

IBM WebSphere Business Integration Adapters



Adapter for Siebel eBusiness Applications User Guide

Version 4.3.x

IBM WebSphere Business Integration Adapters



Adapter for Siebel eBusiness Applications User Guide

Version 4.3.x

Note!

Before using this information and the product it supports, read the information in "Notices" on page 125

05September2003

This edition of this document applies to connector version 4.3.x, and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

© Copyright International Business Machines Corporation 1997, 2003. All rights reserved.

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

Contents

About this document	v
Audience	v
Related documents	v
Typographic conventions	v
New in this release	vii
New in release 4.3.x	vii
New in release 4.2.x	vii
New in release 4.1.x	vii
New in release 4.0.x	vii
Chapter 1. Overview of the connector	1
Definition of the connector	1
Siebel application architecture	1
Siebel virtual business components	2
Connector architecture	2
How the connector works	2
Chapter 2. Installing and configuring the connector	9
Compatibility	9
Prerequisites	9
Installing the adapter	10
Installed files the connector	30
Event and archive tables	32
Configuring the connector	35
Connector startup	39
Chapter 3. Understanding business objects	41
Business object structure and relationships	41
Business object application-specific information	43
Using business object samples	52
Chapter 4. Generating business object definitions using SiebelODA	57
Installation and usage	57
Using SiebelODA in Business Object Designer	60
Contents of the generated definition	72
Adding information to the business object definition	74
Chapter 5. Using the Siebel connectivity DLL	75
Connector architecture and processes	75
Configuring the Siebel connectivity DLL	77
Communicating with the Siebel connectivity DLL	79
Generating WBIA business objects	79
Configuring the IBM generic business services	80
Chapter 6. Using business services	83
Understanding business services	83
Verb processing with business services	85
Events detection with business services	86
Chapter 7. Troubleshooting	87
MAX_LOG_FILE_SIZE error on UNIX	87
Handling errors	87

Decreasing the size of the Siebel log file	88
Appendix A. Standard configuration properties for connectors	89
New and deleted properties	89
Configuring standard connector properties	89
Summary of standard properties	90
Standard configuration properties	94
Appendix B. Connector Configurator	105
Overview of Connector Configurator	105
Starting Connector Configurator	106
Running Configurator from System Manager	107
Creating a connector-specific property template	107
Creating a new configuration file	109
Using an existing file	111
Completing a configuration file	111
Setting the configuration file properties	112
Saving your configuration file	118
Changing a configuration file	119
Completing the configuration	119
Using Connector Configurator in a globalized environment	119
Appendix C. Connector feature list	121
Business object request handling features	121
Event notification features	121
General features	122
Notices	125
Programming interface information	126
Trademarks and service marks	126

About this document

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

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

Audience

This document is for WebSphere business integration system consultants and customers. To use the information in this document, you should be knowledgeable in the following areas:

- Connector development
- Business object development
- Siebel application architecture
- Siebel Tools
- Visual Basic

Note: If you are a consultant or customer located in Japan and are using Siebel 2000, you must use the Adapter for Siebel 2000 User Guide.

Related documents

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

You can install the documentation or read it directly online at one of the following sites:

- If you are using WebSphere MQ Integrator Broker or WebSphere Application Server as your integration broker:
<http://www.ibm.com/software/websphere/wbiadapters/infocenter>
- If you are using WebSphere InterChange Server as your integration broker:
<http://www.ibm.com/websphere/integration/wicsserver/infocenter>

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

Typographic conventions

This document uses the following conventions:

<code>courier font</code>	Indicates a literal value, such as a command name, file name, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
<i>italic, italic</i>	Indicates a variable name or a cross-reference.

<i>blue outline</i>	A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference.
<i>ProductDir</i>	Represents the directory where the product is installed.

New in this release

New in release 4.3.x

Updated in July, 2003. The adapter can now use WebSphere Application Server as an integration broker. For further information, see “Compatibility” on page 9. The adapter now runs on the following platforms:

- Solaris 7,8
- AIX 5.x
- HP-UX 11i

New in release 4.2.x

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

The changes to this version of the connector support Siebel, version 7.5 and the Siebel connectivity DLL.

New in release 4.1.x

The connector delivered with IBM WebSphere Business Integration Adapter for Siebel eBusiness Applications has been internationalized. For more information, see “Processing locale-dependent data” on page 7 and Appendix A, “Standard configuration properties for connectors,” on page 89

New in release 4.0.x

The IBM WebSphere business integration adapter for Siebel eBusiness Applications includes the connector for Siebel eBusiness Applications. This adapter operates with both the InterChange Server (ICS) and WebSphere MQ Integrator integration brokers. An integration broker, which is an application that performs integration of heterogeneous sets of applications, provides services that include data routing. This adapter includes:

- An application-component specific to Siebel eBusiness Applications
- SiebelODA
- Sample business objects
- IBM WebSphere Adapter Framework, which consists of:
 - Connector Framework
 - Development tools (including Business Object Designer and Connector Configurator)
 - APIs (including ODK, JCDK, and CDK)

This manual provides information about using this adapter with both integration brokers: InterChange Server (ICS) and WebSphere MQ Integrator.

Important: Because the connector has not been internationalized, do not run it against InterChange Server version 4.1.1 if you cannot guarantee that only ISO Latin-1 data will be processed.

Chapter 1. Overview of the connector

This chapter provides an overview of the connector component of the WebSphere Business Integration Adapter (WBIA) for Siebel eBusiness Applications and contains the following sections:

- “Definition of the connector”
- “Siebel application architecture”
- “Siebel virtual business components” on page 2
- “Connector architecture” on page 2
- “Business object processing” on page 3
- “Event management” on page 5
- “Handling lost connections to the Siebel application” on page 7
- “Processing locale-dependent data” on page 7

Definition of the connector

Connectors consist of two parts: the connector framework and the application-specific component. The connector framework, whose code is common to all connectors, acts as an intermediary between the integration broker and the application-specific component. The application-specific component contains code tailored to a particular application. The connector framework provides the following services between the integration broker and the application-specific component:

- Receives and sends business objects
- Manages the exchange of startup and administrative messages

This document contains information about the connector framework and the application-specific component, which it refers to as the connector.

The connector component of the adapter provides connectivity to either Siebel 6.2.x, 7.0, or 7.5 (English) and only Siebel 7 (Japanese) using the WebSphere Business Integration Adaptor (WBIA) framework and the Siebel eBusiness Applications components.

The connector uses the Java APIs provided by the Siebel Java Data Bean to communicate with the Siebel Object Manager for data exchange. Siebel Java Data Bean exposes the Siebel business components and business objects, which are part of the business object layer in the Siebel application architecture.

Siebel application architecture

The Siebel application architecture contains three layers, as follows:

- User interface objects layer--This layer contains the visual elements that the user interacts with.
- Business objects layer--This layer contains both business components and business objects. A business component is a fundamental business entity, consisting of multiple fields that represent it. A business object is a collection of related business components. The Siebel connector communicates with this layer using the Siebel Java Data Bean.

- Data objects layer--This layer contains the object definitions which provide logical representation of the underlying physical database. It is independent of the installed relational database management system, and it is not accessible by the Siebel Java Data Bean.

Siebel virtual business components

In addition to the Connector, the WebSphere business integration system includes components for implementing Siebel Virtual Business Components (VBCs). See the *Implementation Guide for Siebel Virtual Business Components* for more information.

Connector architecture

The connector has been designed following the meta-data design principles as outlined in the *Connector Development Guide for Java*. This means that existing application-specific business objects can be extended and customized and new business objects can be defined without requiring additional coding or customization in the connector code.

The following diagram illustrates the Siebel connector architecture.

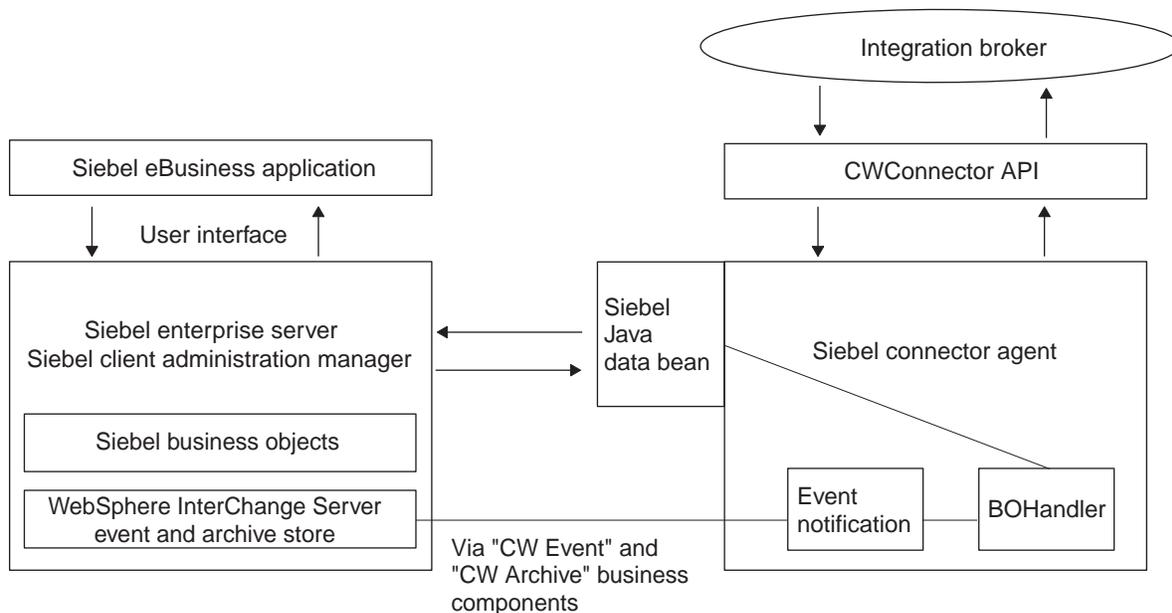


Figure 1. Siebel connector architecture

How the connector works

This section describes how meta-data enhances the connector's flexibility, and presents a high-level description of business object processing and event notification.

The connector and meta-data

The connector is meta-data-driven. Meta-data is application-specific data that is stored in business objects and that assists the connector in its interaction with the application. A meta-data-driven connector handles each business object that it supports based on meta-data encoded in the business object definition rather than on instructions hardcoded in the connector. A business object corresponds to a Siebel business component. For more information about business objects, see Chapter 6, "Using business services," on page 83

Business object meta-data includes the structure of a business object, the settings of its attribute properties, and the content of its application-specific information. Because the connector is meta-data driven, it can handle new or modified business objects without requiring modifications to the connector code.

Business object processing

This section provides an overview of how the connector processes integration broker requests and application events.

Processing integration broker requests

When the connector receives a request from a business object to perform an application operation, the connector processes hierarchical business objects recursively; that is, it performs the same steps for each child business object until it has processed all individual business objects.

Note: The term **hierarchical** business object refers to a complete business object, including all the child business objects that it contains at any level. The term **individual** business object refers to a single business object, independent of any child business objects it might contain or that contain it. The term **top-level** business object refers to the individual business object at the top of the hierarchy that does not itself have a parent business object.

Business object retrieval: When an integration broker asks the connector to retrieve a hierarchical business object from the Siebel application, the connector attempts to return a business object to the integration broker that exactly matches the current representation of a Siebel business component instance. In other words, all simple attributes of each individual business object returned to the integration broker match the value of the corresponding field in the Siebel business components.

To retrieve the complete business component, the connector uses the primary key values in the top-level business object received from the integration broker to recursively descend through the corresponding data in the database.

Business object RetrievalByContent: When an integration broker asks the connector to retrieve a hierarchical business object based on values in non-key attributes in the top-level business object, the connector uses the value of all non-null attributes as the criteria for retrieving the data.

Business object creation: When an integration broker asks the connector to create a hierarchical business object in the Siebel application, the connector creates all the children of the top-level business object prior to creating the parent. An exception to this rule is when the relationship between the parent and child is a multi-value link in Siebel and the link is inactive. In this case, the child is created after the parent, and the keys are generated by the Siebel application.

Business object modification: Business object modification, or updating, involves comparing the retrieved after image of the business object from Siebel with the inbound business object. The process involves setting the correct verb on the child objects. If the keys are set on the parent and all other attributes are set to CxIgnore, the parent update is skipped.

The default behavior is to compare the after image from the Siebel applications with the inbound business object, then change the verbs on the child container objects. This process ensures that all the children in the Siebel application are made the same as the inbound business object. If the verb is not set on the children, the default is set to Update.

Important: If some of the children need to be retained, the inbound object verb must be set to DeltaUpdate, and verbs must be set on each one of the child container objects. In this case, only these objects in the Siebel application are processed while the others are left untouched.

Business object deletion: When an integration broker asks the connector to delete a record, the record is removed from the underlying database. Only the parent needs to be deleted because the Siebel DeleteCascade feature deletes all of the children. If any of the required attributes are missing from the inbound business object, the delete fails.

Exists verb: The primary business component name is typically the same business object name in Siebel. If the ObjectName and ComponentName application specific information match, the keys are set on this business component and the query is executed. If the record exists, it returns True; if the record does not exist, it returns False.

Processing application events

Components: The event notification requires the creation of event and archive tables in the Siebel database. You must create CW Event and CW Archive, two new Siebel business components corresponding to these tables.

Triggering: The creation, update, or delete of any record in the Siebel eBusiness application can be treated as an event. Siebel supports Visual Basic scripts and Siebel eScripts embedded in the Siebel business component event handlers to populate the event table. On a call to pollForEvents, these event records are obtained and processed. The Event business component stores information about the event, as listed in Table 1

Note: The information in Table 1 is used by the connector during event subscription to build corresponding business objects and to send those objects to the connector framework for further processing.

Table 1. Events business component structure

Fields	Description
Object Key	The unique identifier that identifies the business object row for which the event was created
Object Name	Siebel business object for which the event was deleted
Object Verb	Verb for the event
Priority	Event priority

Table 1. Events business component structure (continued)

Fields	Description
Status	Event Status Initially, this is set to READY_FOR_POLL. Other status values include: IN_PROGRESS=1 -- The event has been picked up and is sent to the connector framework. The connector changes the status of the event to IN_PROGRESS after it picks the event for processing. UNSUBSCRIBED=2 -- The event has not been subscribed for. The connector sets the status to UNSUBSCRIBED if the isSubscribed call returns a False. SUCCESS=3 -- The event was successfully processed by the connector framework. The connector sets the status to SUCCESS if the event is processed successfully by the connector framework. ERROR_PROCESSING_EVENT=-1 -- There was an error processing the event. This status is set if there was an error while processing the event. ERROR_POSTING_EVENT=-2 -- There was an error posting the event to the connector framework. This status is set if the call to getApplEvent fails in pollForEvents. ERROR_OBJECT_NOT_FOUND=-3 -- The object for which the event was created could not be found. This status is set if the doVerbFor call could not find the object in pollForEvents.
Description	Any comment associated with the event
Event Id	Id of the event row
ConnectorId	Identifies the connector in a multiple connector configuration
Event Ts	Event creation timestamp

Create notification: When the connector encounters a Create event, it creates a business object of the type specified by the event, sets the key values for the business object (using the object key specified in the CW Event business component), and retrieves the business object from the Siebel application. After it retrieves the business object, the connector sends it with the Create verb to the integration broker.

Update notification: When the connector encounters an Update event, it creates a business object of the type specified by the event, sets the key values for the business object (using the object key specified in the CW Event business component), and retrieves the business object from the database. After it retrieves the business object, the connector sends it with the Update verb to the integration broker.

Delete notification: When the connector encounters a Delete event, it creates a business object of the type specified by the event, sets the key values for the business object (using the object key specified in the CW Event business component), and sends it with the Delete verb to the integration broker. All values other than the key values are set to CxIgnore.

Retrieving business objects for event processing: Retrieval of objects for event processing is based on both key and non-key attributes. It is mandatory that the business object support the RetrieveByContent verb.

Event management

The connector's event detection mechanism uses a CW Event business component and a CW Archive business component. Because there are potential failure points associated with the processing of events, the event management process does not delete an event from the CW Event business component until it has been inserted into the CW Archive business component.

The connector polls the CW Event business component at a regular, configurable interval, retrieves the events, and processes the events first by priority and then sequentially. When the connector has processed an event, the event's status is updated appropriately.

The setting of its ArchiveProcessed property determines whether the connector archives an event into the CW Archive business component after updating its status. For more information on the ArchiveProcessed property, see "Configuring the connector" on page 35

Table 2 illustrates the archiving behavior depending on the setting of the ArchiveProcessed property.

Table 2. Archiving behavior

Archive processed setting	Event processing status	Connector behavior
true or no value	Successful	Event is deleted from the CW Events business component and archived with status of Success
	Unsuccessful	Archived with status of Error
	No subscription for business object	Event is deleted from the CW Events business component and archived with one of the following statuses:Error Processing Event Error Posting Event Error Object Not Found
false	Successful	Not archived and remains in the CW Events business component with a status of Success
	Unsuccessful	Event is not archived and remains in the CW Events business component with one of the following statuses:Error Processing Event Error Posting Event Error Object Not Found
	No subscription for business object	Remains in event table with status of Unsubscribed

Smart filtering

Duplicate events are not saved in the event store. Before storing a new event as a record in the event store, the VB Script or eScript needs to query the event store for existing events that match the new event. The event detection mechanism does not generate a record for a new event in the following cases:

- If the business object name, verb, status, and ConnectorId (if applicable) in a new event match those of another unprocessed event in the event store.
- If the business object name, key, and status for a new event match an unprocessed event in the event table, and the verb for the new event is Update while the verb for the unprocessed event is Create.
- If the business object name, key, and status for a new event match an unprocessed event in the event table, and the verb in the unprocessed event in the event table is Create while the new verb is Delete. In this case, remove the Create record from the event store.

Handling lost connections to the Siebel application

The connector terminates when an error message specified in the `ConnectErrors` connector property is detected. The text from `ConnectErrors` is compared with the Siebel error message. If a match is found, the connector returns `AppResponseTimeout`, which terminates the connector.

The `ConnectErrors` message can be returned by the Siebel application if the connection is lost and the connector tries to:

- Access the CW Event and CW Archive business components
- Retrieve the business object that is related to the event
- Create or update a record pertaining to a business object.

