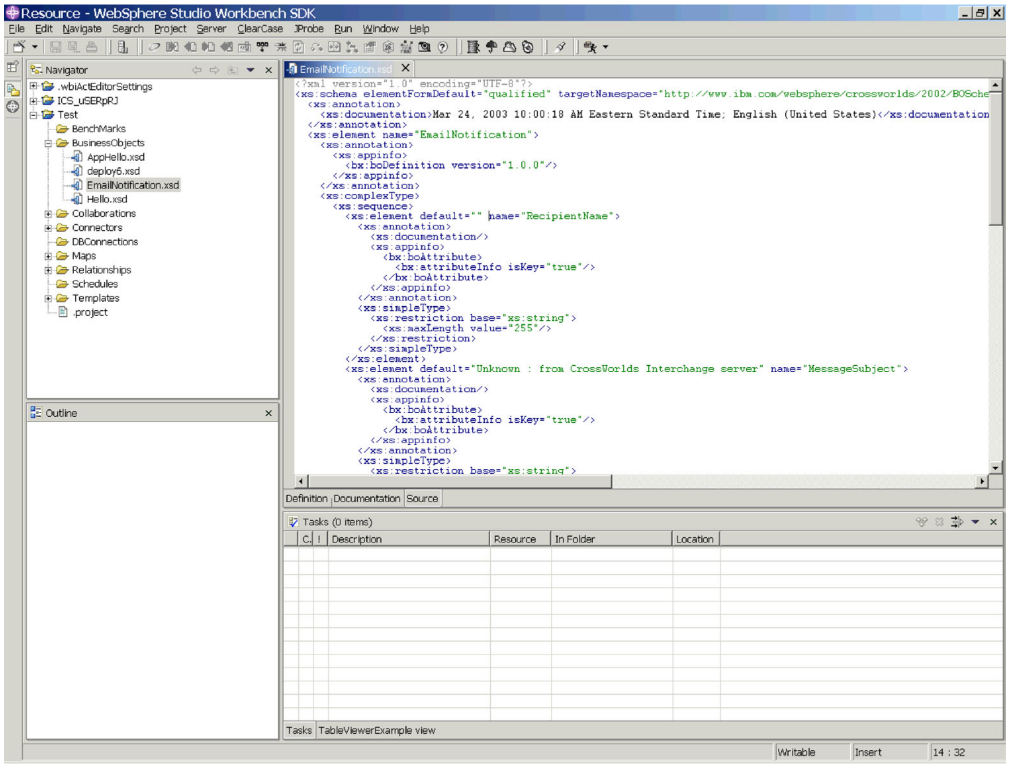
Opening perspectives

Do the following to open a perspective in the workbench:

1. Select **Window > Open Perspective > Other** from the menu bar of your workbench.
2. At the “Select Perspective” dialog, choose the perspective you want to open and click **OK**.

Besides the `System Manager` perspective that you use to work with integration components, you might also want to explore the perspectives described in the following sections:

Resource perspective: This perspective allows you to work directly with the metadata files that represent project materials, such as integration component libraries. Figure 36 shows the `Resource` perspective open where the `.xsd` file that represents the business object has been opened from the `Navigator` view to display its contents in the `Editor` view.

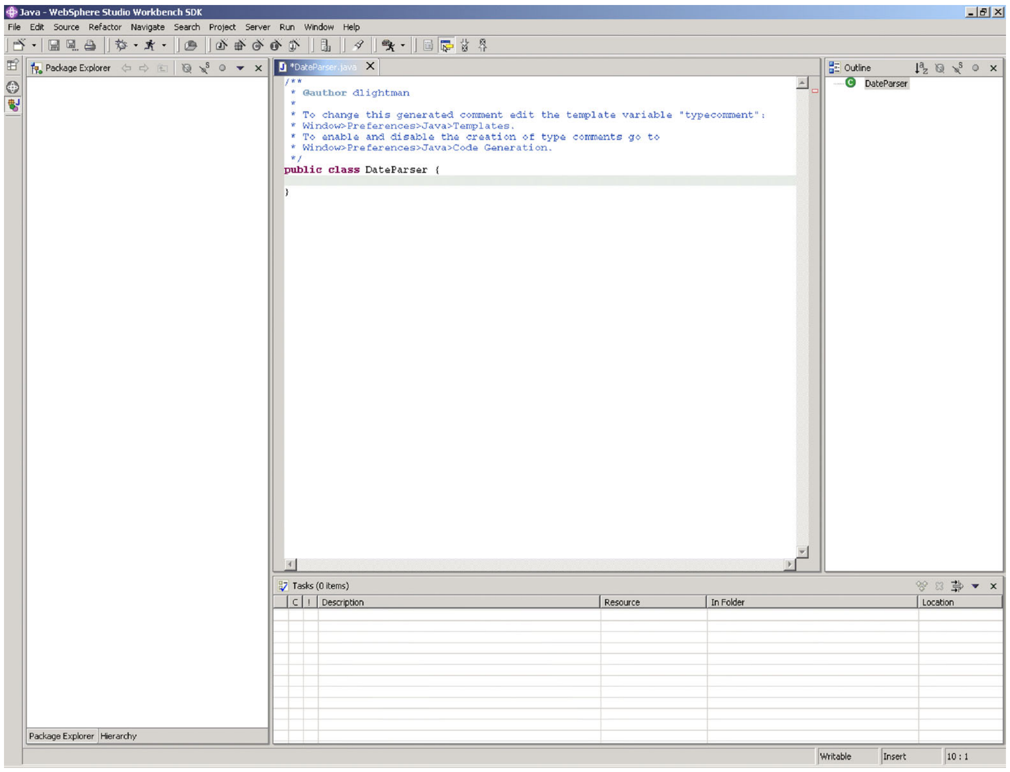


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" targetNamespace="http://www.ibm.com/websphere/crossworlds/2002/BOSche
  <xs:annotation>
    <xs:documentation>Mar 24, 2003 10:00:18 AM Eastern Standard Time; English (United States)</xs:documentation
  </xs:annotation>
  <xs:element name="EmailNotification">
    <xs:annotation>
      <xs:appinfo>
        <bx:boDefinition version="1.0.0"/>
      </xs:appinfo>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element default="" name="RecipientName">
          <xs:annotation>
            <xs:documentation/>
            <xs:appinfo>
              <bx:boAttribute>
                <bx:attributeInfo isKey="true"/>
              </bx:boAttribute>
            </xs:appinfo>
          </xs:annotation>
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="255"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element default="Unknown : from CrossWorlds Interchange server" name="MessageSubject">
          <xs:annotation>
            <xs:documentation/>
            <xs:appinfo>
              <bx:boAttribute>
                <bx:attributeInfo isKey="true"/>
              </bx:boAttribute>
            </xs:appinfo>
          </xs:annotation>
          <xs:simpleType>
            <xs:restriction base="xs:string">
```

Figure 36. Resource perspective

Important: The metadata files exposed through the `Resource` perspective define the WebSphere Business Integration components. The System Manager perspective provides a way to work with those components safely through interfaces. If you manipulate metadata files directly you run the risk of ruining the component definition. It is recommended that you only work with component metadata files if you understand their structures very well, or in situations where you are interacting with Technical Support to troubleshoot a definition and are asked to.

Java Perspective: This perspective provides editors and views that assist with authoring Java files. Although you perform most Java programming in the designer tools, you occasionally have to write external components such as data handlers, or write utility classes. In these cases, this Java perspective can be very useful. The following figure shows the Java perspective:


Java - WebSphere Studio Workbench SDK
File Edit Source Refactor Navigate Search Project Server Run Window Help
Package Explorer
*DateParser.java
Outline
DateParser
Tasks (0 items)
C ! Description Resource In Folder Location
Package Explorer Hierarchy
Writable Insert 10 : 1

```
/**
 * @author dlightman
 *
 * To change this generated comment edit the template variable "typecomment":
 * Window>Preferences>Java>Templates.
 * To enable and disable the creation of type comments go to
 * Window>Preferences>Java>Code Generation.
 */
public class DateParser {