Processing locale-dependent data

The connector has been internationalized so that it can support double-byte character sets, and deliver message text in the specified language. When the connector transfers data from a location that uses one character code set to a location that uses a different code set, it performs character conversion to preserve the meaning of the data. The Java runtime environment within the Java Virtual Machine (JVM) represents data in the Unicode character code set. Unicode contains encodings for characters in most known character code sets (both single-byte and multibyte). Most components in the WebSphere business integration system are written in Java. Therefore, when data is transferred between most Server Access components, there is no need for character conversion. To log error and informational messages in the appropriate language and for the appropriate country or territory, configure the `Locale` standard configuration property for your environment. For more information on these properties, see Appendix A, “Standard configuration properties for connectors,” on page 89

Chapter 2. Installing and configuring the connector

This chapter describes how to install and configure the connector component of the WebSphere Business Integration Adapter (WBIA) for Siebel eBusiness Applications. The following topics are covered:

- “Compatibility”
- “Prerequisites”
- “Installing the adapter” on page 10
- “Installed files the connector” on page 30
- “Event and archive tables” on page 32
- “Configuring the connector” on page 35
- “Connector startup” on page 39

Compatibility

The adapter framework that an adapter uses must be compatible with the version of the integration broker (or brokers) with which the adapter is communicating. The 4.3.x version of the adapter for Siebel eBusiness Applications is supported on the following adapter framework and integration brokers:

Adapter framework: WebSphere Business Integration Adapter Framework version 2.x

Integration brokers:

- WebSphere InterChange Server, versions 4.1.1 and 4.2.x
- WebSphere MQ Integrator, version 2.1.0
- WebSphere MQ Integrator Broker, version 2.1.0
- WebSphere Application Server Enterprise, version 5.0.1, with WebSphere Studio Application Developer Integration Edition, version 5.0.1

Prerequisites

Before you use the connector, you must do the following:

- Install the Siebel 6.2.x or Siebel 7 .jar files that will be used.
- Verify the existence of a user account in the application. This user account must be the same as the user specified in the Siebel scripts for event creation in Siebel Tools.
- Copy the Siebel Connector.txt file from the `%ProductDir%/connectors/messages/Siebel` directory to the `%ProductDir%/connectors/messagesdirectory`

User setup

Before installing the connector, you must create a user account for the connector in Siebel. This user account should have full access privileges, and the login name should be the same as the `ApplicationUserName` configuration property. The default value for the user account login name and password is `CWCONN`.

When installing the connector, be sure to install the files from one of the following lists to the %ProductDir%/Connectors/Siebel/dependencies directory. The files are located on either Siebel 6 or Siebel 7 server.

Important: The start_Siebel.bat file in the %ProductDir%/Connectors/Siebel directory currently has the English and Japanese Siebel .jar files in the JCLASSES variable. This is added to the CLASSPATH. For any other language supported by Siebel, the corresponding .jar file must be added to the JCLASSES variable.

Siebel 6

- SiebelDataBean.jar
- SiebelTC_enu.jar
- SiebelTcCommon.jar
- SiebelTcOM.jar

Siebel 7

- SiebelJI_Common.jar
- SiebelJI_enu.jar

Installing the adapter

The information in this section describes how to install WebSphere Business Integration Adapters (WBIA).

You install the WBIA product by running a platform-specific executable for the installer. Table 3 lists the installer executable for each operating system. The installer executables are located in the WebSphereBI directory on the product CD.

Table 3. Platform-specific executables for WBIA Installer

Operating system	WBIA Installer executable file
Windows	setupwin32.exe
AIX	setupAIX.bin
Solaris	setupsolarisSparc.bin
HP-UX	setupHP.bin

You can use the installer executable file to perform the installation in the following ways:

- You can start the graphical installer as described in “Invoking the graphical WBIA Installer” on page 11 and then proceed through the installation wizard to make your selections as described in “Using the graphical WBIA Installer” on page 12.
- You can perform a silent installation as described in “Performing a silent installation” on page 20.

Note: These procedures assume that you are installing from a product CD. If you obtain your software from Passport Advantage, make sure you have downloaded it. Refer to your Passport Advantage information for those downloading instructions.

Note: If you are installing the adapters to communicate with InterChange Server, you must install the broker first. See the installation guide for InterChange Server on the appropriate platform for information on how to install the broker.

Important: Make sure you are logged in as the WebSphere business integration system administrator before you install the adapters. When you install on a UNIX computer, the permissions of the folders and files that are created are set based on the permissions of the user account that performs the installation.

Important: You must not install WBIA as root. The entry that is added to the Object Data Manager (ODM) when installing as root prevents you from using SMIT to uninstall other applications, so you should not install WBIA as root.

Invoking the graphical WBIA Installer

The graphical WBIA installer presents you with a wizard that allows you to make choices about the installation of the WBIA product. Although the installer is Java-based and therefore platform-independent, there are different ways of invoking the installer for each platform. This section describes the approaches for both Windows and UNIX computers.

Invoking Installer in a Windows environment

To invoke Installer in a Windows environment, navigate to the WebSphereBI directory of the product CD and execute `setupwin32.exe`.

Invoking Installer in a UNIX environment

The WBIA installer in a UNIX environment is invoked through a `.bin` file specific to the platform, located in the WebSphereBI directory. Table 3 on page 10 provides the name of the `.bin` file for each platform.

Follow the steps in one of the following sections to invoke the installer depending on how you are working with the UNIX computer:

- “If you are running CDE on the UNIX computer”
- “If you are connecting to the UNIX computer through X emulation software”

If you are running CDE on the UNIX computer: If you are running the Common Desktop Environment (CDE) and working on the UNIX computer directly then you can navigate to the WebSphereBI directory of the product CD and double-click the `.bin` file specific to the operating system.

You can also navigate to the WebSphereBI directory of the product CD and execute the `.bin` file at the command line. The following example shows how to do so on a Solaris computer:

```
# ./setupsolarisSparc.bin
```

If you are connecting to the UNIX computer through X emulation software: If you are using a Windows computer to connect to the UNIX computer through X emulation software do the following to invoke the installer:

1. Determine the IP address of the Windows computer that you are using to connect to the UNIX computer.

You can execute the `ipconfig` command at the Windows command line interface to display the IP address of the Windows computer.

2. Set the DISPLAY environment variable on the UNIX computer to the IP address determined in step 1 on page 11.

You must be sure to follow the IP address with a colon and the identifier for the monitor or display on the Windows client computer. If the Windows client computer only has a single monitor then the display value is 0.0.

The following example shows the DISPLAY environment variable being set to the single monitor on a Windows computer whose IP address is 9.26.244.30:

```
DISPLAY=9.26.244.30:0.0
```

3. Export the DISPLAY environment variable by executing the following command:
export DISPLAY
4. Start the X emulation client on the Windows computer and connect to the UNIX computer.
5. Navigate to the WebSphereBI directory of the product CD at the command line of the X emulation client.
6. Execute the .bin file specific to the operating system. For example, if the UNIX computer was running AIX then you would execute the following command:
./setupAIX.bin

The graphical installer starts on the Windows computer that you are using to connect to the UNIX computer.

Using the graphical WBIA Installer

Once running, the WBIA Installer prompts you to make your installation choices and then performs the installation.

When you select an adapter to install, the runtime components and the data handlers required to support it are automatically selected for installation. However, if Installer determines that current versions of the runtime components are already on the system, then they are not installed.

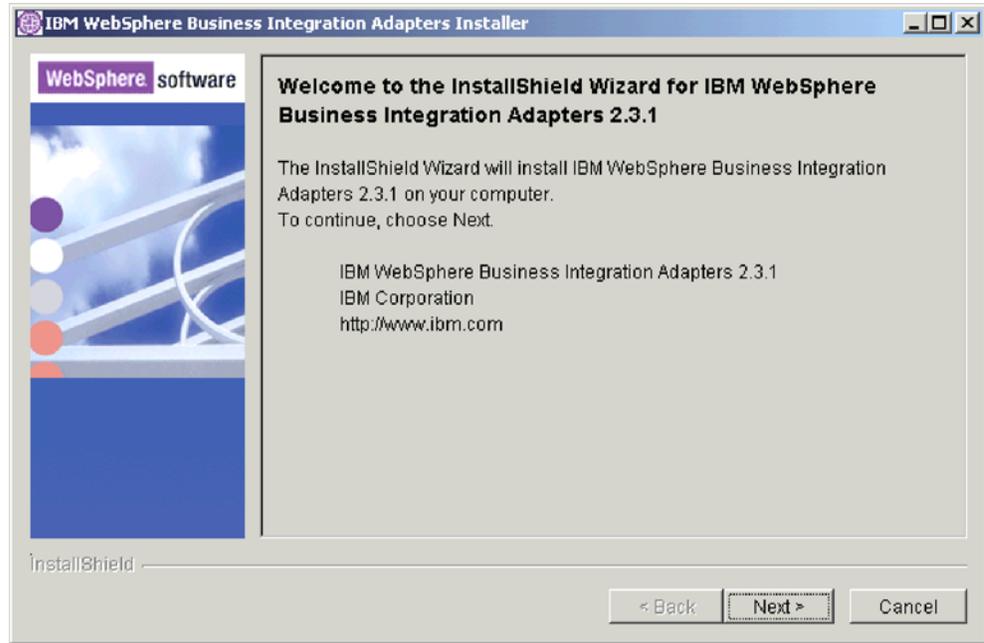
Note: If you click **Cancel** while the business integration adapters are installing, some files will still appear in the newly created directory. The number of files depends on how far the process had progressed before installation was canceled.

Do the following to proceed through the installer:

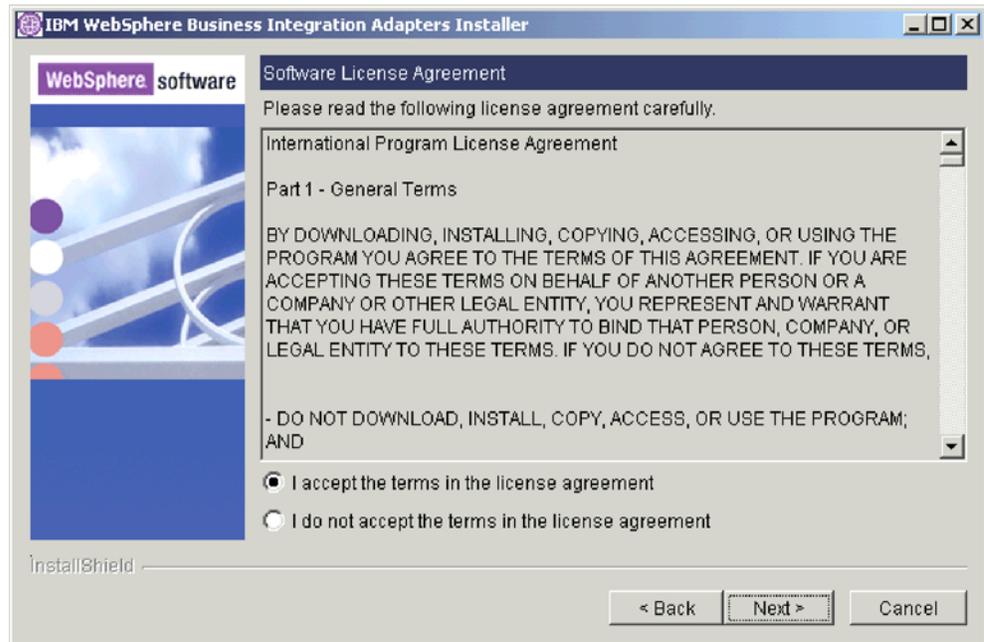
1. At the “language selection” prompt, choose the desired language from the drop-down menu and click **OK**.



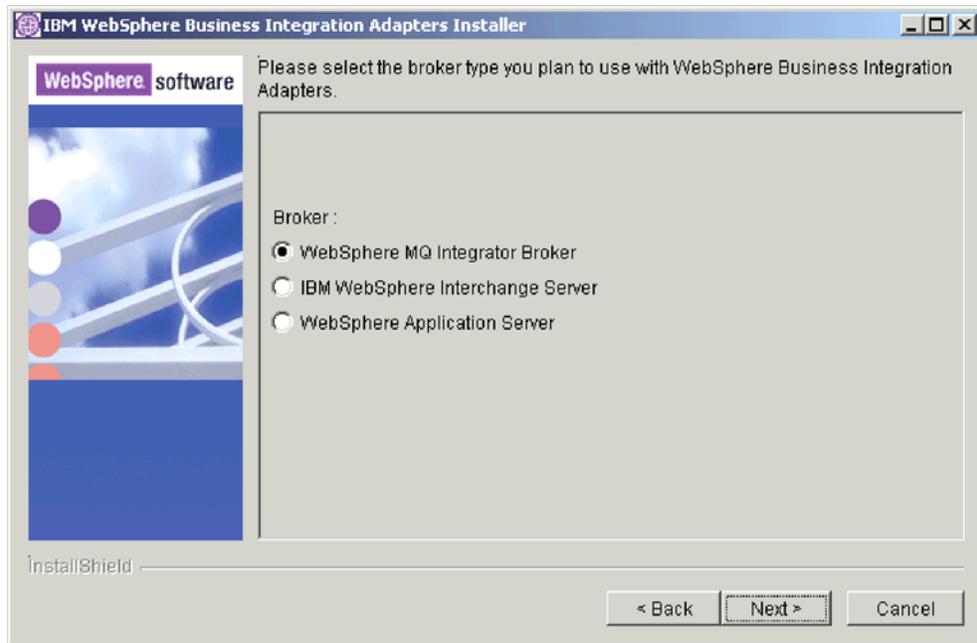
2. At the “Welcome” screen click **Next**.



3. At the "IBM License Acceptance Panel", click **I accept the terms in the license agreement** and then click **OK**.



4. At the "broker selection" screen click the radio button for the type of integration broker with which the adapters will communicate and then click **Next**.



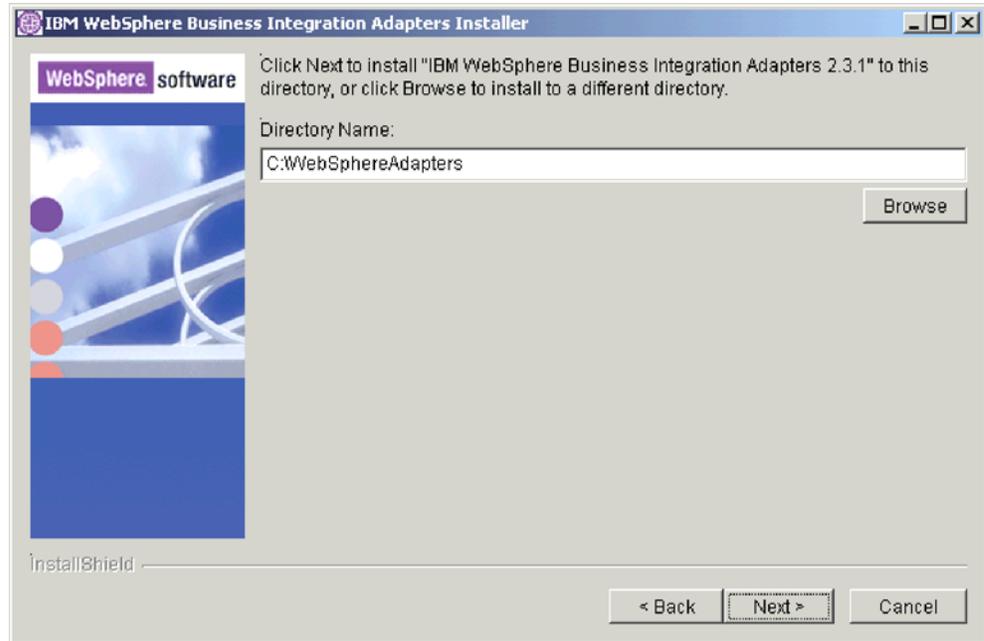
5. At the “installation directory” screen, either type the full path of the directory into which the adapters should be installed, click **Browse** to select a directory, or accept the default path, then click **Next**.

Important: You must specify an installation directory that does not have spaces in the path.

Table 4 shows the default adapter installation directories for the different integration brokers on the different supported platforms.

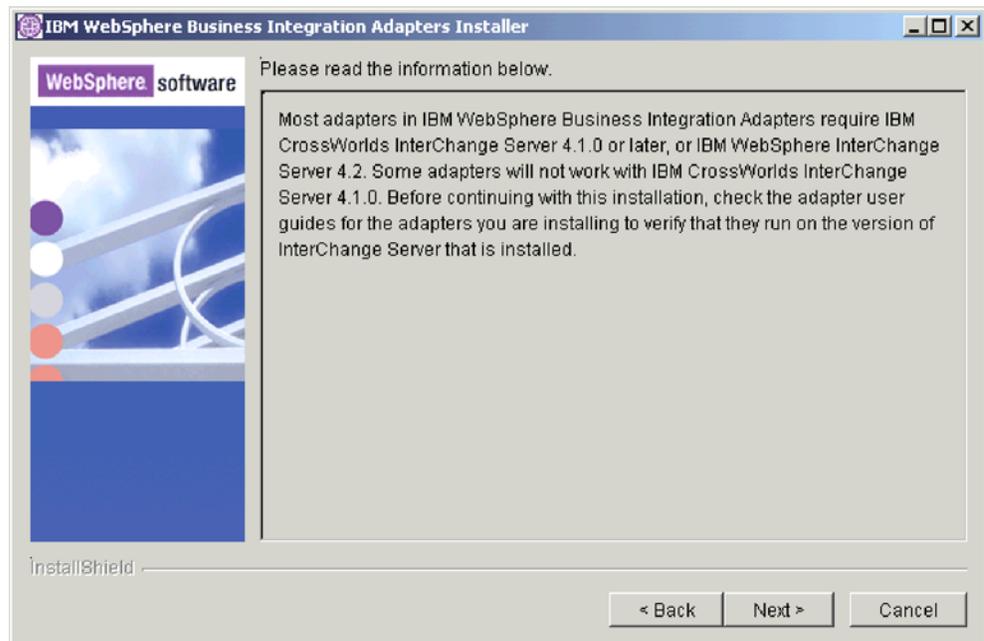
Table 4. Default adapter installation directories

Broker type	Default Windows directory	Default UNIX directory
WebSphere MQ Integrator Broker	C:\WebSphereAdapters	/\$HOME/WebSphereAdapters
WebSphere InterChange Server	C:\IBM\WebSphereICS	/\$HOME/IBM/WebSphereICS
WebSphere Application Server	C:\WebSphereAdapters	/\$HOME/WebSphereAdapters

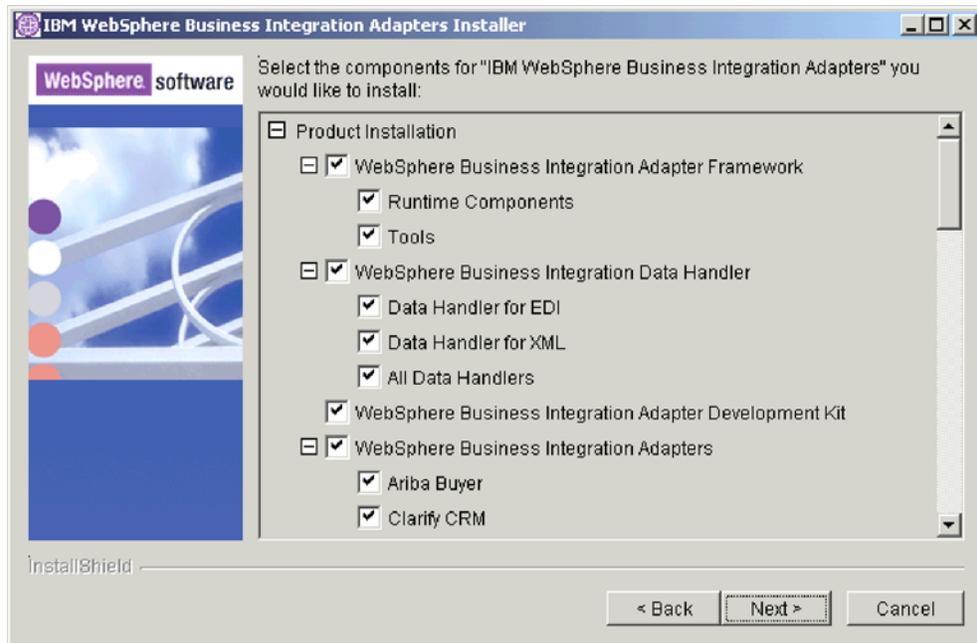


Important: If your broker is WebSphere InterChange Server then you must specify the WebSphere InterChange Server product directory at this screen. If you do not specify the same directory for the adapter installation as for the broker installation then the adapters will not be able to run.

6. If you selected WebSphere InterChange Server as your broker in step 13 then Installer presents an informational screen at this time. Read the information and then click **Next**.



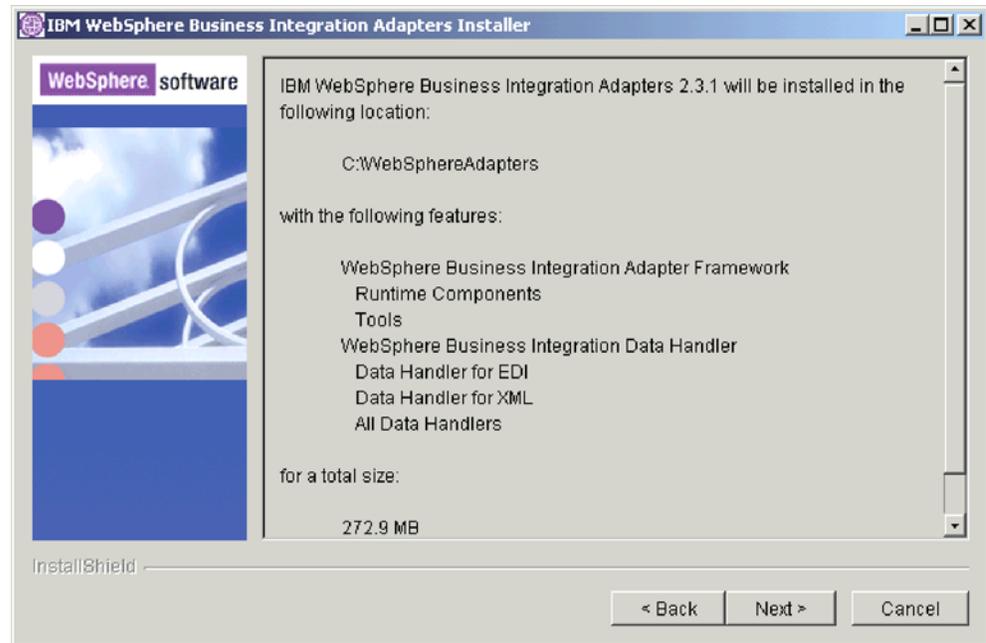
7. At the “component selection” screen, enable checkboxes for the adapters and features you want to install and then click **Next**.



Different features are available for installation depending on the broker. For instance, if you chose WebSphere MQ Integrator Broker or WebSphere Application Server as your broker then the adapter runtime and tools are available for selection. If you chose WebSphere InterChange Server, though, these are not available because the tools and runtime are installed as part of the broker installation.

Different features are also available for installation depending on the operating system. For example, the tools are only supported in Windows environments and are therefore only available for selection when you run the installer on a Windows computer.

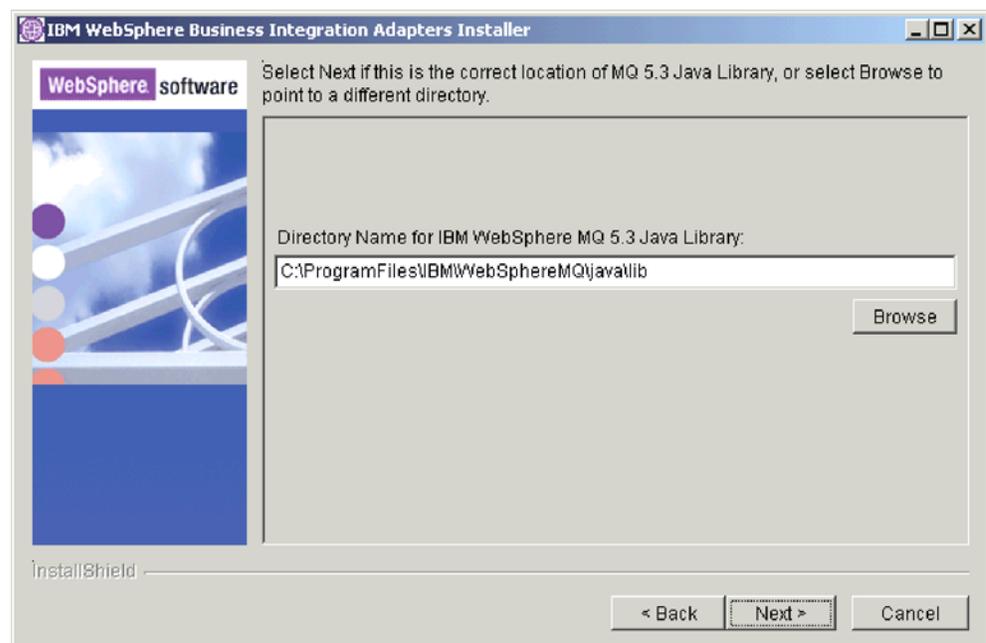
8. The “summary” screen lists the features selected for installation, the specified product directory, and the amount of disk space required. Read the information to verify it and then click **Next**.



9. Do the following depending on which broker you selected during step 4:

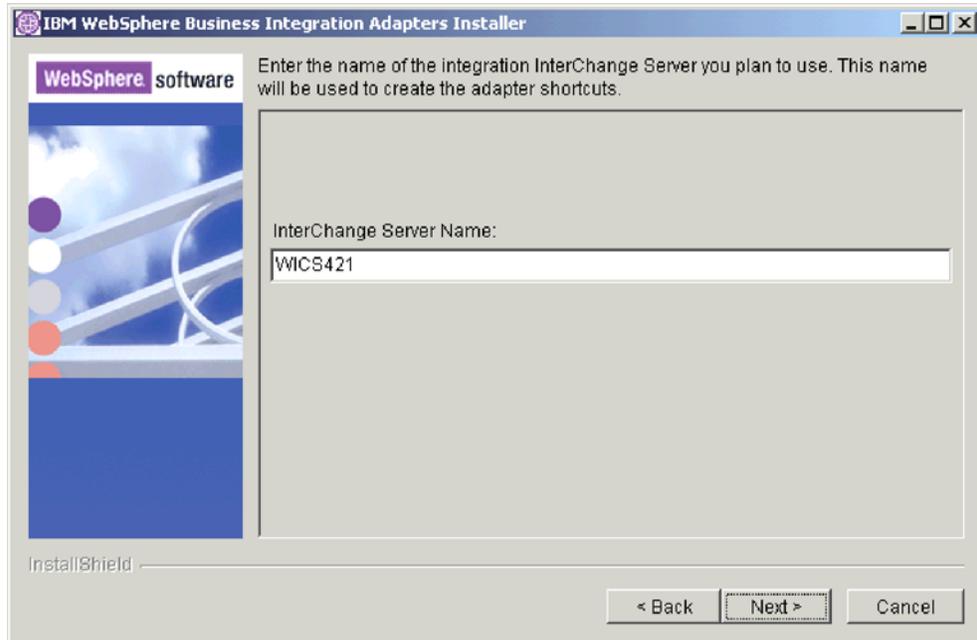
- If you selected WebSphere MQ Integrator Broker or WebSphere Application Server as your broker in step 4 then Installer presents the “MQ Java library directory” screen.

Either type the directory path to the java\lib directory of the WebSphere MQ installation or click **Browse** to select the directory, then click **Next**.



- If you selected WebSphere InterChange Server as your broker in step 13 then Installer presents the “InterChange Server name” screen at this time. Type the name of the InterChange Server instance with which the adapters will communicate in the **InterChange Server Name** field and then click **Next**.

Important: Be sure to specify the name of the InterChange Server instance accurately. The server name is case-sensitive and if it is not typed correctly then the connectors will not be able to communicate with the server.



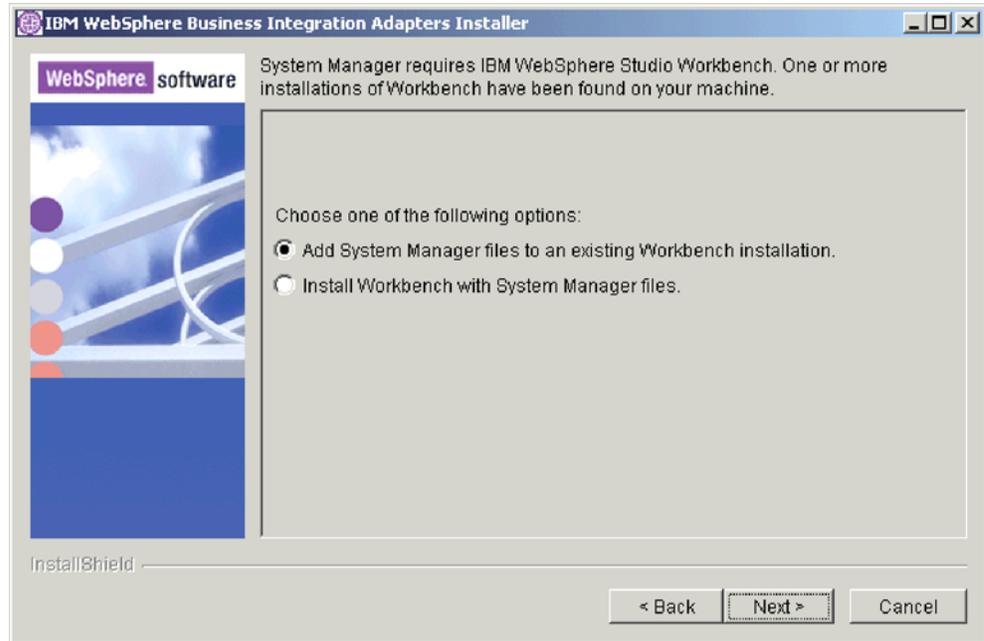
10. If the following are true then Installer next presents the “workbench selection” screen where you specify how you want the tools required to work with the integration components installed:
- If you are installing on a Windows computer
 - If you selected WebSphere MQ Integrator Broker or WebSphere Application Server as your integration broker
 - If you chose the **Tools** feature to be installed
 - If WebSphere Studio Application Developer Integration Edition version 5.0.1 is installed on the computer

Choose **Add System Manager files to an existing Workbench installation** if you want to add the plug-ins to an existing installation of WSADIE 5.0.1. You will be required to specify the product directory of the existing installation of WSADIE 5.0.1 at the next Installer screen, as described in step 11 on page 19.

Important: The plug-ins will not work if there are spaces in the path of the workbench product directory. If there are spaces in the path of your existing WSADIE 5.0.1 installation then you must not make this installation choice.

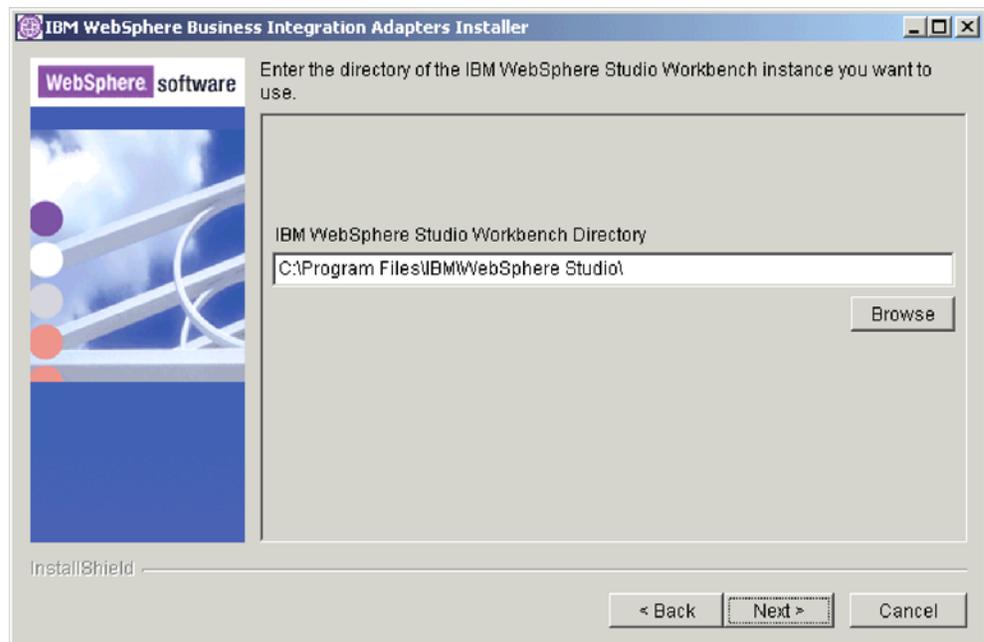
Choose **Install Workbench with System Manager files** if you want to install WebSphere Studio Workbench 2.0.3 and add the plug-ins to that installation. The workbench is installed in a subdirectory of the adapter installation directory named Tools\WSWB203 if you make this selection.

Click **Next** after making your selection.

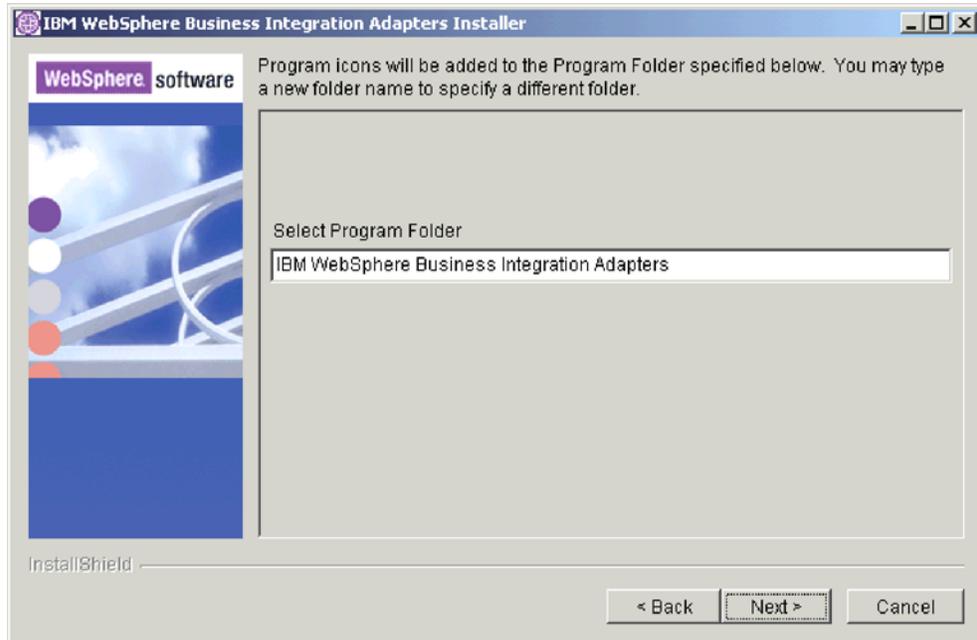


11. If you were presented with the “workbench selection” screen as described in step 10 on page 18 and you chose **Add System Manager files to an existing Workbench installation** then Installer next presents the “workbench installation directory” screen. Type the path to the WSADIE 5.0.1 product directory or click **Browse** to select the directory, then click **Next**.

Important: The plug-ins will not work if there are spaces in the path of the workbench product directory. If there are spaces in the path of your existing WSADIE 5.0.1 installation then you must click **Back** and select **Install Workbench with System Manager files** instead.



- If you are installing on a Windows computer then installer next presents the “select program folder” screen. Type the name of the program group in which shortcuts for the adapters should be created or accept the default program group and then click **Next**.



- After Installer finishes successfully, click **Finish**.

Performing a silent installation

You can perform a silent installation of WBIA, where you provide your installation choices in a file rather than at the screens of the installation wizard. This is particularly helpful when you have to perform multiple installations that are identical.

To perform a silent installation, first create a file with your installation choices as described in “Preparing the installation response file” and then perform the installation using the file as described in “Executing the silent installation” on page 25.

Preparing the installation response file

When performing a silent installation, you prepare a response file that contains your installation choices. IBM provides four response file templates that contain the options for different combinations of integration broker and platform support. The response file templates are located in the WebSphereBI directory on the product CD. Table 5 lists the response file templates for each combination.