}
```

Figure 37. Java perspective

Closing perspectives

You can do the following to close perspectives:

- Select **Window > Close Perspective** to close the currently active perspective.
- Select **Window > Close All Perspectives** to close all currently open perspectives
- Right-click the icon for a perspective in the perspective shortcut bar and choose **Close** from the context menu to close that perspective.
- Right-click the icon for a perspective in the perspective shortcut bar and choose **Close All** from the context menu to close all open perspectives.

Showing and closing views

You can control the panes that are displayed in the WebSphere WorkBench and WebSphere Studio Application Developer Integration Edition perspectives.

Showing views

Do the following to show a view:

1. Select **Window > Show View > Other**.
2. Expand the folder for the view group, such as **WebSphere Business Integration Adapter Monitor Category**.
3. Select the particular view, such as **Fault Queue Manager**.
4. Click **OK**.

Closing views

To close a view, do one of the following:

- Right-click the title bar of the view and choose **Close** from the context menu.
- Click the close button in the title bar of the view.

Customizing perspectives

You can customize a perspective to include the perspectives, views, wizards, and plug-in interfaces that you want so that you can minimize the number of times you have to open elements you use frequently and close elements you do not need. Do the following to customize the currently active perspective:

1. Select **Window > Customize Perspective** from the menu bar.
2. Click on the node you want to customize to expand it.
3. Enable and disable checkboxes for the node elements. Table 27 lists the customizable perspective nodes and the effect that enabling them has.

Table 27. Customizable perspective nodes

Customize perspective node	Result
File > New	Adds or subtracts items from the **File > New** menu.
Window > Open Perspective	Adds or subtracts perspectives from the **Window > Open Perspective** menu.
Window > Show View	Adds or subtracts views from the **Perspective > Show View** menu.
Other	Adds or subtracts from the menu bar and toolbar. For instance, you must enable the **ClearCase** checkbox to display the **ClearCase** menu.

Saving perspectives

You can save a perspective configuration to preserve customizations you have made. Do the following to save a perspective:

1. Select **Window > Save Perspective As** from the tooling framework menu bar.
2. Type a name for the perspective in the **Name** field.
3. Click **OK**.

Setting the default perspective

By default your tooling framework opens to the `Resource` perspective. If you primarily use the tooling framework to work with integration components, you may want to make System Manager the default perspective. Do the following to do so:

1. Select **Window > Preferences** from the menu bar of the tooling framework.
2. Expand the **Workbench** node.
3. Select the **Perspectives** node under the **Workbench** node.
4. Select the `System Manager`from the **Available Perspectives** list.
5. Click **Make Default**.
6. Click **OK**.

Configuring System Manager preferences

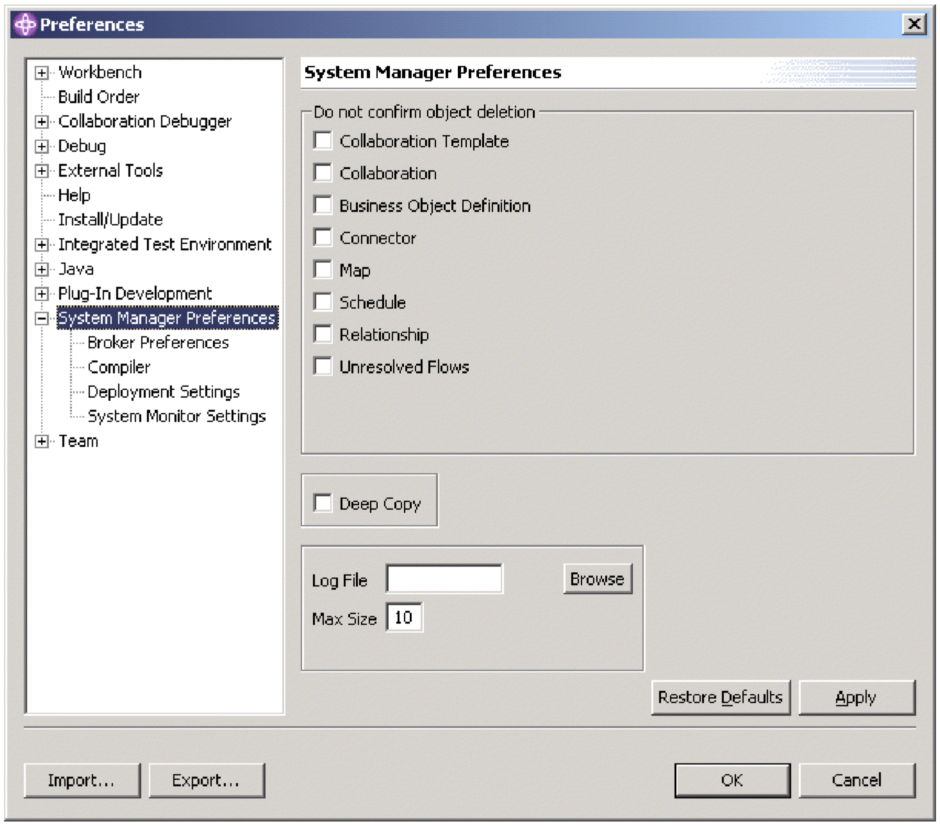
To set your System Manager preferences, do the following:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Select **System Manager Preferences** and do the following to configure the available preference options:
   - If you do not want to be prompted when you delete a component from an integration component library, enable the checkbox for the component type in the "Do not confirm object deletion" pane.
   - Enable the **Deep Copy** checkbox if you want to copy a component's dependencies along with the component when you perform a copy operation on it.

     If **Deep Copy** is enabled and you copy a business object definition from one library to another, then all of the child business objects it contains are copied as well, for example. If **Deep Copy** is not enabled, however, and you copy a business object definition from one library to another, then only the business object itself is copied.

     For more information on dependencies, see "Dependencies and references" on page 148.
   - Type the name and path of a file in the **Log File** field, or click **Browse** to select a file. When errors occur in System Manager, the error information is written to the file specified. Type a number in the **Max Size** field to specify the maximum size of the log file in megabytes.
   - Click **Restore Defaults** to set the preferences elements to their default values.

   Figure 38 on page 155 shows the System Manager preferences interface.



System Manager Preferences

Do not confirm object deletion

- [ ] Collaboration Template
- [ ] Collaboration
- [ ] Business Object Definition
- [ ] Connector
- [ ] Map
- [ ] Schedule
- [ ] Relationship
- [ ] Unresolved Flows

- [ ] Deep Copy

Log File

Max Size 10

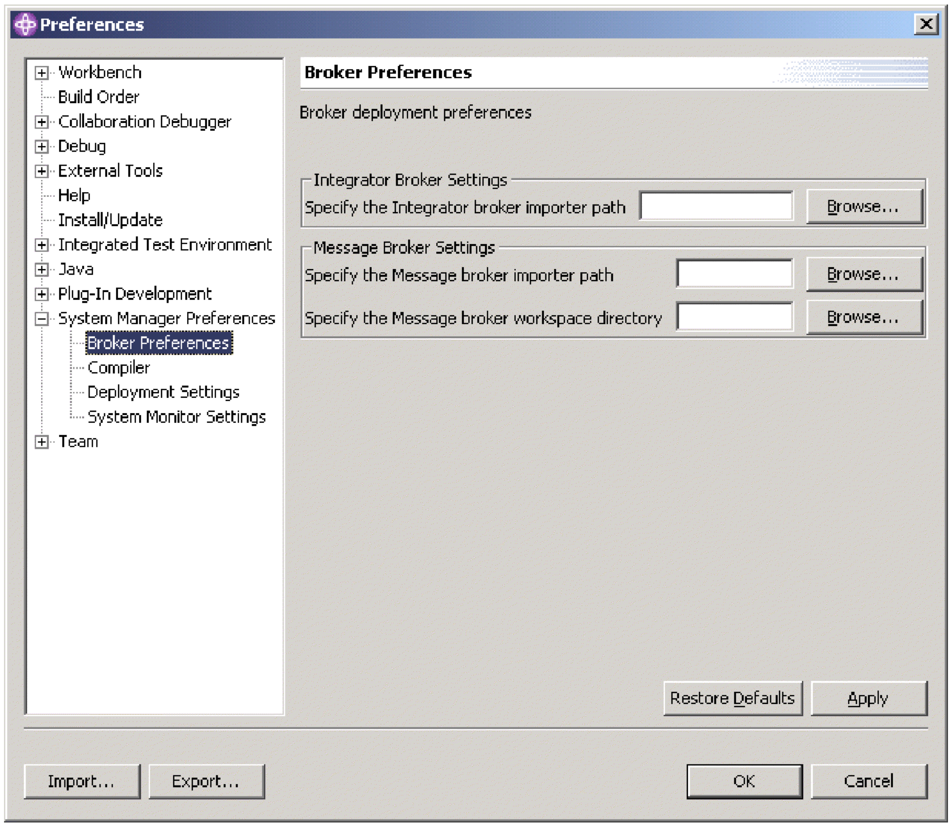
Figure 38. System Manager preferences

3. The “Broker Preferences” interface allows you to configure System Manager to work with the supported WebSphere message brokers.

   Do the following to set your broker preferences:

   a. Expand **System Manager Preferences** and then select **Broker Preferences**.
   b. Either type the full path to the WebSphere MQ Integrator Broker importer utility in the **Specify the Integrator broker importer path** field or click **Browse** to select the directory.