Table 5. Response files templates

Integration broker and platform combination	Response file template
WebSphere InterChange Server on Windows:	settings_WBIAInstaller_ICS.txt
WebSphere Application Server and WebSphere MQ Integrator Broker on Windows:	settings_WBIAInstaller_WSMQ.txt
WebSphere InterChange Server on UNIX:	settings_WBIAInstaller_ICS_unix.txt

Table 5. Response files templates (continued)

WebSphere Application Server and WebSphere MQ Integrator Broker on UNIX:	settings_WBIAInstaller_WSMQ_unix.txt
--	--------------------------------------

Table 6 lists the options that can be used during a silent installation. Be sure to read all the information in the **Option values** column, as it specifies when particular options should be commented out and what the broker and platform compatibilities are.

Table 6. Silent installation options

Option name	Option values
brokerType.brokerSelection	<p>Set to 1 for WebSphere MQ Integrator Broker.</p> <p>Set to 2 for WebSphere InterChange Server.</p> <p>Set to 3 for WebSphere Application Server.</p>
setWINICSDest.value	<p>Set to the directory path in which InterChange Server is installed.</p> <p>If you leave this option commented, the product installs to the default directory listed in Table 4 on page 14.</p> <p>This option is only relevant for the WICS broker when installed on Windows. Make sure it is commented out when installing for the WMQI or WAS brokers, or when installing on a UNIX computer.</p>
setWINWMQIDest.value	<p>Set to a valid directory path, without spaces, to specify the location in which the WBIA product should be installed.</p> <p>If you leave this option commented, the product installs to the default directory listed in Table 4 on page 14.</p> <p>This option is only relevant for WMQI and WAS brokers when installed on Windows. Make sure it is commented out when installing for the WICS broker, or when installing on a UNIX computer.</p>
setUnixICSDest.value	<p>Set to the directory path in which InterChange Server is installed.</p> <p>If you leave this option commented, the product installs to the default directory listed in Table 4 on page 14.</p> <p>This option is only relevant for the WICS broker when installed on UNIX. Make sure it is commented out when installing for the WMQI or WAS brokers, or when installing on a Windows computer.</p>

Table 6. Silent installation options (continued)

Option name	Option values
setUnixWMQIDest.value	<p>Set to a valid directory path, without spaces, to specify the location in which the WBIA product should be installed.</p> <p>If you leave this option commented, the product installs to the default directory listed in Table 4 on page 14..</p> <p>This option is only relevant for WMQI and WAS brokers when installed on UNIX. Make sure it is commented out when installing for the WICS broker, or when installing on a Windows computer.</p>
-P main_product.active	<p>Set to true to install all of the components available in the WBIA product.</p> <p>Set to false to install selected components.</p>
-P f_connectors.active	<p>Set to true to install all adapters.</p> <p>Set to false to install specified adapters.</p>
-P adaptorFrameWork.active	<p>Set to true for adapters to work with WMQI and WAS brokers.</p> <p>Comment out this option when installing for the WICS broker.</p>
-P WindowEnvSetup.active	<p>Set to true for adapters to work with WMQI and WAS brokers.</p> <p>Comment out this option when installing for the WICS broker or when installing on UNIX.</p>
-P support.active	<p>Set to true for adapters to work with WMQI and WAS brokers on UNIX.</p> <p>Comment out this option when installing for the WICS broker or when installing for the WMQI or WAS brokers on Windows.</p>
-P supportFileRequirement.active	<p>Set to true for adapters to work with WMQI and WAS brokers on UNIX.</p> <p>Comment out this option when installing for the WICS broker or when installing for the WMQI or WAS brokers on Windows.</p>
-P frameworkTOOLS.active	<p>Set to true when WMQI is the broker and Windows is the platform to install the tools used to work with the installed components.</p> <p>Set to false if WICS or WAS are the broker, if the platform is UNIX, or if you do not want to install the tools.</p>
-P adaptorDevelopmentKit.active	<p>Set to true when WMQI is the broker and Windows is the platform to install the Adapter Development Kit, which provides the interfaces necessary to develop custom adapters.</p> <p>Set to false if WICS or WAS are the broker, if the platform is UNIX, or if you do not want to install the Adapter Development Kit.</p>

Table 6. Silent installation options (continued)

Option name	Option values
-P cn<adap ^t value>.active	<p>Set to true to install a specific adapter, where <i>adap^tvalue</i> is the value that identifies the adapter.</p> <p>For example, cnJDBC identifies the Adapter for JDBC.</p> <p>Refer to the response file templates for the available adapters and the values associated with them.</p> <p>Not all adapters are available for all brokers and platforms.</p> <p>Refer to the response file templates to determine if a particular adapter is available for that combination of broker and platform.</p>
-P dataHandler.active	Set to true to enable installation of data handlers.
-P <datahandl ^e rvalue>.active	<p>Set to true to install a specific data handler, where <i>datahandl^ervalue</i> is the value that identifies the data handler.</p> <p>For example, xmlDataHandler identifies the XML data handler.</p> <p>Refer to the response file templates for the available data handlers and the values associated with them.</p> <p>Installer evaluates any data handler dependencies for selected adapters, so you do not have to specify data handlers to be installed just to satisfy adapter requirements. You would need to select data handlers for installation if you plan to use them for adapters you plan to develop and if the data handlers will not be installed by default.</p>
-P AllDataHandlers.active	<p>Set to true to install all data handlers.</p> <p>Set to false to install specified data handlers.</p>
-W mqDirectoryUserInput.mqLibraryLocation	<p>Set to the path of the java\lib directory within the WebSphere MQ installation on the computer when installing for either the WMQI or WAS brokers.</p> <p>Comment out this option when installing for the WICS broker.</p> <p>Comment out this option if you want to use the default values. On the Windows platform, installer searches the Windows registry by default and uses the value it finds there. On the AIX platform, installer uses the default value /usr/mqm/java/lib. On the Solaris and HP-UX platforms, installer uses the default value /opt/mqm/java/lib.</p>
-W workbenchChoice.workbenchList	<p>Set to 1 to add the tools plugins to an existing installation of WebSphere Studio Workbench or WebSphere Studio Application Developer Integration Edition.</p> <p>Set to 2 to install WebSphere Studio Workbench.</p>

Table 6. Silent installation options (continued)

Option name	Option values
-W workbenchLocation.workbenchLocDirectory	If you specified that WebSphere Studio Workbench be installed by setting the value of the workbenchChoice.workbenchList option to 2, then set this option to the directory in which the workbench should be installed, for example C:\WebSphereAdapters\Tools.
-W inputServer.name	Set to the name of the InterChange Server instance with which the installed adapters will communicate. The name is case-sensitive and must be specified accurately or the adapters will not be able to communicate with the server. You must make sure this option is not commented and is set to the proper value when installing adapters to communicate with the WICS broker. Comment out the option when installing for WMQI or WAS brokers.
-W inputShortcuts.folder	Set to the name of the program group created for the WBIA product, for example IBM WebSphere Business Integration Adapters. This option is only relevant on the Windows platform. Comment out this option when installing on UNIX.
-W createReposFile.active	Set to true to create a file that contains the definitions for the components selected for installation. Set to false to not create a file containing the definitions. The definition files for individual components are still copied to the repository directory even if you choose not to create this comprehensive file.
-G replaceExistingResponse	Set to yesToAll or yes to replace all files found on the system that have the same name as those being copied by the installer. Set to noToAll or no to not replace any files found on the system that have the same name as those being copied by the installer.
-G replaceNewerResponses	Set to yesToAll or yes to replace all files found on the system that are newer than those being copied by the installer. Set to noToAll or no to not replace any files found on the system that are newer than those being copied by the installer.
-G createDirectoryResponse	Set to yes to create the product directory specified by the option if it does not already exist. Set to no to not have the product directory created if it does not exist. You must set this option to yes if the specified directory does not exist for the installation to succeed.
-G removeExistingResponse	This option is not relevant for any broker on any platform. Comment out this option.

Table 6. Silent installation options (continued)

Option name	Option values
-G removeModifiedResponse	This option is not relevant for any broker on any platform. Comment out this option.

You can modify one of the response file templates and use it for the silent installation. In this case you should be sure that any options that are incompatible with others you require are commented out by placing a hash symbol # in front of them. Alternatively, you can create a new response file with the options you require. This approach has the benefit that no unnecessary options or commented description blocks are present to clutter the response file, making it easier to read and edit. If you take this latter approach, it is recommended that you copy a template file and remove the sections and options that are not necessary, rather than typing the necessary options into a new file.

Executing the silent installation

To execute the silent installation, you run the platform-specific installer executable at the command line with several options, including the name of the response file you prepared.

The following example shows how to do so on a Windows computer where the response file template for the WAS and WMQI integration brokers is used, and exists in the C:\data directory:

```
D:\WebSphereBI>setupwin32.exe -silent -options
C:\data\settings_WBIAInstaller_WSMQ.txt
```

The following example shows how to do so on an AIX computer where a custom response file named install.txt was used, and exists in the /home/icsadmin directory:

```
# ./setupAIX.bin -silent -options /home/icsadmin/install.txt
```

Installing additional adapters

To install additional adapters, run the WBIA Installer again. The word (installed) is displayed in the component selection screen next to all components that have already been installed on the system.

WBIA directories, files, and environment variables

Installer creates a number of directories, files, and environment variables depending on the selections you made while running Installer.

WBIA directories and files

After the installation is complete, you can view the file system and its contents. Table 7 lists some of the directories you might need to know about. The folders and files created vary depending on the choices made during the installation and on the operating system.

Table 7. WebSphere Business Integration Adapter directories

Directory name	Contents
_jvm231	This directory contains the Java Runtime files. Note: If you are upgrading from a previous version of WBIA, the existing directory name is retained from that release.

Table 7. WebSphere Business Integration Adapter directories (continued)

Directory name	Contents
_uninst_WBIA2.3.1	This directory contains files that are required to uninstall the WebSphere Business Integration Adapters product. For more information on uninstalling WBIA, see “Uninstalling WebSphere Business Integration Adapters” on page 27. Note: If you are upgrading from a previous version of WBIA, the existing directory name is retained from that release.
bin	This directory contains the executable files and shell scripts that the business integration adapters use.
connectors	This directory contains files specific to each adapter in the system. It also contains adapter-specific files that you may need to install in the application that the adapter supports.
DataHandlers	This directory contains the .jar files for the data handlers.
DevelopmentKits	This directory contains the files needed to develop connectors.
legal	This directory contains the license files.
lib	This directory contains shared libraries and .jar files for the system.
logs	This directory is provided to contain log and trace files.
messages	This directory contains message text files used by the connectors to generate log and trace messages.
ODA	This directory contains the .jar and .bat files for each Object Discovery Agent.
repository	This directory contains the connector definition files.
templates	This directory contains sample script files for creating and clearing WebSphere MQ queues. If your integration broker is WebSphere MQ Integrator Broker or WebSphere Application Server, see your broker implementation guide for more information on using these scripts. If your integration broker is WebSphere InterChange Server, see the installation guide for InterChange Server on the appropriate platform.
Tools	This directory contains the WSWB203 directory, which in turn contains the installation of WebSphere Studio Workbench if you chose to install it.

Environment variables

If you chose WebSphere MQ Integrator Broker or WebSphere Application Server as your broker, Installer takes the actions described in Table 8 on page 27 to create and update environment variables on the computer. These actions are not taken if you chose WebSphere InterChange Server as your integration broker, because the environment variables required for that broker are created during installation of the broker itself.

Table 8. Actions taken by Installer for environment variables

Environment variable name	Installer action
CROSSWORLDS	Creates this environment variable to reference the WBIA product directory, as specified when using Installer.
MQ_LIB	Creates this environment variable to contain the path to the Java\lib directory within the WebSphere MQ installation, as specified when using Installer.
CLASSPATH	Adds the following entries: <i>ProductDir\lib\rt.jar;</i> <i>ProductDir\DataHandlers\CwDataHandler.jar;</i>
PATH	Adds the following entries: <i>ProductDir\bin\hotspot;</i> <i>ProductDir\bin\classic;</i> <i>ProductDir\bin;</i>

Uninstalling WebSphere Business Integration Adapters

The method for uninstalling WBIA or selected adapters depends on how you installed the product.

If you installed the WBIA product from CD, follow the directions in “Uninstalling adapters installed from CD.”

If you installed the WBIA product using Electronic Software Delivery (ESD), follow the directions in “Uninstalling adapters installed from ESD” on page 30.

Uninstalling adapters installed from CD

You can uninstall adapters that were installed from a CD in the following ways:

- You can start the graphical uninstaller as described in “Invoking the graphical WBIA Uninstaller” and then proceed through the uninstallation wizard to make your selections as described in “Using the graphical WBIA Uninstaller” on page 28.
- You can perform a silent installation as described in “Performing a silent uninstallation” on page 29.

Important: The WBIA Uninstaller does not remove adapters installed from an ESD download. For information on removing adapters installed from an ESD download, see “Uninstalling adapters installed from ESD” on page 30.

Invoking the graphical WBIA Uninstaller: Follow the steps in one of the following sections to invoke the installer depending on the platform on which the WBIA product is installed:

- “To invoke the uninstaller in a Windows environment”
- “To invoke the uninstaller in a UNIX environment” on page 28

To invoke the uninstaller in a Windows environment: Navigate to the *ProductDir/_uninst_WBIA2.3.1* directory and execute the WBIA Uninstaller, *uninstaller.exe*.

To invoke the uninstaller in a UNIX environment: Navigate to the *ProductDir/_uninst_WBIA2.3.1* directory and execute the WBIA Uninstaller, *uninstaller.bin*.

If you are running the Common Desktop Environment and are working directly on the UNIX computer then you can double-click the *uninstaller.bin* file.

If you are using X emulation software to connect to the UNIX computer from a Windows computer then you must execute the *uninstaller.bin* file from the command line, as in the following example:

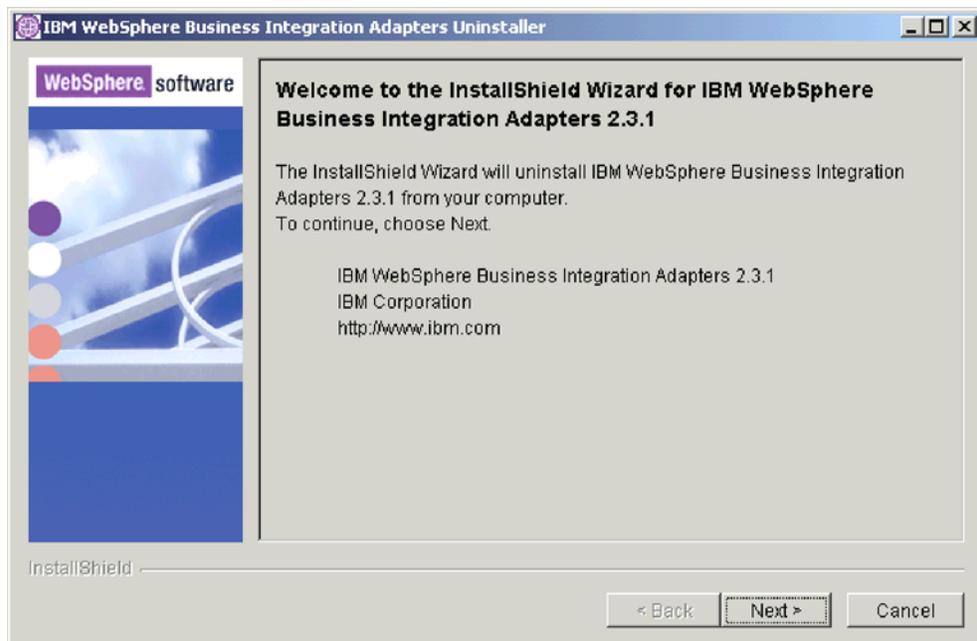
```
# ./uninstaller.bin
```

Using the graphical WBIA Uninstaller: Using the WBIA Uninstaller, do the following to uninstall either the entire system or selected components:

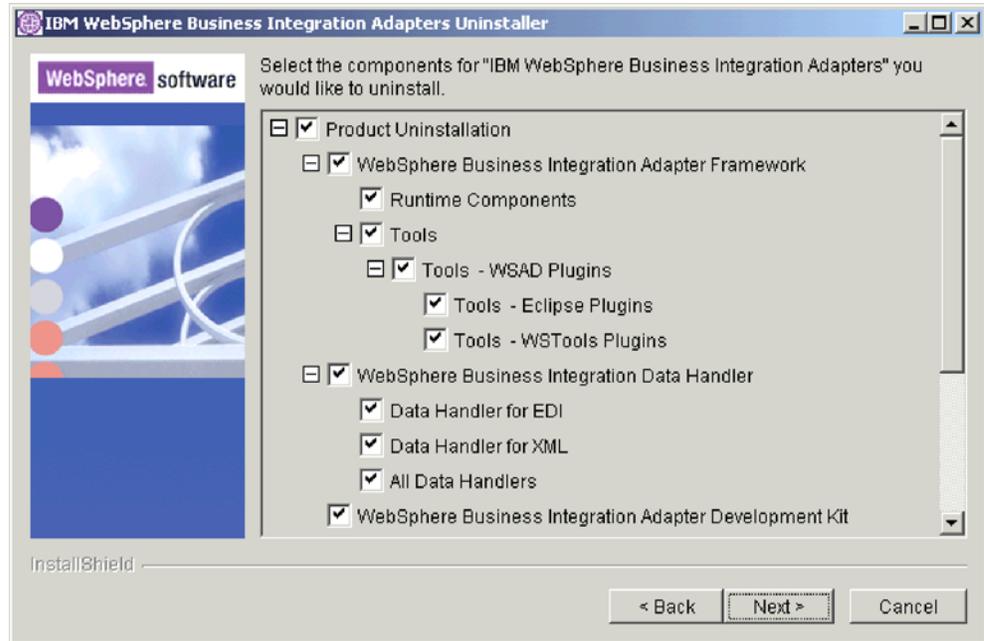
1. At the language selection prompt, choose the desired language from the drop-down menu and click **OK**.



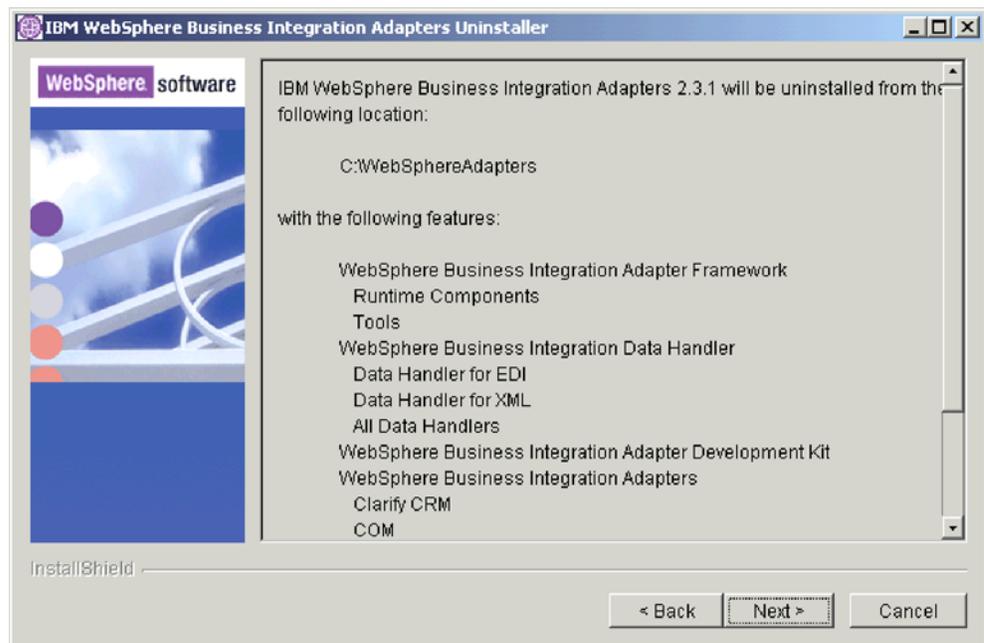
2. At the "Welcome" screen click **Next**.



3. At the "component selection" screen, ensure that the components you want uninstalled have checkboxes next to them. By default all WBIA product components are selected for uninstallation; clear the checkbox for any component you want to leave installed, and then click **Next**.



4. The “summary” screen lists the components that will be uninstalled and the product directory from which they will be removed. Read the information to verify it and then click **Next**.



5. After Uninstaller finishes successfully, click **Finish**.

Performing a silent uninstallation: To perform a silent uninstallation, you run the platform-specific uninstaller executable in the `_uninst_WBIA2.3.1` directory within `ProductDir` at the command line with the `-silent` option.

The following example shows how to do so on a Windows computer if the WBIA product is installed in `C:\WebSphereAdapters`:

```
C:\WebSphereAdapters\_uninst_WBIA2.3.1>uninstaller.exe -silent
```

The following example shows how to do so on an AIX computer:

```
# ./uninstaller.bin -silent
```

Note: If you upgraded from a previous version of WBIA, the directory `_uninst_WBIA2.3.1`, was not created. The old `_uninst_WBIA2.x.x` uninstallation directory is retained but now contains upgraded contents.

Note: Silent uninstallation does not uninstall adapters installed from an ESD. For information on how to uninstall adapters that were installed from an ESD, see "Uninstalling adapters installed from ESD."

Uninstalling adapters installed from ESD

If you installed a WebSphere Business Integration Adapters adapter using Electronic Software Delivery (ESD), the uninstaller is in a directory labeled `_uninstXXX`, where XXX identifies the adapter it uninstalls. For example, the uninstaller for the WBIA adapter for JMS is located in a directory named `_uninstJMS`. To uninstall an adapter that you downloaded using ESD, do the following:

1. From the command line, navigate to the appropriate uninstall directory.
2. To launch the uninstaller for an adapter, execute the following command:

```
java -cp uninstall.jar run
```

Note: The Java Runtime Environment must be installed to run this command.

Important: To uninstall the entire WebSphere Business Integration Adapters product, navigate to the *ProductDir*/`_uninst_WBIA2.3.1` and execute the WBIA Uninstaller named `Uninstaller.exe` on Windows and `uninstaller.bin` on Unix.

Installed files the connector

The following subsections describe the file structures of an installed connector on a UNIX or Windows system.

Installed files on a UNIX system

Table 9 describes the UNIX file structure used by the connector.

Table 9. Installed UNIX file structure for the connector

Subdirectory of $\$ProductDir$	Description
connectors/Siebel	Contains the CWSiebel.jar and the start_Siebel.sh files for the adapter. The startup script for the Siebel adapter is called from the generic connector manager script. When you click Install from the Connector Configurator, WebSphere MQ Integrator Broker as the integration broker, or the Connector Configuration screen of System Manager (ICS as the integration broker), the Installer creates a customized wrapper for this connector manager script. When the connector works with ICS, use this customized wrapper to start and stop the connector. When the connector works with WebSphere MQ Integrator Broker, use this customized wrapper only to start the connector; use mqsi remotestopadapter to stop the connector.
connectors/Siebel/dependencies	Contains the patch files for event management in the Siebel eBusiness application. Should also contain the siebel.jar files used by the Siebel connector.
connectors/messages/Siebel	Contains the relevant message file, SiebelConnector.txt. This file must be copied by the user from the
connectors/messages/Siebel/Java	connectors/messages/Siebel/Java directory. Contains the SiebelConnector.txt file (for Siebel 7 Java connector).
connectors/Siebel/Samples/Repository	Contains the following BO samples: Siebel_BCAccount Siebel_BCQuote Siebel_BCContact Siebel_BCInternalProduct Siebel_BCAsset
repository/Siebel	Contains the CN_Siebel.txt file.
/lib	Contains the WBIA.jar file.
/bin	Contains the CWConnEnv.sh file.

Installed files on a Windows system

Table 10 describes the Windows file structure used by the connector.

Table 10. Installed Windows file structure for the connector

Subdirectory of $\%ProductDir\%$	Description
\connectors\Siebel	Contains the connector CWSiebel.jar and the start_Siebel.bat files.
\connectors\Siebel\dependencies	Contains the patch files for event management in the Siebel eBusiness applications. This folder should also contain the Siebel.jar files.
\connectors\messages	Contains the relevant message file, SiebelConnector.txt. This file must be copied by the user from the connectors\messages\Siebel\Java directory

Table 10. Installed Windows file structure for the connector (continued)

Subdirectory of %ProductDir%	Description
\connectors\messages\Siebel\Java	Contains the SiebelConnector.txt file as well as the SiebelConnector_11_TT.txt files (message files specific to a language (_11) and a country or territory (_TT)).
\connectors\Siebel\Samples\Repository	Contains the following BO samples: Siebel_BCAccount Siebel_BCQuote Siebel_BCContact Siebel_BCInternalProduct Siebel_BCAsset
\repository\Siebel\ \lib	Contains the CN_Siebel.txt file. Contains the WBIA.jar file.
\bin	Contains the CWConnEnv.bat file.

Installer adds an icon for the connector file to the WebSphere business integration menu. For a fast way to start the connector, create a shortcut to this file on the desktop.

Note: For more information on WebSphere business integration Installer, refer to the *System Installation Guide for Windows* or *for Unix*.

Event and archive tables

The connector uses the event table to queue events for pickup. If you have set the ArchiveProcessed property to true or to no value, the connector uses the archive table to store events after updating their status in the event table.

For each event, the connector gets the business object's name, verb, and key from the CW Event business component. The connector uses this information to retrieve the entire entity from the application. If the entity was changed after the event was first logged, the connector gets the initial event and all subsequent changes. In other words, if an entity is created and updated before the connector gets it from the event table, the connector gets both data changes in the single retrieval.

The following three outcomes are possible for each event processed by a connector:

- Event was processed successfully
- Event was not processed successfully
- Event was not subscribed to

If events are not deleted from the event table after the connector picks them up, they occupy unnecessary space there. However, if they are deleted, all events that are not processed are lost and you cannot audit the event processing. Therefore, you should also create an archive table and keep the ArchiveProcessed property set to true. Whenever an event is deleted from the event table, the connector inserts it into the archive table.

Configuring event and archive processing

To configure event and archive processing, you must use configuration properties to specify the following information:

- The interval frequency
- The number of events for each polling interval
- Whether the connector archives unsubscribed and unprocessed events

- The unique ID of the connector, which is important when multiple connectors poll the same table

Creating the event and archive tables

This procedure uses the Siebel Sales Enterprise application as an example. Substitute all references to Siebel Sales Enterprise with the name of the Siebel application in use.

To create the event and archive tables and to trigger the business objects, perform the following procedure:

1. Ensure that all current projects have been checked in.
2. On your local database, check out and lock the following files:
 - New Table Project

Note: New Table Project does not apply to Siebel, version 7.5.

- Siebel Sales Enterprise project
- Projects that include objects that you want to modify, such as the Account project
- Dock project

Note: Ensure that the projects are locked on both the local and development servers.

3. Apply the seven patch files in the following order to your local database:
 - cwttable.sif
 - cwview.sif
 - cwapplet.sif
 - cwbo.sif
 - cwbc.sif
 - cwdo.sif
 - cwscreen.sif
 - schemabo.sif

Note: There is no schemabo.sif file for Siebel, version 7.5.

When you apply WebSphere business integration system patch files in a Japanese environment, edit all the patch files as follows:

Edit the first line of each file from:

```
<xml version="1.0" encoding="windows-1252"?>
```

to:

```
<xml version="1.0" encoding="Shift_JPN"?>
```

Replace all instances of the “ENU” language setting with “JPN.” If you use the search and replace function of your text editor, make sure you use quotation marks around the language setting to make sure no similar words (for example, MENU) are replaced.

4. When you are prompted, lock the CW Audit project on your local database.
5. Ensure that the following have been created:

- Two new tables, CX_CW_Archive_Q and CX_CW_Event_Q
 - One new business object, CW Events
 - One new business object, schema version
 - Two new business components, CW Archive and CW Events
 - One new view, CW Event List View
 - Two new applets, CW Archive List Applet and CW Event List Applet
 - One new screen CW Events and one new screen view, CW Event List view
 - Two new dock objects, CX_CWArchive and CX_CWEvent
6. Create a page tab as follows:
 - a. Access the Application > Siebel Sales Enterprise > Page tab.
 - b. Right-click and select New Record from the menu.
 - c. Enter CW Events as the screen name and IBM Events as the text name.
 - d. For the sequence, enter a number greater than the rest of the sequence numbers. This selection determines where the tab is displayed in the application.
 - e. Leave the inactive field unchecked.
 - If you are using Siebel 6, proceed to Step 7.
 - If you are using Siebel 7, go to the Page tab locale and create a new record for CW Events. Add ENU for the Language Code and IBMEvents for text, if it does not exist.
 7. Create a screen menu item as follows:
 - a. Access the Application > Siebel Sales Enterprise > Screen Menu Item.
 - b. Right-click and select New Record.
 - c. Enter CW Events as the screen and IBM Events as the text name.
 - d. For the sequence, enter a number greater than the rest of the sequence numbers. This selection determines where the tab is displayed in the screen pull-down menu.
 - e. Leave the inactive field unchecked.
 - If you are using Siebel 6, proceed to Step 8.
 - If you are using Siebel 7, go to the screen menu item locale and create a new record for CWEvents. Add ENU for language and IBMEvents for text, if it does not exist.
 8. Add or modify the Siebel VB scripts for the business components that correspond to the business objects used at your site. The Siebel VB scripts trigger event notification for business objects.
 - If you want to sort events by priority, edit the priority values in the business objects VB scripts before compiling them.
 - If you are installing is multiple connectors, set and activate the Connector Id in the VB scripts.
- Siebel 6**
- If you want to use the Additional Object Key field, you must set it in the VB script.
9. Apply the physical schema for the new tables to your local database. You can do this by querying for the two new tables, CX_CW_ARCHIVE_Q and CX_CW_EVENT_Q, and selecting the current query to create a physical schema. Make sure that you leave the table space and index space blank.
 10. Activate the new schema using the activate button.

11. Compile the updated and locked projects on your local database to create a new Siebel repository (.srf) file.
12. Open Siebel Sales Enterprise on your local database. You must have administrative privileges to perform the following:
 - a. .Create a new view called CW Event List View. Tip: Copy the view name from tools and paste it into the View Name field.
 - b. .Create a new responsibility called CW Responsibility for CW Event List View.
 - c. .Add the employees or teams who are responsible for reviewing events to the newly created CW Responsibility.
 - d. .Create the CWCONN user and add it to CW Responsibility and Administrative Responsibility.
13. Test the application in your local environment. Ensure that you have visibility to CW Event List View and that an event is generated in the view after you create a supported object. For example, create a new account in Siebel and check that a new account event appears in the CW Event List View.
14. Check in the following updated and locked projects to your development server.
 - New Table
 - CW Audit
 - Dock
 - Siebel Sales Enterprise
 - The project for the business objects that you want to use

Note: You should check in your locked projects only through the query.

15. Apply the physical schema to your development database. You can do this by querying for the two new tables, CX_CW_ARCHIVE_Q and CX_CW_EVENT_Q, and select the current query to create a physical schema. Make sure that you leave the table space and index space blank.
16. Activate the queried tables in the development database.
17. Move to test and production environments accordingly.
18. Move your newly compiled Siebel.srf file to the server.

Note: Enable Enterprise Application Integration by going to:
 Sitemap > Server Administration > Component Group and selecting Enable.

To set Siebel JAVABean:

1. Select, Site Map->Server Admin-> Components (Sales Object Manager).
2. In the lower applet, go to Component Parameter and enter a timeout value.

Note: The Request Timeout current value is set to 600. This means that the connector will die after ten minutes. Based on Siebel, you can change this value to be as large as you want.

Configuring the connector

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

You configure connector properties from Connector Configurator (when WebSphere MQ Integrator Broker is the integration broker) or from Connector Configurator, which is accessed from System Manager (when ICS is the integration broker). For detailed configuration information, see Appendix B, “Connector Configurator,” on page 105 or the *Connector Development Guide for Java*.

Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, “Standard configuration properties for connectors,” on page 89 for documentation of these properties.

Table 11 provides information specific to this connector about configuration properties in the appendix.

Table 11. Property information specific to this connector

Property	Note
Character Encoding	This connector does not use this property.
Locale	Because this connector has been internationalized, you can change the value of this property. See release notes for the connector to determine currently supported locales.

Important: WebSphere MQ Integrator Broker does not support multiple locales. Ensure that every component of your installation (for example, all adapters, applications, and the integration broker itself), are set to the same locale.

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

Connector-specific properties

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

Table 12 lists the connector-specific configuration properties for the connector. See the sections that follow for explanations of the properties.

Table 12. Connector-specific configuration properties

Name	Meaning	Possible values	Default value
ConnectString	With the new Siebel architecture, the Siebel Java Data Bean can connect to the Siebel Object Manager using the connect string. There is no need to know the configuration file, simply read the file and connect to the Object Manager.	protocol://machinename/enterprisename/objectmanager/servername	N/A

Table 12. Connector-specific configuration properties (continued)

Name	Meaning	Possible values	Default value
ConnectString (for Siebel, version 7.5)	With the new Siebel architecture, the Siebel Java Data Bean can connect to the Siebel Object Manager using the connect string. There is no need to know the configuration file, simply read the file and connect to the Object Manager.	protocol://machinename/enterprisename/objectmanager_languageCode/ servername	N/A
ConnectString (for ODA, version 7.5)	With the new Siebel architecture, the Siebel Java Data Bean can connect to the Siebel Object Manager using the connect string. There is no need to know the configuration file, simply read the file and connect to the Object Manager.	siebel.tcpip.none.none://CW_ECOM2:2320/siebel/SSEObjMgr_Enu/ CW_ECOM2	N/A
ApplicationUserName	User account for the Siebel application		CWCONN
ApplicationPassword	Password for the Siebel user account		CWCONN
ConnectErrors	A set of errors returned from Siebel which are checked in the connector. These errors are considered to be fatal, and the connector is terminated when it encounters these errors.	Any network failure or connectivity failure messages. These messages are separated by the ';' delimiter.	
ConnectorId	Used in case the system has been configured to handle multiple connectors.	An integer value denoting the number of connectors configured.	
ViewMode	Retained for backward compatibility. The value is used if there is not VM asi specified for the business component.	An integer value. Refer to VM asi for details.	
DataBeanRefreshInterval	The value is used to refresh the Siebel data bean resources when the connector is running against Siebel 6.2.x The connector logs off after the requests processed are equal to this value and logs back in.	An integer value indicating the DataBeanRefreshInterval which corresponds to the number of requests to be processed by the connector before a refresh call.	
DataBeanPoolSize	Indicates the maximum number of beans in the data bean pool.	An integer determining the bean pool size.	

Table 12. Connector-specific configuration properties (continued)

Name	Meaning	Possible values	Default value
SiebelLanguageCode	Three letter NLS character set code used by Siebel for the languages supported. Default is US English with ENU as the NLS representation.	With Siebel 7, the languages supported with their language codes is listed below: Italian (Std) -- ITA Japanese -- JPN Korean --KOR Norwegian -NOR (Bokmal) Polish -- POL Portuguese -- PTB (Brazil) Portuguese -- PTG (Portugal) Russian - RUS Spanish -- ESN (Modern Sort) Swedish -- SVE Turkish -- TUR English (US) -- ENU + all the other languages supported by NLS	
DateFormat	String date format for the specified locale.	String representing the date format. Ex: MM/DD/YY HH:MM:SS AM/PM for US.	
PollQuantity	Determines the number of events that gets processed with a pollForEvents call.	Integer representing the number of events that gets processed with a pollForEvents call.	1
SupportNameValuePairs	Used for determining the event object key format. If not set or if set to true, the object key value needs to be a name-value pair with an "=" between the name and the value. If set to false, only one rowId can be specified.	True or False	False
PollAttributeDelimiter	In case of multiple name-value pairs in the object key, this value determines the delimiter between the keys. If not set, the default is; (semi-colon).	Character	;
ResonateSupport	Indicates if Resonate has been installed with the Siebel server. The connector bean pool uses Attach/Detach calls (Siebel7) only if Resonate is installed. If not, it logs off after processing a certain number of requests.	Boolean(Logoff from the bean is decided by the DataBeanRefreshInterval) setting.	false

ApplicationPassword

Password for the application user account.

There is no default value.

ApplicationUserName

Name of the application user account.

There is no default value.

ArchiveProcessed

Specifies whether the connector archives events for which there are no current subscriptions.