   c. Either type the full path to the WebSphere Business Integration Message Broker importer utility in the **Specify the Message broker importer path** field or click **Browse** to select the directory.
   d. Either type the full path to the Eclipse workspace directory for WebSphere Business Integration Message Broker in the **Specify the Message broker workspace directory** field or click **Browse** to select the directory.

   Figure 39 on page 156 shows the “Broker Preferences” interface:



Broker Preferences

Broker deployment preferences

Integrator Broker Settings

Specify the Integrator broker importer path

Message Broker Settings

Specify the Message broker importer path

Specify the Message broker workspace directory

Figure 39. Broker preferences

Note: The “Compiler” preferences interface, the “Deployment Settings” interface, and the “System Monitor Settings” interfaces are for use when using WebSphere Interchange Server as the integration broker. They are not relevant for other integration brokers.

4. Click **Apply** to save your preferences and continue working in the “Preferences” dialog, or click **OK** to save your preferences and exit the dialog.

Troubleshooting problems connecting to the integration broker in System Manager

Investigate these possibilities to troubleshoot connection problems:

- Make sure that the importer path is specified.
- Make sure that the System Manager is deployed in the correct workspace.

Appendix F. Using the Connector Script Generator tool

The Connector Script Generator utility creates or modifies the connector script for connectors running on the UNIX platform. Use this tool to do either of the following:

- To generate a new connector startup script for a connector you have added without using the WebSphere Business Integration Adapters installer.
- To modify an existing startup script for a connector to include the correct configuration file path.

To run the Connector Script Generator, do the following:

1. Navigate to the `ProductDir/bin` directory.
2. Enter the command `./ConnConfig.sh.`

   The Connector Script Generator screen appears as shown in Figure 40.

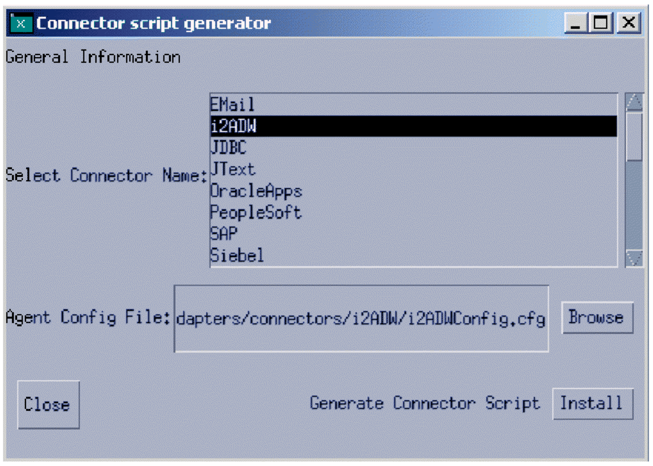


Figure 40. Connector Script Generator.

3. From the Select Connector Name list, select the connector for which the startup script is to be generated.
4. For Agent Config File, specify the connector's configuration file by entering its full-path name or by clicking **Browse** to select a file.
5. To generate or update the connector script, click **Install**.

   The `connector_manager_ConnectorName` file (where `ConnectorName` is the name of the connector you are configuring) is created in the `ProductDir/bin` directory.
6. Click **Close**.

Appendix G. Using Visual Test Connector

Visual Test Connector simulates the activities of a connector to allow you to test your integration components without the complexity of running an actual connector. This chapter consists of the following sections:

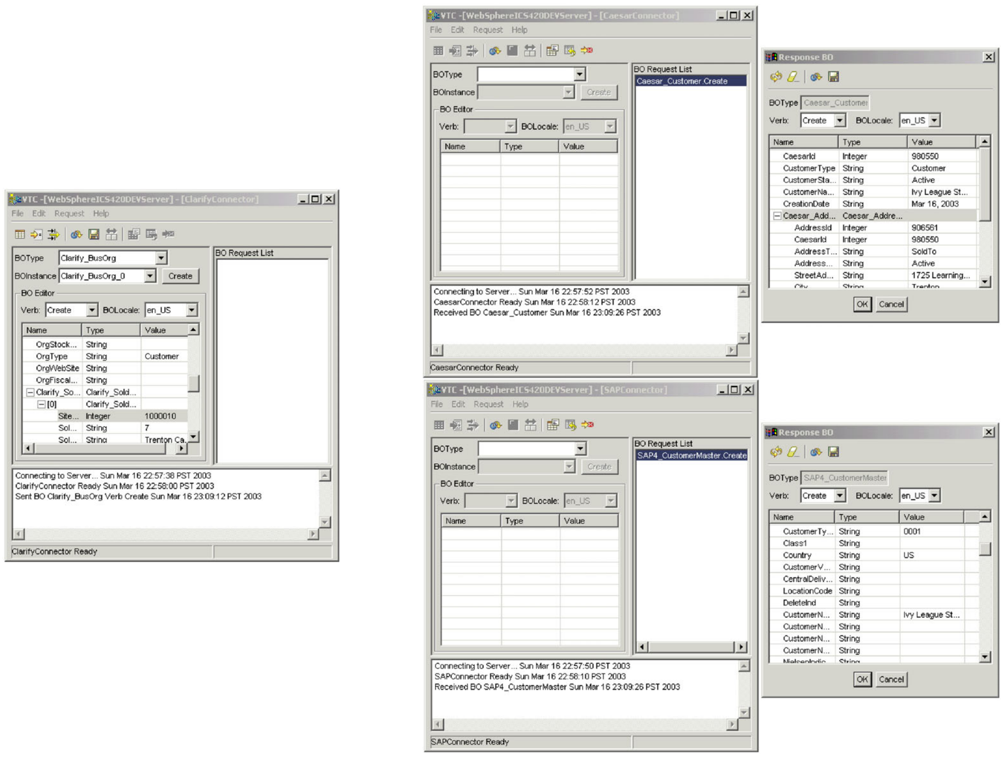
- "Recommended testing procedure"
- "Starting Test Connector" on page 160
- "Shutting down Test Connector" on page 161
- "Creating and editing connector profiles" on page 161
- "Emulating a connector" on page 164
- "Working with business objects" on page 164

Recommended testing procedure

This is the recommended test procedure for testing components in the WebSphere business integration system:

1. If your integration broker is InterChange Server, consider using the System View view, which can be very helpful in determining if a flow you have sent ends in success or failure.

   For more information, see the *System Administration Guide*.
2. Set up Test Connector to emulate a source connector.
   a. Launch Test Connector as described in "Starting Test Connector" on page 160.
   b. Create a profile for the source connector in the interface as described in "Creating a new profile" on page 162.
   c. Connect Test Connector to the agent to begin emulating the source connector, as described in "Emulating a connector" on page 164.
3. Set up instances of Test Connector to emulate each destination connector involved in the interface.
   a. Launch Test Connector as described in "Starting Test Connector" on page 160.
   b. Create a profile for a destination connector as described in "Creating a new profile" on page 162.
   c. Connect Test Connector to the agent to begin emulating the destination connector as described in "Emulating a connector" on page 164.
   d. Repeat 3athrough 3cabove for all destination connectors involved in the interface.
4. Arrange the instances of Test Connector on your screen so that you can easily identify the connector being emulated in each Test Connector window. For example, in Figure 41 on page 160the source Test Connector is arranged to the left of the destination Test Connector.


VTC -[WebSphereICS420DEVServer] - [ClarifyConnector]
File Edit Request Help
Create
VTC -[WebSphereICS420DEVServer] - [CaesarConnector]
VTC -[WebSphereICS420DEVServer] - [SAPConnector]
Response BO
OK
Cancel

ClarifyConnector:

BOType: Clarify_BusOrg
BOInstance: Clarify_BusOrg_0
BO Editor — Verb: Create BOLocale: en_US

Name	Type	Value
OrgStock...	String	
OrgType	String	Customer
OrgWebSite	String	
OrgFiscal...	String	
Clarify_So...	Clarify_Sold...	
[0]	Clarify_Sold...	
Site...	Integer	1000010
Sol...	String	7
Sol...	String	Trenton Ca...

BO Request List