Set this property to true to cause events to be inserted into the CW Archive business component after they are deleted from the CW Event business component.

Set this property to false to cause the connector not to perform archive processing. If `ArchiveProcessed` is set to false, the connector behaves as follows:

- If the event is successfully processed, the connector deletes it from the CW Event business component.
- If the connector does not subscribe to the event's business object, the connector leaves the event in the CW Event business component and changes its event status to Unsubscribed.
- If the business object encounters a problem while being processed, the connector leaves the event in the event table with event status set to that of error.

If this property is set to false and the poll quantity is low, the connector appears to be polling the event table, but it is simply picking up the same events repeatedly.

If this property has no value, the connector assumes the value to be true. If the `ArchiveTableName` property also has no value, the connector assumes the archive table's name is `xworlds_archive_events`.

The default value is true.

ConnectorID

A unique ID for the connector. This ID is useful to retrieve events for a particular instance of the connector.

Default value is null.

PollQuantity

Number of rows in the database table that the connector retrieves per polling interval. Allowable values are 1 to 500.

The default is 1.

UseDefaults

If `UseDefaults` is set to true or is not set, the connector checks whether a valid value or a default value is provided for each required business object attribute. If a value is provided, the Create succeeds; otherwise, it fails.

If `UseDefaults` is set to false, the connector checks only whether a valid value is provided for each required business object attribute; the Create operation fails if a valid value is not provided.

The default value is false.

Connector startup

For information on starting a connector, stopping a connector, and the connector's temporary startup log file, see the *IBM WebSphere Business Integration Adapter Implementation Guide for WebSphere MQ Integrator Broker*, or, for ICS, see the *IBM WebSphere InterChange Server System Administration Guide*.

Chapter 3. Understanding business objects

This chapter describes how the connector processes business objects and provides suggestions for modifying them.

To properly create or modify business objects for Siebel, you must understand the object relationships within the Siebel architecture.

The following topics are covered:

- “Business object structure and relationships”
- “Business object application-specific information” on page 43
- “Using business object samples” on page 52

Business object structure and relationships

The connector supports Create, Retrieve, Update, Delete, Exists, Retrieve By Content, and DetlaUpdate verbs for a Siebel application-specific business object whose semantics are encapsulated in its business object definition. There is no connector logic that processes a Siebel application-specific business object according to hard-coded instructions in the connector. For example, the connector does not expect a particular business object to consist of a certain type and number of entities. What the connector expects is that any object may consist of one or more entities.

Siebel business components can be associated in three ways. They can be linked in one-to-one relationships through single-valued links, or they can have Multi-Value Link (MVL) fields representing one-to-many relationships, or they can have a simple link.

Business components can be associated in many-to-one relationships by means of PickLists. Business component methods provide support for searching a PickList business component for a specific value and placing that value in a field. Finally, business components can be associated in many-to-many relationships through intersection tables.

In order to support the Siebel concept of a business object context encapsulating numerous business components, a top-level business object should be corresponded to the appropriate Siebel business object. The top-level business object application-specific information should contain the name of the corresponding Siebel business object. Each top-level attribute should then correspond to a Siebel business component.

Within a business object definition that corresponds to a business component, each attribute specifies either a simple field, or a Multi-Value Group (MVG) field. The attribute data in simple attributes should have simple data types. Attributes that correspond to MVG fields should be treated as child (container) business objects.

If there are two unrelated single cardinality business components under the same business object in Siebel, a separate business object wrapper needs to be created.

This business object structure is part of the meta-data that allows the connector to handle all business objects in the same manner. The connector can support additional Siebel objects if a business object definition is specified for the object.

Specifying key attributes

When developing a Siebel business object, always place the key attribute at the top of the object. This ensures that the connector has the key value before processing the rest of the object. Placing the key attribute elsewhere in the object may lead to processing errors. The key attribute for an object is its RowId in Siebel.

Note: The connector does not support specifying an attribute that represents a child business object or an array of child business objects as a key attribute, except for the child of a top-level business object (the Siebel BO).

Note: When developing business objects for the connector, you must ensure that there is a 1-to-1 correspondence between the business object and the Siebel business component.

Attribute properties

The following tables describe simple attributes and child object attributes.

Table 13. Simple attribute

Name	Name of attribute
Type	Data Type of the attribute. Currently, this is not used, but for forward compatibility reasons, SiebelODA sets the type to either boolean, String, Date, int, double. All types are treated as Strings.
MaxLength	Applies to String types and represents the maximum length allowed for the attribute. This is not used by the connector. If the data is large, it must be handled in the business processes.
IsKey	If set, this denotes that the attribute is a key. It is used with Update to update a specific record in Siebel. With Retrieve, these attributes are used in the search specification to get the record from Siebel. During Delete, The keys are set on the top-level business components.
IsForeignKey	Not used.
IsRequired	Set to true if the attribute for the fields in the Siebel business component whose "Required" property is checked.
AppSpecificInfo	Text comprised of information about communicating with the application and getting the Siebel business objects and business components associated with this business object.
DefaultValue	If set for the attribute, this value is used by the connector if one is not set in the inbound business object and the connector property UseDefaults is set to True.

Table 14. Child object attributes

Name	Name of the child object
Type	Business object type for the child.
ContainedObjectVersion	The child business object version
Relationship	If the child is a container attribute, this is set to Containment.
IsKey	This attribute has to be set on the primary business component.
IsForeignKey	Not used.
IsRequired	If set to True, the child is expected to have a representation in the parent business object. During Create verb processing, the primary business component is required to be present. A check is made to see if this component is present in the inbound business object. If found, the create proceeds unless an error is thrown indicating that the required object was not found in the inbound business object.
Cardinality	1 or N depending on the number of child records that can be chosen for a parent record.

Business object application-specific information

Application-specific information in business object definitions provides the connector with application-dependent instructions on how to process business objects. Because a meta-data-driven connector makes assumptions about how its supported business objects are designed, modifications to business objects must match the connector's rules for the connector to process modified business objects correctly. Therefore, if you modify or create Siebel application-specific business objects, you must be sure that the application-specific information in the business object definition matches the syntax that the connector expects.

This section describes the application-specific information for the Siebel business object, attributes, and verbs.

Business object application-specific information

The application-specific information at the top level of a business object specifies the name of the Siebel business object. For example, the object application-specific information for the parent business object Siebel_BCAccount specifies the Siebel Account object, as shown below.

```
[BusinessObjectDefinition]
Name = Siebel_BCAccount
Version = 1.0.0
AppSpecificInfo = ON=Account;CN=Account
```

Example of multiple unrelated business components:

```
[BusinessObjectDefinition]
Name = Siebel_BCInternalProduct
Version = 1.0.0
AppSpecificInfo = CN=InternalProduct
```

```
[BusinessObjectDefinition]
Name = Siebel_BCProductDefect
Version = 1.0.0
```

```

AppSpecificInfo = CN=ProductDefect

[BusinessObjectDefinition]
Name = Siebel_BOInternalProduct
Version = 1.0.0
AppSpecificInfo = ON=InternalProduct

[Attribute]
Name = Siebel_BCInternalProduct
Type = Siebel_BCInternalProduct
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =

[Attribute]
Name = Siebel_BCProductDefect
Type = Siebel_BCProductDefect
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =

```

Attribute application-specific information format

The connector uses application-specific information for simple attributes and container attributes. The application-specific property field must be empty for the ObjectEventId attribute.

Application-specific information for simple attributes

For simple attributes, the application-specific information consists of the name-value pairs listed in the following table. The name-value pairs are order-independent and are delimited by semicolons.

Parameter	Description
FN =	The name of the corresponding field in the Siebel business component.
PLK = ...;Restrict=<field name>:Siebel GUI Name>,<field name>:Siebel GUI Name	Business components in Siebel can be associated with a many-to-one relationship using PickLists. The PickList can be searched, and a specific value can be chosen to be placed in a field. The PickListKey is set in case there is a PickList associated with the field in Siebel. The PickListKey identifies the field in the PickList that is used in the search specification to obtain the PickList record. In some cases, PickList values are chosen based on more than one attribute. In such cases, the PickList can be restricted by more than one field.

Example of the use of these parameters are provided in the sections that follow.

Field names for a simple attributes

application-specific information for simple business objects attributes must specify the name of the corresponding field in the Siebel business component. The application-specific information for this is:

```
FN=fieldname
```

For example, in the Siebel_BCAccount business object the application-specific information for the Main Phone attribute specifies that Main Phone Number is the corresponding field in the Siebel Account business component. The application-specific information in the business object attribute is shown below.

```
Name = Main Phone  
Type = String  
IsKey = false  
AppSpecificInfo = FN=Main Phone Number
```

Foreign key relationship using a pickList

In Siebel, a foreign key relationship between two business components is defined by a PickList. If a field has an associated PickList, the field's PickList property and PickList correspondence define the relationship between the two business components. One of the attributes in the PickList correspondence is usually an Id, such as Account Id or Product Id.

On a simple attribute in a business object, if a Siebel business component field has an associated PickList, the attribute application-specific information in the business object should be coded to provide the connector with this information so that the connector can use the attribute as a foreign key.

To specify a PickList for an attribute, you need to include two attributes in the business object. The first attribute identifies the foreign key field of the related business component, and the second attribute corresponds to the field in the business component that has the PickList as a field property. Two attributes are required because the PickList relationship is based on the object name rather than the object Id.

In the application-specific information for the PickList attribute, specify that this attribute is a PickList using the text PLK. Then, to identify which record in the PickList should be selected, use the text PLK=...;Restrict=<field name>:<Siebel GUI Name>,<field name>:Siebel GUI Name>.

For example, suppose that you are creating a Siebel_BCAsset business object, and you want to add an attribute in the business object as a foreign key to the Siebel_BCInternalProduct business object. The Product Name field in the Siebel Asset Mgmt business component is a PickList to the Internal Product business component, so you add an attribute for the key and another attribute for the PickList. The attributes might be defined in the business object as shown below.

```
[Key Attribute]  
Name = Id  
Type = String  
Cardinality = 1  
IsForeignKey = true  
AppSpecificInfo = Product Id
```

```
[PickList Attribute]  
Name = ProductName  
Type = string  
Cardinality = n  
AppSpecificInfo = FN=ProductName;PLK=Id
```

In some cases, PickList values are chosen based on more than one attribute. For example, where there is more than one Account with the same name, a Contact retrieve will get the first Account with that name if Account name is set as the only PickList value. To ensure that the correct data is retrieved, you can restrict the PickList by more than one field. In the following example, the Contact business object is restricted by Account, Site and City:

```
[Key Attribute]
Name = ContactId
Type = String
Cardinality = 1
AppSpecificInfo = FN=Id

Name = Last_Name
Type = String
Cardinality = 1
AppSpecificInfo = FN=Last Name

Name = First_Name
Type = String
Cardinality = 1
AppSpecificInfo = FN=First Name

Name = Site
Type = String
Cardinality = 1
AppSpecificInfo = N/A

Name = City
Type = String
Cardinality = 1
AppSpecificInfo = N/A

Name = Account
Type = String
Cardinality = 1
AppSpecificInfo = FN=Account;PLK=Name;Restrict=Location:Site, City:City
```

The AppSpecificInfo for restricting PickList fields follows this syntax:

```
Restrict=<field name>:<Siebel GUI name>,<field name>:<Siebel GUI name>
```

There is no limit to the number of restricting fields. Do not use spaces between the attributes after the Restrict parameter. All restricting fields must be added as attributes to the business object, and should have no AppSpecificInfo. These attributes serve as place holders for the restricting fields.

On a Retrieve, the application-specific information PLK=Id specifies that the ProductName attribute corresponds to a PickList business component, and the set parameter specifies that the value of the Id identifies which record the connector should pick.

Some PickList relationships require the creation of a picked child, for example, the PickList relationship between Account and Quote in Siebel_BCQuote. On a Create, you must create a new Account business component for the PickList with a containment relationship to the Quote business component as follows:

```
[Business Object Definition]
Name = Siebel_BCQuote
Version = 7.0.0
Relationship = Containment
AppSpecificInfo = ON=Quote;CN=Quote

Name = Account
```

```

Type = String
AppSpecificInfo = FN=AccountId

Name = Account
Type = Siebel_BCAccount
Relationship = Containment
Cardinality = 1
IsForeignKey = false
AppSpecificInfo = LFN=Account;PL=true;From=AccountId;To=AccountId

[Siebel_BCAccount]
Name = AccountId
IsKey = true
AppSpecificInfo = ...

```

Note the following business processing tips for PickList attributes:

- On a Retrieve operation, correspond the value of the PickList attribute to the name of the PickList business component, and correspond the value of the key attribute to the key.
- On a Create or Update operation, correspond the PickList attribute to the key, and correspond the key attribute to a null value. Because the PickList link is defined on the name of a field, the connector could set the key attribute value to any value, and Siebel does not validate the value. If the PickList attribute contains the key value, and a pick operation is performed using the PickList component, validation of the key is performed. If the Pick operation finds the field, it adds all the attributes in the pick correspondence to the new object, and the new object is created.
- To remove the link from the PickList, correspond the value for the PickList attribute to null and correspond the value for the key attribute to blank.

Application-specific information for container attributes

For container attributes, the application-specific information contains the name-value pairs listed in the following table. The name-value pairs are order independent and are separated by semicolons.

Parameter	Description
LFN = ...;	Multi Value Field Name related to the Siebel business component.
MVL = ...;	When MVL is set to Active, it specifies a one-to-many relationship. Setting MVL to Inactive indicates that there is an inactive multi-value link relationship between the parent and child objects, which means that the parent object does not have a multi-value field.
PL = ...	When PL is set to True, it indicates that there is a many-to-one relationship.
Assoc = ...	When Assoc is set to True, it indicates that a relationship is many-to-many through an intersection table.
From = ...; To = ...	These are preprocessing instructions to the connector to set the To attribute to the value of the From attribute. The From attribute must be populated, while the True attribute is set only if it is null. The objects containing the attributes must have a one-to-one relationship. This is used in a Retrieve operation and to specify which child record needs to be fetched.
SF	Simple link which provides the master detail view of the business component structure under a business object. SF represents the source field.
DF	Simple link which provides the master detail view of the business component structure under a business object. DF represents the destination field (foreign key).

Field names for container attributes

application-specific information for a container attribute referencing a child business object must specify the name of the Multi-Value Field related to the parent business component. A Multi-Value Field represents the Multi-Value Link that defines the relationship between the parent and child business components in Siebel. The application-specific information for this is:

```
LFN=multiValueFieldName
```

For example, the WebSphere business integration system business object Siebel_BCContact has a container attribute for the child business object Siebel_BCBusinessAddress. The application-specific information for this container attribute specifies Street Address as the Multi-Value Field that contains the link to the Siebel Business Address business component.

```
[Example of Container Attribute]
Name = PrimaryAddress
Type = Siebel_BCBusinessAddress
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
IsKey = false
IsForeignKey = false
AppSpecificInfo = MVL=Active;LFN=Street Address
```

Relationships between parent and child business objects

In addition to the field name, the application-specific information for a container attribute can include a parameter that defines the type of relationship between the parent and child business components in Siebel.

Setting a relationship type parameter to 0 is not valid. To set a relationship type to false, do not include the parameter.

As an example, the container attribute for the child business object Siebel_BCBusinessAddress, shown in the previous section, might include the parameter to indicate that Street Address is a Multi Value Field that links the Siebel Account business component to the Siebel Business Address business component.

```
[Example of Container Attribute]
Name = PrimaryAddress
Type = Siebel_BCBusinessAddress
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
IsKey = false
IsForeignKey = false
AppSpecificInfo = MVL=Active;LFN=Street Address
```

Another example for the child business object Siebel_BCOpportunity is shown here which has a many-to-many relationship to Siebel_BCContact or Association. In this case, on a Create operation, the connector searches for the business component using the populated fields of the business object in the container. If the connector finds a matching object, it associates it with the parent business component. If the object is not found, an error is logged, and the business object request fails.

```
Name = Siebel_BCContact
Version = 1.0.0
AppSpecificInfo = ON=Contact;CN=Contact

Name = ContactId
```

```

Type = String
Cardinality = 1
MaxLength = 10
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = FN=Id

Name = ...

Name = ...

Name = Siebel_BCOppportunity
Type = Siebel_BCOppportunity
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = N
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = LFN=Oppportunity;Assoc=true

```

This example is for a simple link relationship. In this case, there is a simple link between Quote and Order Entry:

```

Name = Siebel_BCQuote
Version = 1.0.0
AppSpecificInfo = ON=Quote;CN=Quote
Name = QuoteId
Type = String
Cardinality = 1
MaxLength = 10
IsKey = true
IsFireignKey = false
IsRequired - false
AppspecificInfo = FN=Id

Name = ...

Name = ...

Name = Siebel_BCOrderEntry
Type = Siebel_BCOrderEntry
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = N
MaxLength = 0
IsKey = true
IsFireignKey = false
IsRequired - false
AppspecificInfo = SF=QuoteId;DF=QuoteId

```

Assigning the value of an attribute to another attribute

Attribute application-specific information can be coded so that the connector obtains a value for an attribute and assigns it to another attribute before the second attribute is processed. This functionality is used in a Retrieve operation and is primarily used on container attributes to specify which record for the child should be retrieved.

To use this functionality, edit the attribute application-specific information in the business object definition to include the following text:

```
From=attribute;To=attribute;
```

The attribute path value can be an attribute name in the current business object. Note the following rules:

- The From attribute is an attribute from the parent, and the To attribute is the child attribute.
- The From attribute must be populated before the to attribute in the instruction can be processed.
- The To attribute is set only if it has a null value.
- If the path is invalid for the from parameter, the to parameter is set to null. If the path is invalid for the to parameter, no error is flagged.
- The From/To directive can be specified only in the application-specific information of an attribute on a child business object. In other words, it cannot be specified on a top-level business object.

For example, if a Siebel_BCQuote business object includes a child business object Siebel_BCAccount, attributes in the Siebel_BCQuote object can specify which address from the PickList is retrieved. In this example, AccountId is the key attribute, and Siebel_BCAccount is the picked object. The connector gets the value of the AccountId attribute, and uses that value to retrieve the specific account. The child attributes are processed after the attributes in the parent business object. The following example shows the processing flow of attributes from parent to child business objects.

```
[Siebel_BCQuote]
Name = Account
Type = String
AppSpecificInfo = FN=Id
```

```
Name = Account
Type = Siebel_BCAccount
Relationship = Containment
Cardinality = 1
IsForeignKey = false
AppSpecificInfo = LFN=Address;PL=true;From=AccountId;To=AccountId
```

```
[Siebel_BCAccount]
Name = AccountId
IsKey = true
AppSpecificInfo = ...
```

Specifying pickList relationships

Some PickList relationships require the creation of the picked child object in the same transaction. In WebSphere business integration system business objects, a PickList relationship between parent and child business objects is represented by two attributes: a key attribute and a single cardinality container attribute for the picked object. This set of attributes can be used to retrieve some or all of the attributes of the PickList business component that are not included in the PickList map.

For example, the Siebel_BCQuote business object might be designed to include two attributes to specify a PickList relationship between Quote and Opportunity. As shown below, OpportunityId is the key attribute, and Opportunity is the PickList object.

```
Name = Siebel_BCQuote
Version = 1.0.0
AppSpecificInfo = ON=Quote;CN=Quote
```

```
Name = ...
```

```
Name = ...
```

```
Name = OpportunityId
Type = String
Cardinality = 1
MaxLength = 10
IsKey = false
IsForeignKey = true
IsRequired = false
AppSpecificInfo = FN= OpportunityId
```

```
Name = Siebel_BCOppportunity
Type = Siebel_BCOppportunity
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = LFN=Opportunity;PL=true;From=OpportunityId;To=OpportunityId
```

In the application-specific information, PL=true indicates that the container attribute represents a PickList, the From= parameter is the pointer to the key attribute, and the To= parameter points to the key attribute of the Siebel_BCOpprotunity business object.

The order of the attributes is significant in a Retrieve operation, because the value of OpportunityId must be retrieved before it can be defined as the foreign key in the child object. On a Create or Update operation, the value of OpprotunityId is a foreign key and is retrieved after the object is created.

It is not necessary to use a complete business object as a PickList container. An object with only the required keys set is sufficient. The connector uses the following rules for processing a PickList container:

- If none of the key attributes of the PickList business object are set, a new object is created in Siebel and picked.
- If any of the key attributes of the PickList business object are set, the connector searches for the object and picks it. If a PickList object for that business object is not found, the connector logs an error. An error might occur when the object keys are not valid.

The following are guidelines for maps for PickList attributes on container business objects:

- Mapping of the key attribute when it is a business object request from the collaboration to the connector should follow the same guidelines for simple attributes as described above.
- Mapping of the container attribute should be keys only if keys are known.
- If the PickList object will be created, map all the required attributes as specified for the PickList object.
- On a Delete operation, set the key attribute to a space and the PickList container attribute to null.

Verb application-specific information format

Application-specific information for a business object Retrieve verb can specify that the connector retrieve a limited number of objects on each retrieve. The application-specific information to retrieve a subset of objects is max=n. An example of a Retrieve verb that specifies that only five objects are retrieved is:

[Verb]
Name = Retrieve
AppSpecificInfo = max=5

For other verbs, the application-specific property is not used and should be left blank or omitted when creating business object definitions.

Key attribute for create and update verbs

On a Create or Update request, if the Object Key value is different from RowId, the Siebel application blanks out the Object Key attribute and creates its own RowId for that record.

Important: You must use RowId as the key attribute in Create and Update requests.

Using business object samples

Business object samples are provided with the connector component of the adapter. All objects must have Siebel triggers on them for polling. In some cases, you may need to customize the object in Siebel Tools. The following objects are examples of how to do these tasks:

- “Siebel_BCAccount” on page 52
- “Siebel_BCQuote” on page 53
- “Siebel_BCContact” on page 54
- “Siebel_BCInternalProduct” on page 54
- “Siebel_BCAsset” on page 55

Siebel_BCAccount

1. Locate the following Siebel VB files:

Siebel 6

Account_Write.svb, Account_PreDelete.svb, Business_Address_PreDelete.sbv, and Business_Address_Write.svb.

Siebel 7

Account.sbl or Account.js

The Siebel VB files are located in the Common/Siebel/Dependencies/Siebel_VB directory.

2. Save the files in a temporary directory. For example, use Temp as the directory name.
3. Give the CWCONN account the System Administrator responsibility within Siebel.
4. In your Siebel Tools environment, check out and lock the Account and Contact projects.
5. Add the VB script to the Account business component as follows:
 - a. Right-click on the Account business component, and select Edit Basic Scripts from the menu. This launches the script editor.
 - b. Import the VB code by selecting Import from the File menu, choosing the Temp directory, and picking Account_Write.svb for import into the Bus Comp object and BusComp_WriteRecord procedure.

- c. Import the second VB code by selecting Import from the File menu, choosing the Temp directory, and picking Account_PreDelete.svb for import into the Bus Comp object and BusComp_PreDeleteRecord procedure.
 - d. Save changes to the object.
6. Add the script to the Business Address business component as follows:
 - a. Right-click on the Business Address business component, and select Edit Basic Scripts from the menu. This launches the script editor.
 - b. Import the VB code by selecting import from the File menu, choosing the Temp directory, and picking Business_Address_Write.svb, for import into the Bus Comp object and BusComp_WriteRecord procedure.
 - c. Import the second VB code by selecting import from the File menu, choosing the Temp directory, and picking Business_Address_PreDelete.svb for import into the Bus Comp object and BusComp_PreDeleteRecord procedure.
 - d. Save changes to the object.
 - e. In the Business Address business component for the Account Id attribute, set the Force Active field to TRUE.

Note: Sometime this field does not get populated because of the view being used.
 7. You will not be able to test the object until you have saved all objects and compiled the result.

Siebel_BCQuote

1. Locate the following Siebel VB files:

Siebel 6

Quote_Write.svb, Contact_PreDelete.svb

Siebel 7

Quote.sbl, Quote.js

The Siebel VB files are located in the Common/Siebel/Dependencies/Siebel_VB directory.

2. Save the file in a temporary directory. For example, use Temp as the directory name.
3. Give the CWCONN account the System Administrator responsibility within Siebel.
4. In your Siebel Tools environment, check out and lock the Quote project.
5. Add the script to the Quote business component as follows:
 - a. Right-click on the Quote business component, and select Edit Basic Scripts from the menu. This launches the script editor.
 - b. Import the VB code by selecting Import from the File menu, choosing the Temp directory, and picking Quote_Write.svb for import into the Bus Comp object and the BusComp_WriteRecord procedure.
 - c. Save changes to the object.
6. You will not be able to test the object until you have saved all objects and compiled the result.

Siebel_BCContact

1. Locate the following Siebel VB files:

Siebel 6

Contact_Write.svb, Contact_PreDelete.svb.

Siebel 7

Contact.sbl, Contact.js

The Siebel VB files are located in the Common/Siebel/Dependencies/Siebel_VB directory.

2. Save the Siebel VB file in a temporary directory. For example, use Temp as the directory name.
3. Give the CWCONN account the System Administrator responsibility within Siebel.
4. In your Siebel Tools environment, check out and lock the Contact project.
5. Add the Siebel VB script to the Contact business component as follows:
 - a. Right-click on the Contact business component, and select Edit BasicScripts from the menu. This launches the script editor.
 - b. Import the VB code by selecting Import from the File menu, choosing the Temp directory, and picking Contact_Write.svb for import into the Bus Comp object and BusComp_WriteRecord procedure.
 - c. Import the second VB code by selecting Import from the File menu, choosing the Temp directory, and picking Contact_PreDelete.svb for import into the Bus Comp object and BusComp_PreDeleteRecord procedure.
 - d. Save changes to the object.
6. You will not be able to test the object until you have saved all objects and compiled the result.

Siebel_BCInternalProduct

1. Locate the file InternalProduct_Write.svb. The Siebel VB files are located in the Common/Siebel/Dependencies/Siebel_VB directory.
2. Save the file in a temporary directory. For example, use Temp as the directory name.
3. Give the CWCONN account the System Administrator responsibility within Siebel.
4. In your Siebel Tools environment, check out and lock the Product project.
5. Add the Siebel VB script to the Internal Product business component.
 - a. Right-click on the Internal Product business component, and select Edit Basic Scripts from the menu. This launches the script editor.
 - b. Import the VB code by selecting Import from the File menu, choosing the Temp directory, and picking InternalProduct_Write.svb for import into the Bus Comp object and BusComp_WriteRecord procedure.
 - c. Save changes to the object.
6. Change the business component properties as follows:
 - a. In the tool bar, select View > Property Window.
 - b. Go to Business Component/Internal Product.
 - c. Change the following attributes to False:

No Insert = False
No Merge = False
No Update = False

Note: The purpose of changing the above properties is to allow Com Data Server Interface to create and update products inbound to Siebel.

7. You will not be able to test the object until you have saved all objects and compiled the result.

Siebel_BCAsset

1. Locate the Siebel VB file Asset_Write.svb. The Siebel VB files are located in the Common/Siebel/Dependencies/Siebel_VB directory.
2. Save the Siebel VB file in a temporary directory. For example, use Temp as the directory name.
3. Give the CWCONN account the System Administrator responsibility within Siebel.
4. In your Siebel Tools environment, check out and lock the Asset Management project.
5. Add the Siebel VB script to the Asset Mgmt - Asset business component as follows:
 - a. Right-click on the Asset Mgmt - Asset business component, and select the Edit Basic Scripts from the menu. This launches the script editor.
 - b. Import the Write VB code by selecting Import from the File menu, choosing the Temp directory, and picking Asset_Write.svb into Bus Comp object and BusComp_WriteRecord procedure.
 - c. Save changes to the object.
6. Change the business component properties as follows:
 - a. In the tool bar, select View > Property Windows.
 - b. Go to Business Component/Asset Mgmt - Asset.
 - c. Go to fields.
 - d. Change the field property values as follows:
 - Select Account Id and set the value for Inactive to False. (This field is required in the WebSphere business integration system object.)
 - Select Name and set the value to Not Required.
 - Select Product Id and set the value to Required.

Note: WebSphere business integration system tracks the Products by their Product Id and not their Name.

7. You will not be able to test the object until you have saved all objects and compiled the result.

Chapter 4. Generating business object definitions using SiebelODA

This chapter describes SiebelODA, an object discovery agent (ODA) that generates business object definitions for the connector. SiebelODA uses the Siebel Java APIs to get the information about the Siebel business objects and business components from the Siebel application server. It then uses this information to build new business object definitions. SiebelODA also enables the conversion of existing business object definitions to those which are supported by the connector.

The following topics are covered:

- “Installation and usage”
- “Using SiebelODA in Business Object Designer” on page 60
- “Contents of the generated definition” on page 72
- “Adding information to the business object definition” on page 74

Installation and usage

This section discusses the following:

- “Installing SiebelODA” on page 57
- “Before using SiebelODA” on page 58
- “Launching the SiebelODA” on page 58
- “Running SiebelODA on multiple machines” on page 59
- “Working with error and trace message files” on page 59

Installing SiebelODA

To install SiebelODA, use the WebSphere Business Integration Adapter (WBIA) Installer. Follow the instructions in the *System Installation Guide for UNIX or for Windows*. When the installation is complete, the following files are installed in the directory on your system where you have installed the product:

- ODA\Siebel\SiebelODA.jar
- ODA\messages\SiebelODAAgent.txt
- ODA\Siebel\start_SiebelODA.bat (Windows only)
- ODA/Siebel/start_SiebelODA.sh (UNIX only)
- bin\CWODAEV.bat (Windows only)
- bin\CWODAEV.sh (UNIX only)

Note: If ICS is your broker, CWODAEV.bat must be modified to reflect the version of ICS. For ICS version 4.2.x, change CWVERSION to 4.2. For ICS version 4.1.1, change CWVERSION to 4.1.

Note: Except as otherwise noted, this document uses backslashes (\) as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes. All WBIA product pathnames are relative to the directory where the product is installed on your system.

Before using SiebelODA

Before you can run SiebelODA, you must copy the required Siebel application's .jar files to the %ProductDir%/connectors/Siebel/dependencies directory. The following files must be copied to this directory:

Siebel 7

SiebelJI_enu.jar
SiebelJI_Common.jar

The default version of Siebel is set to 7.x. Ensure that the REM in the following line is not removed:

```
REM set SIEBELVERSION="6.x"
```

Siebel 6.2.x

SiebelDataBean.jar
SiebelTC_enu.jar
SiebelTcCommon.jar
SiebelTcOM.jar

You must edit the start_SiebelODA.bat file to remove the REM in the line:

```
REM set SIEBELVERSION ="6.X"
```

After installing the SiebelODA, you must do the following to generate business objects:

1. Start the ODA.
2. Start Business Object Designer.
3. Follow a six-step process in Business Object Designer to configure and run the ODA.

The following sections describe these steps in detail.

Launching the SiebelODA

You can launch SiebelODA with the startup script appropriate for your operating system.

UNIX

```
start_SiebelODA.sh
```

Windows

```
start_SiebelODA.bat
```

Note: The Windows Installer provides shortcuts to startup the ODAs it installs. If you have used this Installer to install SiebelODA, you will find a shortcut to start it under the menu Programs > IBM WebSphere Business Integration Adapters > Adapters > Object Discovery Agents.

You configure and run SiebelODA using Business Object Designer. Business Object Wizard, which Business Object Designer starts, locates each ODA by the name specified in the AGENTNAME variable of each script or batch file. The default ODA name for this connector is SeibelODA.

Running SiebelODA on multiple machines

You can run multiple instances of the ODA, either on the local host or a remote host in the network. Each instance has to run on a unique port. You can specify this port number as part of launching the ODA from within Business Object Designer. Figure 2 on page 61 illustrates the window in Business Object Designer from which you select the ODA to run.

Working with error and trace message files

Error and trace message files (the default is `SiebelODAAgent.txt`) are located in the `\ODA\messages`, subdirectory under the product directory. These files use the following naming convention:

AgentNameAgent.txt

If you create multiple instances of the ODA script or batch file and provide a unique name for each represented ODA, you can have a message file for each ODA instance. Alternatively, you can have differently named ODAs use the same message file. There are two ways to specify a valid message file:

- If you change the name of an ODA and do not create a message file for it, you must change the name of the message file in Business Object Designer as part of ODA configuration. Business Object Designer provides a name for the message file but does not actually create the file. If the file displayed as part of ODA configuration does not exist, change the value to point to an existing file.
- You can copy the existing message file for a specific ODA, and modify it as required. Business Object Designer assumes you name each file according to the naming convention. For example, if the `AGENTNAME` variable specifies `SiebelODA1`, the tool assumes that the name of the associated message file is `SiebelODA1Agent.txt`. Therefore, when Business Object Designer provides the file name for verification as part of ODA configuration, the file name is based on the ODA name. Verify that the default message file is named correctly, and correct it as necessary.

Important: Failing to correctly specify the message file's name when you configure the ODA causes it to run without messages. For more information on specifying the message file name, see Table 16 on page 62.

During the configuration process, you specify:

- The name of the file into which SiebelODA writes error and trace information
- The level of tracing, which ranges from 0 to 5.

Table 15 describes these values.

Table 15. Tracing levels

Trace Level	Description
0	Logs all errors
1	Traces all entering and exiting messages for method
2	Traces the ODA's properties and their values
3	Traces the names of all business objects
4	Traces details of all spawned threads
5	• Indicates the ODA initialization values for all of its properties • Traces a detailed status of each thread that SiebelODA spawned • Traces the business object definition dump

For information on where you configure these values, see Table 16 on page 62.

Using SiebelODA in Business Object Designer

This section describes how to use Business Object Designer to generate business object definitions using SiebelODA. For information on launching Business Object Designer, see the *Business Object Development Guide*. Business Object Designer provides a wizard, called Business Object Wizard, that guides you through each of these steps. After you launch an ODA, you must launch Business Object Designer to obtain access to Business Object Wizard (which configures and runs the ODA). There are six steps in Business Object Wizard to generate business object definitions using an ODA.

After starting the ODA, do the following to start the wizard:

1. Open Business Object Designer.
2. From the File menu, select the New Using ODA... submenu.
Business Object Wizard displays the first window in the wizard, named Select Agent. Figure 2 on page 61 illustrates this window.

To select, configure, and run the ODA, follow these steps:

1. "Select the ODA"
2. "Specify configuration properties" on page 61
3. "Selecting the source" on page 63
4. "Confirm selection of objects" on page 66
5. "Generate the business object definition" on page 67 and, optionally, "Provide additional information" on page 68
6. "Save the business object definition" on page 71

Select the ODA

Figure 2 on page 61 illustrates the first dialog box in Business Object Wizard's six-step wizard. From this window, select the ODA to run.

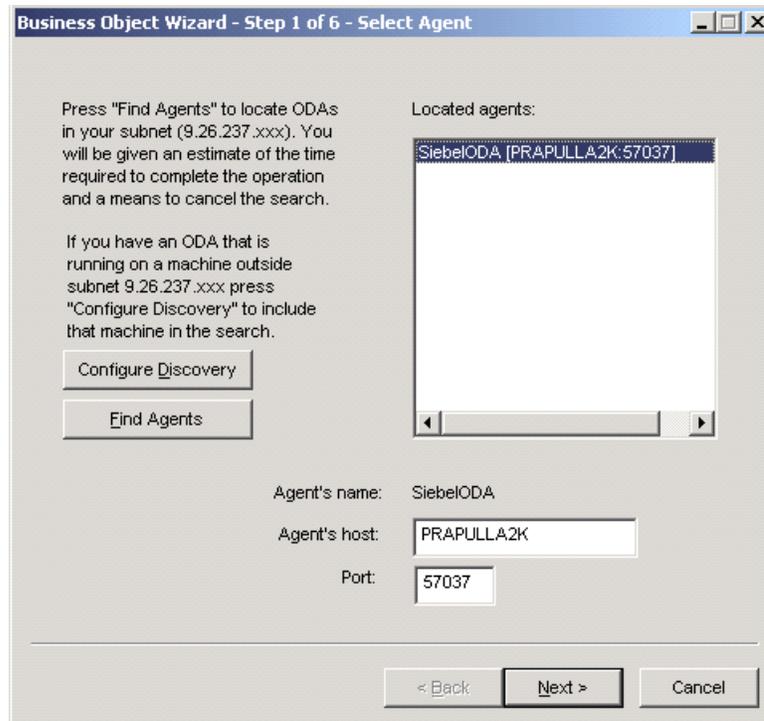


Figure 2. Business Object Wizard, Select ODA screen

To select the ODA:

1. Click the Find Agents button to display all registered or currently running ODAs in the Located agents field. Alternatively, you can find the ODA using its host name and port number.