```
Connecting to Server... Sun Mar 16 22:57:38 PST 2003
ClarifyConnector Ready Sun Mar 16 22:58:00 PST 2003
Sent BO Clarify_BusOrg Verb Create Sun Mar 16 23:09:12 PST 2003
```

ClarifyConnector Ready

CaesarConnector:

BO Request List: Caesar_Customer.Create

```
Connecting to Server... Sun Mar 16 22:57:52 PST 2003
CaesarConnector Ready Sun Mar 16 22:58:12 PST 2003
Received BO Caesar_Customer Sun Mar 16 23:09:26 PST 2003
```

CaesarConnector Ready

Response BO — BOType: Caesar_Customer Verb: Create BOLocale: en_US

Name	Type	Value
CaesarId	Integer	980550
CustomerType	String	Customer
CustomerSta...	String	Active
CustomerNa...	String	Ivy League St...
CreationDate	String	Mar 16, 2003
Caesar_Add...	Caesar_Addre...	
AddressId	Integer	906561
CaesarId	Integer	980550
AddressT...	String	SoldTo
Address...	String	Active
StreetAd...	String	1725 Learning...

SAPConnector:

BO Request List: SAP4_CustomerMaster.Create

```
Connecting to Server... Sun Mar 16 22:57:50 PST 2003
SAPConnector Ready Sun Mar 16 22:58:10 PST 2003
Received BO SAP4_CustomerMaster Sun Mar 16 23:09:26 PST 2003
```

SAPConnector Ready

Response BO — BOType: SAP4_CustomerMaster Verb: Create BOLocale: en_US

Name	Type	Value
CustomerTy...	String	0001
Class1	String	
Country	String	US
CustomerV...	String	
CentralDeliv...	String	
LocationCode	String	
DeleteInd	String	
CustomerN...	String	Ivy League St...
CustomerN...	String	
CustomerN...	String	
CustomerN...	String	

Figure 41. Source and destination instances of Test Connector.

5. Send a request business object from the source connector. From the source Test Connector, do the following:
   a. Create a business object that is managed by the interface you need to test as described in "Creating request business objects" on page 164.
   b. Save the business object to a file to use in subsequent tests as described in "Saving a business object" on page 168.
   c. Send the business object as described in "Sending request business objects" on page 165.
6. Simulate the response to the request business object from the destination connector. From the destination Test Connector window, do the following:
   a. Accept the request business object as described in "Accepting a request business object" on page 169.
   b. Send the business object as a response as described in "Sending a response business object" on page 170.
7. Repeat step 5through step 6as many times as necessary to test each interface.

Starting Test Connector

To start Test Connector, do one of the following depending on your integration broker:

- If your integration broker is InterChange Server, select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Development > Test Connector**.
- If your integration broker is WebSphere Application Server or a WebSphere message broker (WebSphere MQ Integrator, WebSphere MQ IntegratorBroker, or WebSphere Business Integration Message Broker) select **Start > Programs > IBM WebSphere Business Integration Adapters > Tools > Test Connector**.

Figure 42 shows Test Connector after starting.

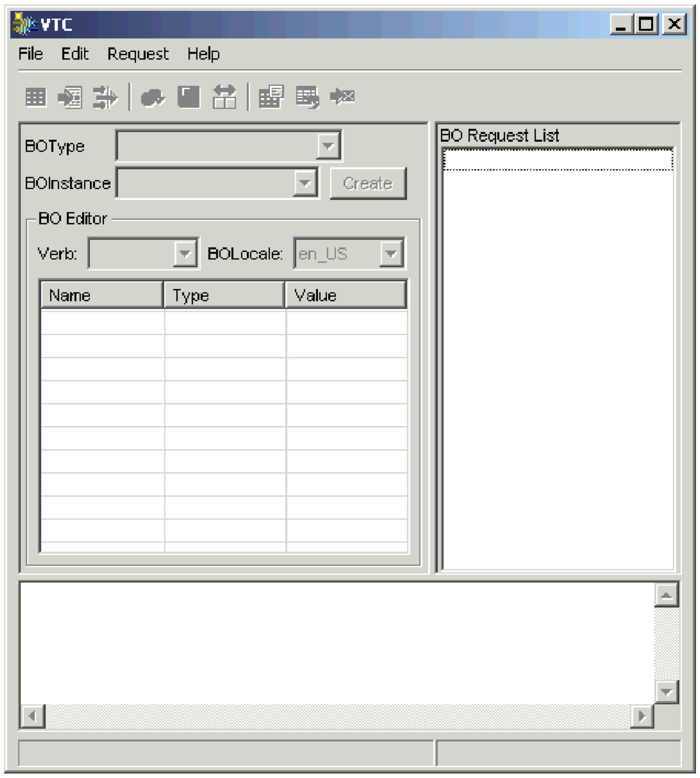


Figure 42. Test Connector

The Test Connector window includes the following panes:

- The "Supported Business Objects" pane in which you can create business object instances to send
- The "BO Request List" pane, which displays any business object requests that the connector has received
- The "Output" pane, which displays messages about Test Connector's operations, such as when a business object has been sent.

Shutting down Test Connector

To shut down Test Connector and cause it to stop emulating a connector agent, select **File > Exit** from the menu bar. When presented with the "Shutdown" prompt, click **Yes**.

Creating and editing connector profiles

Test Connector uses profiles to store the information it needs to emulate a connector. You must create a profile for each connector you want to emulate. You can edit and delete existing profiles.

Saving the connector definition to a file

To emulate a connector using Test Connector, you must save the connector definition to a file. Do the following to save a connector definition to a file:

1. Open the connector definition in Connector Configurator.
2. Select **File > Save As > To File** from the menu bar.

3. Navigate to the directory in which you want the file saved, type a name in the **File name** field, ensure that the value `Configuration (*.cfg)` is displayed in the Save as type drop-down menu, and click **Save**.

   Connector Configurator saves the connector definition to a file with the specified name.

Creating a new profile

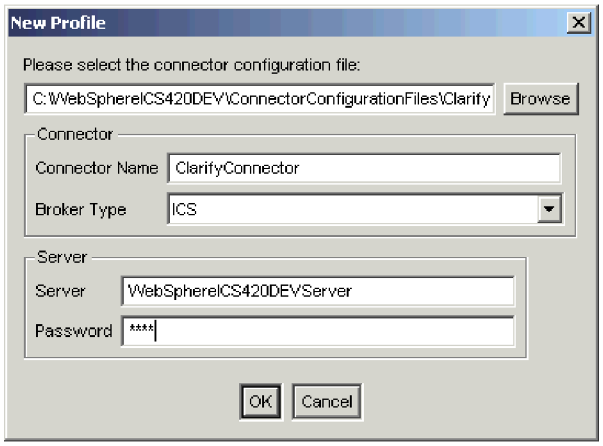
You must create a profile for any connector you want to emulate in Test Connector. The profile specifies information such as the name of the connector, the configuration file to be used, and the type of integration broker with which the connector communicates. To create a new connector profile, do the following:

1. Select **File** > **Create/Select Profile** from the menu bar to display the “Connector Profile” window.
2. In the “Connector Profile window”, select **File** > **New Profile** from the menu bar.
3. In the “New Profile” window, click **Browse** and then navigate to the configuration file for the connector you preparing in “Saving the connector definition to a file” on page 161.
4. Type the name of the connector in the **Connector Name** field. You must type the exact name of the connector definition as it exists in the integration broker repository. For the adapter for JText, for instance, you must type `JTextConnector`, without any spaces between the words `JText` and `Connector`, and with each letter being the proper case.
5. Select the proper integration broker in the **Broker Type** drop-down menu—`ICS`, `WMQI` or `WAS`.

   Note: Select `WMQI` if your broker is any WebSphere message broker.
6. If you selected `ICS` as your broker type in step 5,do the following as well:
   a. Type the name of the InterChange Server instance in the **Server** field.

      Be sure to type the name precisely; it is case-sensitive and Test Connector will not be able to communicate with InterChange Server if the name is not correct.
   b. Type the password for the `admin` user account in the **Password** field. The default password is `null`.

      Figure 43 shows the “New Profile” window:



Please select the connector configuration file: C:\WebSphereICS420DEV\ConnectorConfigurationFiles\Clarify

Connector

Connector Name: ClarifyConnector

Broker Type: ICS

Server

Server: WebSphereICS420DEVServer

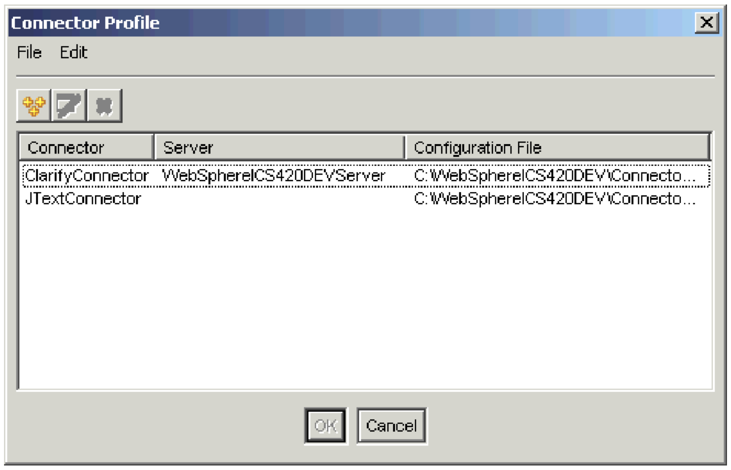
Password: ****

Figure 43. Creating a new connector profile

7. Click **OK** to close the “New Profile” window.

   The “Connector Profile” window displays the name of the connector in the **Connector** column, the name of the InterChange Server instance in the **Server** column (if the integration broker is ICS), and the path and name of the connector configuration file in the **Configuration File** column.

   Figure 44 shows the “Connector Profile” window with a profile for the `ClarifyConnector` configured to communicate with an InterChange Server instance, and a profile for the `JTextConnector` configured to communicate with a WMQIB server.



Connector	Server	Configuration File
ClarifyConnector	WebSphereICS420DEVServer	C:\WebSphereICS420DEV\Connecto...
JTextConnector		C:\WebSphereICS420DEV\Connecto...

Figure 44. The “Connector Profile” window

8. Click **OK** to close the “Connector Profile” window.

Editing a profile

Follow the steps below to make changes to an existing connector profile:

1. Select **File > Create/Select Profile** from the menu bar of Test Connector or use the keyboard shortcut **Ctrl+N** to display the Connector Profile window.
2. In the “Connector Profile” window select the profile you want to edit and then select **Edit > Edit Profile** from the menu bar.
3. Type new values in the fields of the “New Profile” window and use the **Browse** button to change the configuration file as necessary to make your edits.
4. Click **OK** to close the “New Profile” window.

Deleting a profile

Do the following to delete a connector profile:

1. Select **File > Create/Select Profile** from the menu bar of Test Connector or use the keyboard shortcut **Ctrl+N** to display the “Connector Profile” window.
2. In the “Connector Profile” window, select the profile you want to delete and then select **Edit > Delete Profile** from the menu bar.

Emulating a connector

After creating a profile for a connector, you may use that profile to connect Test Connector to the agent. Once you connect Test Connector to the agent, Test Connector begins emulating the connector defined in the selected profile.

To connect Test Connector to the agent, do the following:

1. Select **File > Create/Select Profile** from the menu bar of Test Connector.
2. In the “Connector Profile” window, select the name of the connector whose profile you want to open.
3. Click **OK**.
4. Select **File > Connect** from the menu bar.

   Test Connector displays messages in the “Output” pane as it attempts to emulate the connector. When it finishes connecting, it displays a message indicating that it is “ready” in the “Output” pane and populates the **BOType** list in the “Supported Business Objects” pane.

Working with business objects

To test whether a business process interface has been developed correctly, you need to verify that business objects can be successfully exchanged and processed. This section describes how to:

- Create, modify, delete, and save business object test data
- Compare the attribute values of business objects to easily and quickly view changes made during processing
- Send and receive business objects

Working with request business objects

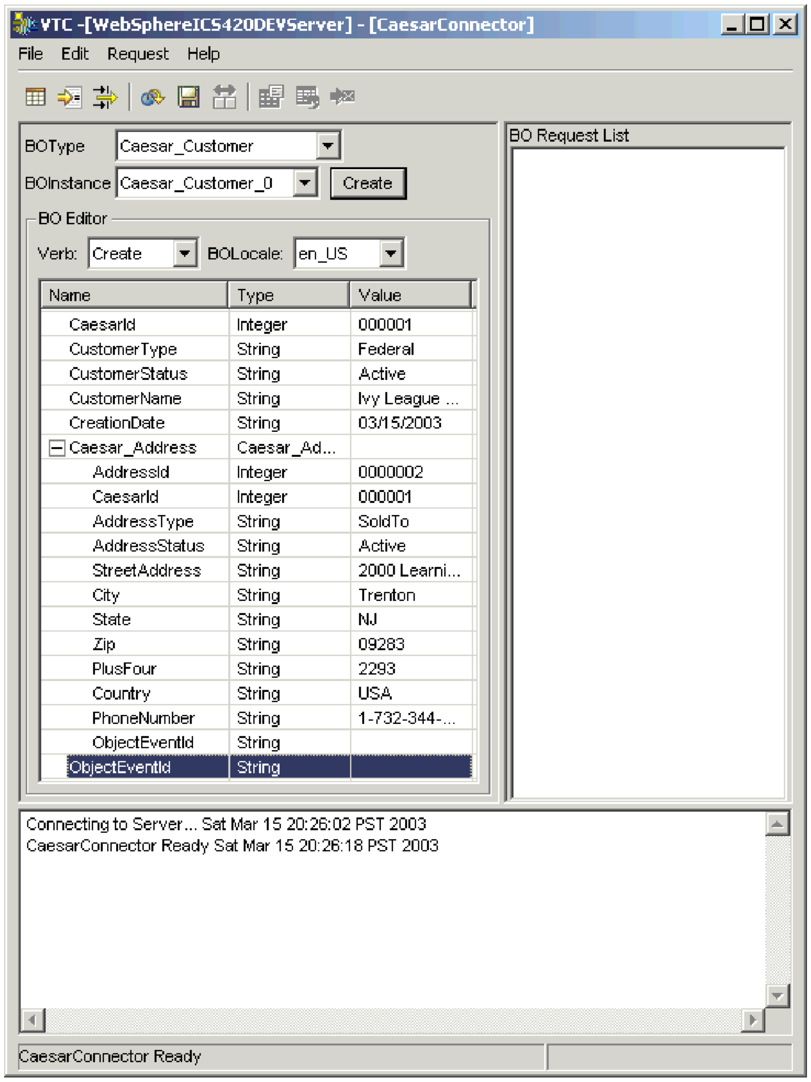
Request business objects are those that you send from Test Connector when it is emulating a connector that is the source of the events that trigger an interface. Working with request business objects consists of creating a business object instance, populating it with data, and sending the request.

Creating request business objects

To create a new business object in Test Connector, do the following:

1. In the “Supported Business Objects” pane, select the name of the business object you want to create from the **BOType** drop-down menu.
2. Click **Create** next to the **BOInstance** field.
3. When presented with the “New Instance” dialog, type a name for the instance in the **Enter Name** field.
4. Select the desired verb from the **Verb** drop-down menu.
5. Select the desired locale from the the **BOLocale** drop-down menu.
6. Provide values for the simple attributes and child business objects within the top-level object, as described in “Setting values for business object attributes” on page 167.

   Figure 45 on page 165 shows a business object named `Caesar_Customer` with the `Create` verb, the `en_US` locale, values specified for each of its simple attributes, and a single instance of the `Caesar_Address` child business object.



BOType: Caesar_Customer
BOInstance: Caesar_Customer_0

BO Editor

Verb: Create BOLocale: en_US

Name	Type	Value
CaesarId	Integer	000001
CustomerType	String	Federal
CustomerStatus	String	Active
CustomerName	String	Ivy League ...
CreationDate	String	03/15/2003
Caesar_Address	Caesar_Ad...	
AddressId	Integer	0000002
CaesarId	Integer	000001
AddressType	String	SoldTo
AddressStatus	String	Active
StreetAddress	String	2000 Learni...
City	String	Trenton
State	String	NJ
Zip	String	09283
PlusFour	String	2293
Country	String	USA
PhoneNumber	String	1-732-344-...
ObjectEventId	String	
ObjectEventId	String	

```
Connecting to Server... Sat Mar 15 20:26:02 PST 2003
CaesarConnector Ready Sat Mar 15 20:26:18 PST 2003
```

CaesarConnector Ready

Figure 45. Populating a business object with data

7. Click **OK**.

Sending request business objects

Once you have created or loaded a business object and specified values for its attributes, you have several ways to send the business object as a request to the integration broker.

Sending request business objects asynchronously: When a source connector sends a request business object in asynchronous mode, it does not expect to get back a response business object. Once the request business object is dispatched, the source connector's role in the transaction is finished. The response business object is typically processed by the integration broker. The default mode for Test Connector is asynchronous.

To send a business object asynchronously, do the following:

1. Select **Request > Mode >Asynchronous** from the menu bar.

   Note: Test Connector operates in "Asynchronous" mode by default, so you only have to perform this step if you previously were sending synchronous requests from the connector. Furthermore, you do not have to set the mode before sending each request.

2. Select **Request > Send** from the menu bar.

If the broker specified in the connector definition is InterChange Server then the business object request is sent to the server for processing.

If the broker specified in the connector definition is one of the supported message brokers or WebSphere Application Server then the business object is placed on the queue specified in the `RequestQueue` standard property.

Sending request business objects synchronously: When a source connector sends a request business object synchronously, it expects to get back a response business object from the integration broker after any destination applications have processed the request. In synchronous mode, Test Connector puts the response business object on the queue specified by the source connector's Synchronous Request Queue property. The default mode for Test Connector is asynchronous.

1. Set Test Connector to synchronous mode by selecting **Request > Mode > Synchronous** from the menu bar.
2. Select **Request > Send** from the menu bar.
3. If the broker specified in the connector definition is InterChange Server then the "Select Collaboration" dialog is displayed. Select the collaboration to which the business object should be sent from the **Collaboration** drop-down menu and click **OK**.

If the broker specified in the connector definition is InterChange Server then the business object request is sent to the configured port of the collaboration object chosen for processing.

If the broker specified in the connector definition is one of the supported message brokers or WebSphere Application Server then the business object is placed on the queue specified in the `SynchronousRequestQueue` standard property.

Sending request business objects in batch mode: In batch mode, Test Connector lets you specify the number of instances of a particular business object you want to send, as well as one attribute in the top-level object —a primary key attribute, for example—that you want set to a unique value for each instance. Test Connector copies the business object as many times as you have specified, incrementing the value of the single attribute you specified, and sends each business object. This option allows you to create a large number of business objects quickly and easily.

If the selected attribute is a key field that participates in dynamic cross-referencing as part of an identity relationship, then you must guarantee that the initial value and all those that follow it are unique. Otherwise, the cross-referencing logic will fail, causing the request business objects to fail.

To ensure that the values are unique, you can use Relationship Manager or execute SQL statements against the table for the relationship participant as follows.

- Determine the highest current value for the participant and set the Initial Value field to an even higher value. The first business object instance in the batch and all those that follow will then be unique.
- Delete the existing table entries for the participant, thus guaranteeing that no entries have the same attribute value as any of the batch business objects.

To send business objects in batch mode, do the following:

1. Select the name of the business object you would like to send from the **BOType** drop-down menu.
2. Select **Request > Send Batch** from the menu bar.
3. In the “Batch Mode” window, select the desired verb from the **Verb** drop-down menu.
4. Select the desired locale from the the **BOLocale** drop-down menu.
5. Select from the **Attribute** list the attribute in the top-level business object that you want incremented with each business object request in the batch.

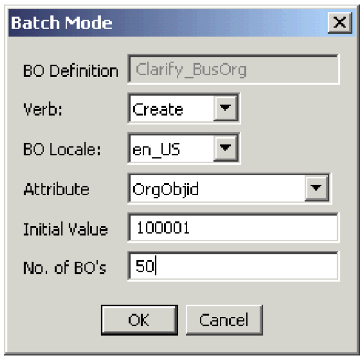
   The selected attribute should typically be an attribute that uniquely identifies the business object, such as a primary key.
6. In the **Initial Value** field, type the starting value for the attribute to be incremented.
7. In the **No. of BO's** field, type the number of business object instances you want generated and sent.
8. Click **OK**.

   Test Connector generates the number of business objects you specified, all identical with the exception of the one specified attribute, whose value is incremented for each instance.

   If the broker specified in the connector definition is InterChange Server then the business object request is sent to the server for processing.

   If the broker specified in the connector definition is one of the supported message brokers or WebSphere Application Server then the business object is placed on the queue specified in the `RequestQueue` standard property.

Figure 46 shows a batch mode configuration in which:
- Fifty business objects are to be sent.
- The value of the attribute `OrgObjid` is to be incremented.
- The starting value for the attribute is 100001.



BO Definition	Clarify_BusOrg
Verb:	Create
BO Locale:	en_US
Attribute	OrgObjid
Initial Value	100001
No. of BO's	50

Figure 46. The Batch Mode Window

Setting values for business object attributes

The following sections describe the various ways you can set the values of simple and compound attributes in a business object instance:
- “Setting values for simple attributes” on page 168
- “Adding child business objects” on page 168
- “Removing child business objects” on page 168
- “Setting the verb of a child business object” on page 168

Setting values for simple attributes

To provide a value for a simple attribute, click its cell in the **Value** column and enter a value.

Adding child business objects

To add an instance of a child business object, right-click the attribute that represents the child object and select **Add Instance** from the context menu.

A plus sign (+) is added next to the attribute that represents child business object to show that there is at least one child business object instance. If you expand the child object attribute, numbered entries are displayed for each instance. The individual instances also have plus signs (+) next to them, so you can expand them and set values for their attributes.

To add more child business object instances, right-click the attribute that represents the child object and select **Add Instance** from the context menu.

Note: If the **Card** property of the attribute that references the child business object is set to the value 1 (indicating it is of single-cardinality), then you will only be able to add one instance of the child object.

Removing child business objects

To remove an instance of a child business object, right-click the instance and select **Remove Instance** from the context menu.

To remove all instances of a child business object, right-click the attribute that represents the child business object and select **Delete All Instances** from the context menu.

Setting the verb of a child business object

You can set the verb of a child business object to test the effect that value has on the business process. This can be helpful when you are troubleshooting logic that involves the cross-referencing of child objects.

To set the verb of a child business object instance, right-click it and choose **Set Verb** from the context menu. When presented with the “Select Verb” prompt, selected the desired verb and click **OK**.

Using the Response BO toolbar

You can edit the attributes of a business object received by a destination connector before you send it as a response. The toolbar of the “Response BO” dialog that you use when doing so has several toolbar buttons that can be used to set the values of the business object. For more information, see “Editing response business objects” on page 170.

Saving a business object

You can save a business object in Test Connector so that it can be used for later tests, shared with technical support (to help troubleshoot problems), or used as response data. You can save any business object, including ones that you have created and ones that appear as requests in the Test Connector window of a destination connector. By default, business objects are saved to a file with a business object extension (`.bo`).

It is recommended that you create a directory or directory structure specifically for test data files, with subdirectories dedicated to each interface or to each connector, as appropriate. This organization makes the necessary files are easy to locate and makes testing more efficient. Furthermore, it is recommended that you give the test data file for a business object the same name as the business object definition itself.

Saving request business objects

Do the following to save a business object instance that you have created as a request:

1. Select the business object you want to save.
2. From the menu bar, select **Edit > Save BO**.
3. Navigate to the desired directory and specify a name for the file in the **File name** field.
4. Click **Save**.

Saving response business object

Do the following to save a business object instance that has been received by a destination instance of Test Connector and will be sent as a response:

1. Select the business object instance in the "BO Request List" pane.
2. Select **Request > Edit Response** from the menu bar.
3. Click **Save BO**.
4. Navigate to the desired directory and specify a name for the file in the **File name** field.
5. Click **Save**.

Loading a business object

To load a business object that has been saved to a file, do the following:

1. Select **Edit > Load BO** from the menu bar of Test Connector.
2. Navigate to the business object test data file and open it.
3. When presented with the "New Instance" dialog, type a name for the instance in the **Enter Name** field.
4. Click **OK**.

Deleting a business object

To delete a business object from Test Connector, select **Edit > Delete BO** from the menu bar.

Note: This action only removes the business object from the Test Connector. It does not remove the connector's support for the business object definition.

Accepting a request business object

When you send a business object as a request, the business object appears in the "BO Request List" pane of any Test Connector instances that are emulating destination connectors in the interface, provided that the transaction did not fail.

After you have accepted the request business object, you can edit it if necessary as described in "Editing response business objects" on page 170.

Working with response business objects

Response business objects are those that you send from Test Connector when it is emulating a connector that is the recipient of business object requests in an interface. Working with request business objects consists of editing the values in the business object instance and sending the response back to the broker.

Editing response business objects

When you receive a business object request in a destination instance of Test Connector, you commonly want to edit the values of the attributes. For instance, you will want to provide unique values for primary key attributes that participate in relationships, or you will want to modify the value of other attributes to test map or collaboration logic that responds differently depending on the exact values in the business object. Do the following to set the values of business object attributes :

1. Select the business object instance in the "BO Request List" pane.
2. Select **Request > Edit Response** from the menu bar.
3. Do the following to edit the attributes of the business object:
   - Use one of the techniques described in "Setting values for business object attributes" on page 167to modify the values of the business object attributes.
   - Click **Reset BO to default** to set the values of the business object attributes to their default values as specified in the business object definition.
   - Click **Clear BO values** to clear the values of all the attributes in the business object.
   - Click **Load BO** to populate the attributes of the business object with test data from a file.

     The ability to load saved data into a business object request is very useful in situations where you have to populate a response business object with data before sending it as a reply. Instead of manually typing a value for each attribute that requires response data, you can type the values once, save the business object (as described in "Saving a business object" on page 168), and then load the saved data on subsequent tests.

Sending a response business object

After you accept a request business object, edit the business object, if needed, and send it back as a reply.

Table 28 lists Test Connector's reply options and shows their corresponding connector return codes for both C++ and Java connectors. For more detailed information about C++ or Java Connector return codes, see the *Connector Development Guide for Java or C++*.

Table 28. Test Connector reply types and connector return codes.

Test Connector reply type	C++ connector return code	Java connector return code
Success	BON_SUCCESS	SUCCESS
Fail	BON_FAIL	FAIL
Multiple Hits	BON_MULTIPLE_HITS	MULTIPLE_HITS
Retrieve By Content Fail	BON_FAIL_RETRIEVE_BY_CONTENT	RETRIEVEBYCONTENT_FAILED
Not Found	BON_BO_DOES_NOT_EXIST	BO_DOES_NOT_EXIST
Value Duplicate	BON_VALDUPES	VALDUPES

To reply to a request business object, do the following:

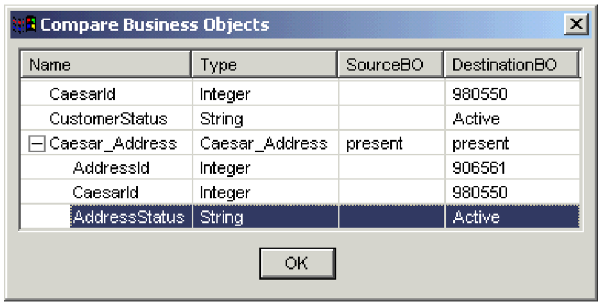
1. Select the business object in the "BO Request List" pane.
2. From the menu bar, select **Request > Reply**.
3. Select an item from the **Reply** submenu.

Comparing business object instances

Test Connector can compare two business objects of the same type and display the attributes that differ in value. You can use this function to view changes to a business object at different points in the execution of a transaction (for instance, you could compare a business object that has been sent to the integration broker with the same business object after the integration broker has updated it). To compare two business objects, do the following:

1. Create a request business object instance by following the instructions in either "Creating request business objects" on page 164or "Loading a business object" on page 169.
2. Select the response business object instance in the "BO Request List" pane that you would like to compare the request business object instance to.
3. From the menu bar, select **Edit > Compare BO's**.

   Test Connector opens the "Compare Business Objects" window with a table that displays the attributes which have different values in the two business objects. Figure 47 on page 171shows a comparison between two business object instances.



Name	Type	SourceBO	DestinationBO
CaesarId	Integer		980550
CustomerStatus	String		Active
Caesar_Address	Caesar_Address	present	present
AddressId	Integer		906561
CaesarId	Integer		980550
AddressStatus	String		Active

Figure 47. The Business Object Comparison Window.

4. Click **OK** to close the window.

Appendix H. Upgrading WebSphere Business Integration adapters

This appendix describes the process for upgrading to new releases of WebSphere Business Integration Adapters. It includes the following topics:

- "Assumptions"
- "Installing WebSphere Business Integration adapters"
- "Upgrading WebSphere message brokers"
- "Creating the response queue" on page 174
- "Updating the value of the local queue manager's coded character set ID" on page 174
- "Upgrading existing adapters" on page 174
- "Configuring new adapters" on page 177
- "Customizing the new WebSphere MQ batch files" on page 177.

Assumptions

These upgrade procedures assume the following:

- You are upgrading on a system that uses WebSphere MQ Integrator Broker as the integration broker. (Prior to the current release, other message brokers were not supported.)
- You have WebSphere Business Integration Adapters version 2.0, 2.0.1, or 2.1.0, 2.2.0, or 2.3.1 currently installed on your system.
- You have backed up any necessary files.
- You have stopped your system and any connectors that are running.
- You have read the release notes for any release-specific upgrade information that may affect your installation. You can find the release notes online in the WebSphere Business Integration Adapters InfoCenter at: http://www.ibm.com/websphere/integration/wbiadapters/infocenter.
- You will perform the upgrade in a development environment, then move the upgrades to your production environment after testing is completed.

Installing WebSphere Business Integration adapters

To install WebSphere Business Integration Adapters, follow the instructions in the *Installation Guide for WebSphere Business Integration Adapters*, available at the following Web address: http://www.ibm.com/software/websphere/wbiadapters/infocenter.

Upgrading WebSphere message brokers

Note: If you are upgrading from WebSphere MQ Integrator Broker or WebSphere MQ Integrator to WebSphere Business Integration Message Broker, be sure to import the message set and flow generated in the prior message broker (WebSphere MQ Integrator Broker or WebSphere MQ Integrator) to WebSphere Business Integration Message Broker as part of the upgrade process.

WebSphere MQ Integrator Broker

If you are upgrading from WebSphere Business Integration Adapters version 2.3.1, apply the WebSphere MQ Integrator Broker CSD05 Service Pack. Obtain the package from the following URL:

```
http://www.ibm.com/software/integration/mqfamily/support/
summary/mqsib.html
```

WebSphere MQ Integrator

Contact your IBM representative for details of how to upgrade WebSphere MQ Integrator.

WebSphere Business Integration Message Broker

This is the first release for which WebSphere Business Integration Message Broker is supported, so no upgrades are necessary.

Installing service packs for WebSphere MQ

If you are upgrading from WebSphere Business Integration Adapters version 2.3.1, apply the WebSphere MQ CSD05 Service Pack. Obtain the package from the following URL:

```
http://www.ibm.com/software/integration/mqfamily/support/summary/
```

Creating the response queue

If you are upgrading from a WebSphere Business Integration Adapters release prior to 2.2.0, you need to define a new response queue, as follows. If you already have release 2.2.0 or later, you do not need to perform this task.

WebSphere Business Integration Adapters version 2.2.0 requires an additional WebSphere MQ queue, a response queue. Define this queue using a method listed under “Ways to define queues” on page 78. In “Updates to the connector configuration files” on page 175, you must assign the name of this queue to the new connector configuration standard property, ResponseQueue.

Updating the value of the local queue manager’s coded character set ID

If you are upgrading from WebSphere Business Integration Adapters version 2.0, execute the following command to update each queue manager’s CCSID:

```
runmqsc alter qmgr ccsid(1208)
```

For further information about this command, see *WebSphere MQ: Script (MQSC) Command Reference*.

Upgrading existing adapters

For each existing adapter you are upgrading, perform the tasks listed in the topics below:

- “Upgrading business objects” on page 175
- “Updates to the connector configuration files” on page 175
- “Configuring the new connector startup scripts” on page 177

Upgrading business objects

Upgrading from releases 2.0.1, or 2.1.0, 2.2.0, or 2.3.1

It is not necessary to upgrade business objects if you are upgrading from WebSphere Business Integration Adapters versions 2.0.1, or 2.1.0, 2.2.0, or 2.3.1.

Upgrading from release 2.0

Note: Perform this step only if upgrading existing adapters from WebSphere Business Integration Adapters version 2.0.

For each adapter being upgraded, perform the following actions.

1. Using the new version of Business Object Designer that you have installed, load the adapter's existing business object definitions and save them using the same names. The business object definitions will be saved in the latest format used by WebSphere Business Integration Adapters.
2.

   Note: If you are upgrading using WebSphere Business Integration Message Broker 5.0 as the integration integration broker, skip this step. Instead, update the business objects by reimporting them with System Manager, as described in "Deploying to a message broker workspace" on page 85.

   Import the revised business object definitions into WebSphere MQ Integrator Broker's MRM using the XML Schema Importer with reset mode, which removes all existing information from the message set before the import of the updated XML Schema is started. To re-import the business object definitions, issue the following command for each definition:

   ```
   mqsiimpxmlschema -c -f parameter_file -o filename
   ```

   Where:

   parameter_file
   : is the name of the XML Schema importer parameters file.

   filename
   : is the file in which the importer is to write the message set name followed by the message set identifier, separated by an equal sign (=). If the file already exists the line will be appended to this file.

   For more information on importing XML schemas and using the `mqsiimpxmlschema` command, see *WebSphere MQ Integrator Broker: Working with Messages*
3. If Business Object Designer and the WebSphere MQ Integrator Broker Control Center are on separate machines that do not use a shared file system, copy the generated files to the machine where the integration broker is running.
4. From WebSphere MQ Integrator Broker's Control Center, replace the existing message sets and messages in your workspace.
5. Replace the existing assigned message sets with the new ones.
6. Re-deploy the message sets on the integration broker.

Updates to the connector configuration files

New releases of WebSphere Business Integration Adapters include certain changes to the connector configuration files. Refer to the sections below depending upon the release from which you are upgrading.

Upgrades from release 2.3.1

If you are upgrading from WebSphere Business Integration Adapters versions 2.3.1, be aware that there is a new standard configuration property XMLNameSpaceFormat. The default value for this is `short`. If you want the XMLNameSpaceFormat to be `long`, you must explicitly specify this in the connector configuration file. You might also have to change the value of the RFH2messagedomain property, depending on which message broker you are using. Refer to "Choosing XML Namespace length" on page 90for instructions on choosing the correct XMLNameSpaceFormat.

Upgrades from release 2.2.0

WebSphere Business Integration Adapters versions 2.3.0 and later include the following change to the connector configuration file . A new property, RFH2MessageDomain, has been added to the standard configuration properties. This property allows you to set the message domain to **xml**, if desired (default is **mrm**). Refer to Appendix A, "WebSphere MQ message formats," on page 99 and Appendix C, "Standard configuration properties for connectors," on page 109 for more information.

When you open your existing connector configuration file using the new version of Connector Configurator, the necessary changes to the file are made automatically. However, you may wish to verify that the new property is present and set as desired. To do so, perform the following actions:

1. Using the new version of Connector Configurator that you have installed, open the adapter's connector configuration file.
2. Click the **Standard Properties** tab.
3. Make any additional configuration property changes specific to the connector as specified in its adapter user guide.
4. Save the file using its existing name.

Upgrades from releases prior to 2.2.0

WebSphere Business Integration Adapters versions 2.2.0 and later include the following changes to the connector configuration file :

- The file format is now XML.
- The delivery transport mechanism is now called JMS not WMQI-JMS.
- A new property, ResponseQueue, is required but not used with WebSphere MQ Integrator Broker. Assign this property the name of the response queue you have defined in "Creating the response queue" on page 174.
- Certain standard connector configuration properties have been renamed.
- Additional standard and application-specific connector configuration properties are now defined.
- You can now specify a client mode queue configuration using an existing standard property. You are no longer required to manually edit the connector configuration file. The messaging section you needed to add manually to the connector configuration file to specify client mode is automatically deleted when you save the connector configuration file using the current version of Connector Configurator.

Tip

For information about using the Connector Configurator and to learn about changes to connector configuration properties in the new version of the adapter, refer to the adapter 's user guide.

When you open your existing connector configuration file using the new version of Connector Configurator, the necessary changes to the file are made automatically. However, you should still verify that the changes are correct as described below.

For each upgraded adapter, perform the following actions.

1. Using the new version of Connector Configurator that you have installed, open the adapter's connector configuration file.
2. Click the **Standard Properties** tab.
3. Ensure that the values for the following properties are set correctly for the business integration system:
   a. CharacterEncoding
   b. jms.MessageBrokerName (formerly called QueueManagerName)
   c. jms.Password (formerly QueueManagerPassword)
   d. jms.UserName (formerly QueueManagerLogin)
   e. jms.NumConcurrentRequests (formerly ConcurrentRequests)
   f. Locale
   g. SynchronousRequestTimeout (formerly Timeout).
4. Make any additional configuration property changes specific to the connector as specified in its adapter user guide.
5. Save the file using its existing name.

Configuring the new connector startup scripts

The startup scripts for connectors have changed sufficiently in WebSphere Business Integration Adapters 2.2.0 that you need to use the new versions. Customize the startup scripts, as described in "Configuring the connector startup files, shortcuts, and environment variables" on page 95.

Important
The current version of the WBIA Installer creates shortcuts in a Windows installation only. If you have existing CDE shortcuts, they are not updated.

Configuring new adapters

For each new adapter you have installed, perform the configuration tasks listed in the topics below:

- "Creating business object definitions" on page 81
- "Creating a message broker project" on page 82
- "Enabling the application for use with the connector" on page 91
- "Configuring the connector" on page 91
- "Defining message flows" on page 96.

Customizing the new WebSphere MQ batch files

WebSphere Business Integration Adapters version 2.2.0 provides batch files that you can use to configure the WebSphere MQ queues needed for your adapters or to clear the messages from WebSphere MQ queues when necessary. You can use these batch files to configure or clear the WebSphere MQ queues for every adapter in your business integration system using a single command. For more information about customizing and running these batch files see "Using WebSphere Business

Integration Adapters batch files to configure WebSphere MQ queues” on page 78 and “Clearing messages from WebSphere MQ queues” on page 67

Glossary

A

adapter. A set of software modules that communicate with an integration broker and with applications or technologies to perform tasks such as executing application logic and exchanging data. An IBM WebSphere Business Integration Adapter always consists of the adapter framework and a connector specific to an application or technology. An adapter might also contain a sample business object specific to the application or technology, an Object Discovery Agent (ODA) designed to generate business object definitions specific to the application or technology, or both.

adapter development kit (ADK). A development environment for creating custom adapters.

adapter framework. The software that IBM provides to install, configure, and run an adapter.

application connector. A connector that is designed to interact with a specific application. Application-specific connectors are intermediaries between an integration broker and applications. These connectors convert application-specific data into business objects that can be manipulated by components of the integration broker, and convert business objects from the components into data that can be received by the specific application.

application-specific component. The component of a connector that contains code tailored to a particular application or technology. This component initializes a business object handler to respond to requests, and, if needed, implements an event-notification mechanism to detect and respond to events that an application or external programmatic entity initiates. The code for this component is written in C++ or Java, depending on the language of the API provided by the application or technology.

B

business integration system. A system, consisting of an integration broker and a set of integration adapters, which allows heterogeneous business applications to exchange data through the coordinated transfer of information in the form of business objects.

business object. A set of attributes that represent a business entity (such as an Employee), an action on the data (such as a create or update operation), and instructions for processing the data. Components of the business integration system use business objects to exchange information and trigger actions.

C

character conversion. Encoding applied to a character so that it retains its meaning when it is transferred from a location that uses one character code set to a location that uses a different code set. See also **character encoding**.

character encoding. The mapping from a character (a letter of the alphabet) to a numeric value in a character code set. For example, the ASCII character code set encodes the letter "A" as 65, while the EBCIDIC character set encodes this letter as 43. The character code set contains encodings for all characters in one or more language alphabets.

connector. A set of software modules (the connector framework and a connector's application-specific component) that uses business objects to send information about an event to an integration broker or to receive information about a request from the integration broker. See also **application connector** and **technology connector**.

connector framework. The component of a connector that manages interactions between a connector's application-specific component and the integration broker. This component provides all required management services, and retrieves the metadata that the connector requires from the repository. The connector framework, whose code is common to all connectors, is written in Java and includes a C++ extension to allow the development of application-specific components written in C++.

I

integration broker. A program that integrates data among heterogeneous applications. An integration broker typically provides a variety of services that include: the ability to route data, a repository of rules that govern the integration process, connectivity to a variety of applications, and administrative capabilities that facilitate integration.

L

local repository. A collection of metadata that describes components of the business integration system such as business objects whose data is transferred across applications. It also contains the configuration information associated with the connector framework and a connector's application-specific component. It is also referred to as the connector's local repository.

locale. The part of a user's environment that brings together information about how to handle data that is specific to the end user's particular country, language, or territory. The locale is typically specified when configuring the operating system or internationalized software products.

O

Object Discovery Agent (ODA). A tool designed to "discover" business object requirements specific to a data source and to generate business object definitions from those requirements. Business Object Designer presents a forms-based interface to available ODAs, and helps manage the discovery and definition generation processes.

Object Discovery Agent development kit (ODK). An API for creating Object Discovery Agents (ODAs).

T

technology connector. A connector that is designed for interactions that conform to a specific technology. The WebSphere Business Integration Adapter for XML, for example, can be an intermediary through which an integration broker sends data to a web server (or other programmatic entity) using the XML format, even if that web server resides on a network that is not running a WebSphere business integration system.

Index

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

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

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

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

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

IBM WebSphere Business Integration Adapter Framework V 2.4.0

Printed in USA