Note: If Business Object Wizard does not locate your desired ODA, check the setup of the ODA.

2. Select the desired ODA from the displayed list.
Business Object Wizard displays your selection in the Agent's name field.
3. Click Next.

Specify configuration properties

The first time Business Object Wizard communicates with SiebelODA, it prompts you to enter a set of ODA configuration properties as shown in Figure 3 on page 62.

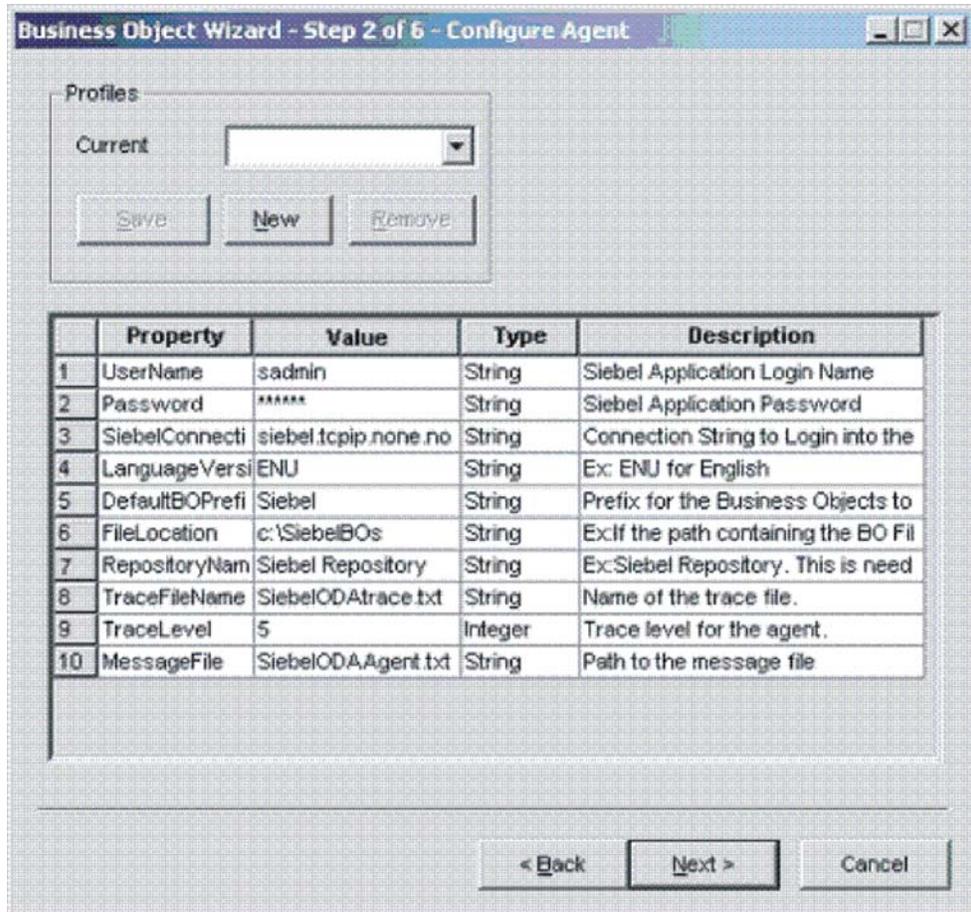


Figure 3. Business Object Wizard, Configure Agent screen

Configure the SiebelODA properties described in Table 16.

Table 16. SiebelODA configuration properties

Row number	Property name	Property type	Description
1	UserName	String	Siebel application login name
2	Password	String	Siebel application password
3	SiebelConnectionString	String	Connect string to log into the Siebel application. Examples: For Siebel 7.0: //machinename/enterprisename/objectmanager/servername For Siebel 7.5: //machinename/enterprisename/objectmanager_ languageCode/servername
4	Language version	String	Language version. For example, use ENU for English.
5	DefaultBOPrefix	String	Prefix that the ODA applies to the name of each business object definition for the Siebel document. If you do not specify a business-object prefix, the ODA does <i>not</i> prepend any string to the name of the business object definition.

Table 16. SiebelODA configuration properties (continued)

Row number	Property name	Property type	Description
6	FileLocation	String	The absolute path containing the files with previous versions of business object definitions. For example, in UNIX, the path is /home/SiebelBos, and in Windows, the path is C:\SiebelBos.
7	RepositoryName	String	The name of the Siebel repository in the Siebel application.
8	TraceFileName	String	Full pathname of the file into which SiebelODA writes trace information. If the file does not exist, SiebelODA creates it in the specified directory. If the file already exists, SiebelODA appends to it. By default, SiebelODA creates a trace file named SiebelODATrace.txt in the ODA\Siebel subdirectory of the product directory. Use this property to specify a different name for the trace file.
9	TraceLevel	Integer	Level of tracing enabled for SiebelODA. Valid values are zero through five (0-5). Property defaults to a value of 5 (full tracing enabled). For more information, see "Working with error and trace message files" on page 59.
10	MessageFile	String	Full pathname of the error and message file. By default, SiebelODA creates a message and error file named SiebelODAAgent.txt. Important: The error and message file <i>must</i> be located in the ODA\messages subdirectory of the product directory. Use this property to verify or specify an existing file.

Important: Correct the name of the message file if the default value displayed in Business Object Designer represents a non-existent file. If the name is not correct when you move forward from this dialog box, Business Object Designer displays an error message in the window from which the ODA was launched. This message does not pop up in Business Object Designer. Failing to specify a valid message file causes the ODA to run without messages.

You can save these properties in a named profile so that you do not need to re-enter them each time you use SiebelODA. For information on specifying an ODA profile, see the *Business Object Development Guide*.

Selecting the source

After you configure all initialization properties for SiebelODA, the Select Source screen appears (see Figure 4 on page 64).

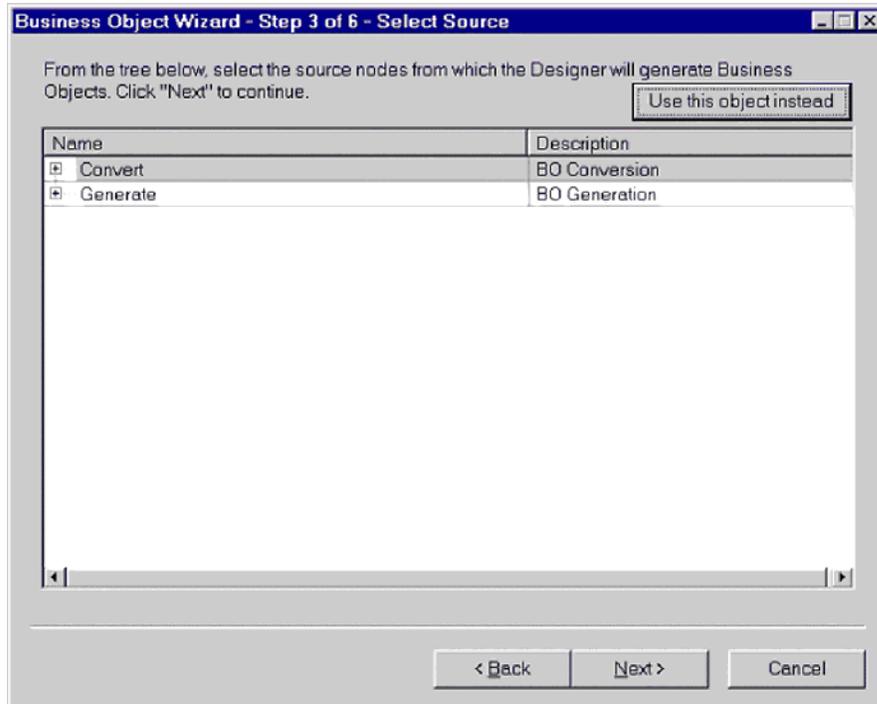


Figure 4. Business Object Wizard, Select Source screen

This screen has two expandable options, Convert and Generate. If you need to convert old business objects into new ones, expand Convert. This displays the repository files that need to be converted (see Figure 5).

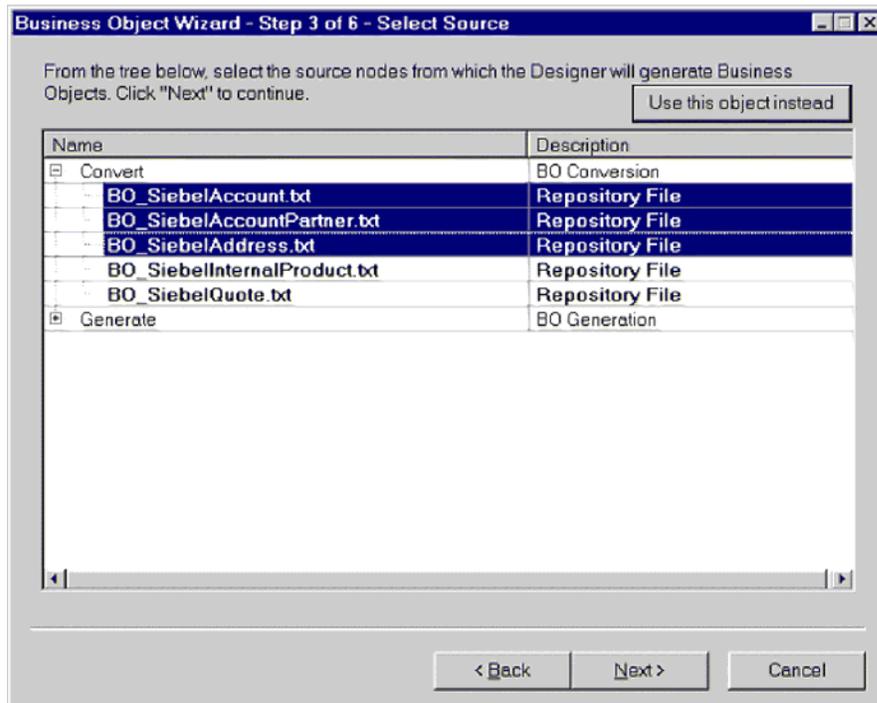


Figure 5. Business Object Wizard, screen displaying business objects to be converted

If you need to generate new business objects, expand Generate. From there, you have three expandable options: Business objects, Integration objects, and Application services. For examples of these options expanded, see Figure 6, Figure 7 on page 66, and Figure 8 on page 66. When you expand a business object, you can select a business component for that object. Similarly, when you expand an integration object, you can select an integration component for that object. When you expand an application service, however, a corresponding integration object is already selected.

Note: If an integration component is listed in both the Application Services and Integraion Object options, that integraion object can be generated only by using Application Services.

Note: When you generate an integration objects, all of the components listed for that object are generated.

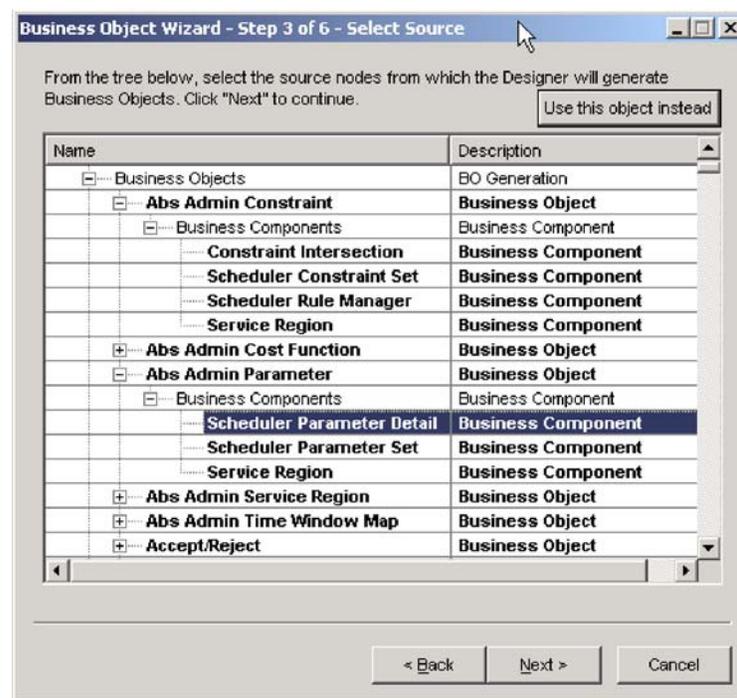


Figure 6. Business Object Wizard, displaying Business Objects expanded

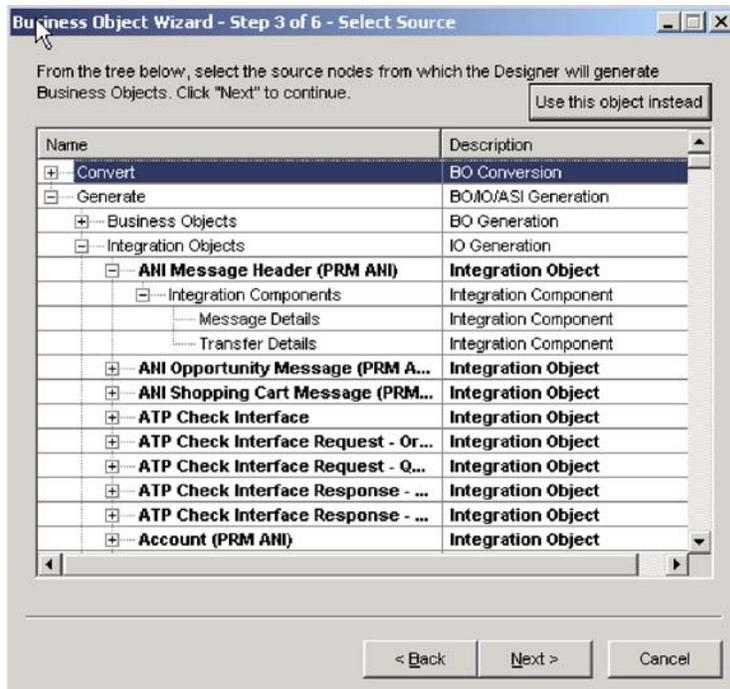


Figure 7. Business Object Wizard, displaying Integration Objects expanded

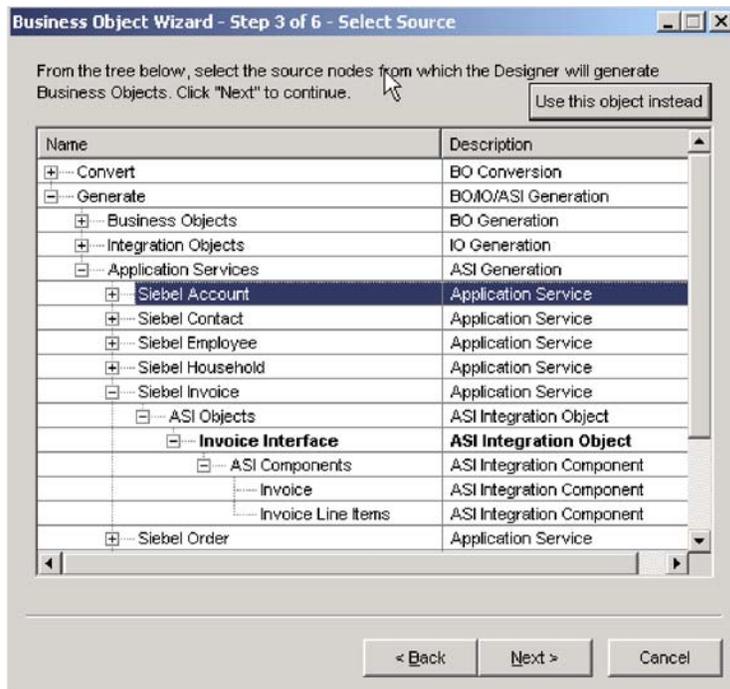


Figure 8. Business Object Wizard, displaying Application Services expanded

Confirm selection of objects

After you identify all the Siebel elements to be associated with the generated business object definitions, Business Object Designer displays the dialog box with

only the selected objects and components. Figure 9 illustrates this dialog box.

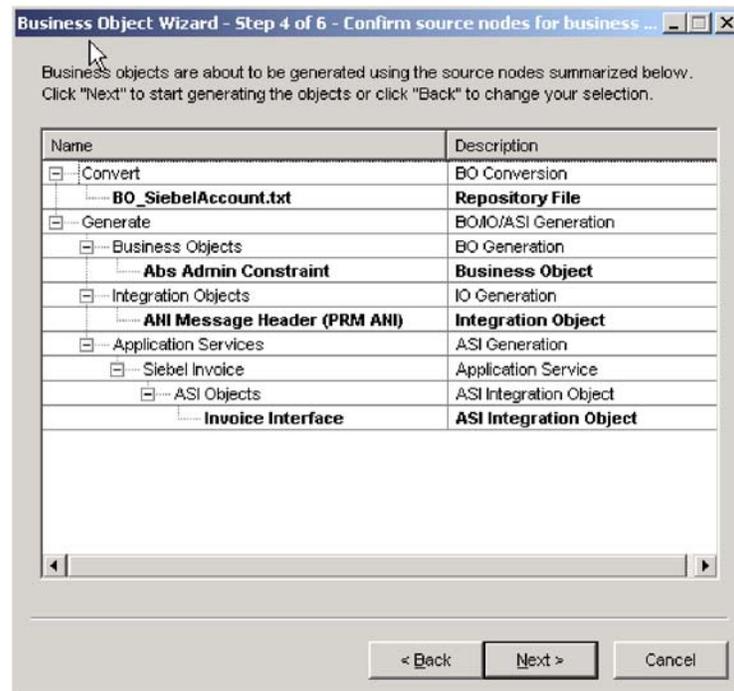


Figure 9. Business Object Wizard, confirming selection of objects and components

This window provides the following options:

- To confirm the selection, click Next.
- If the selection is not correct, click Back to return to the previous window and make the necessary changes. When the selection is correct, click Next.

Generate the business object definition

After you confirm the Siebel elements, the next dialog box informs you that Business Object Designer is generating the business object definition. If a large number of Component Interfaces has been selected, this generation step can take time.

Figure 10 on page 68 illustrates this dialog box.

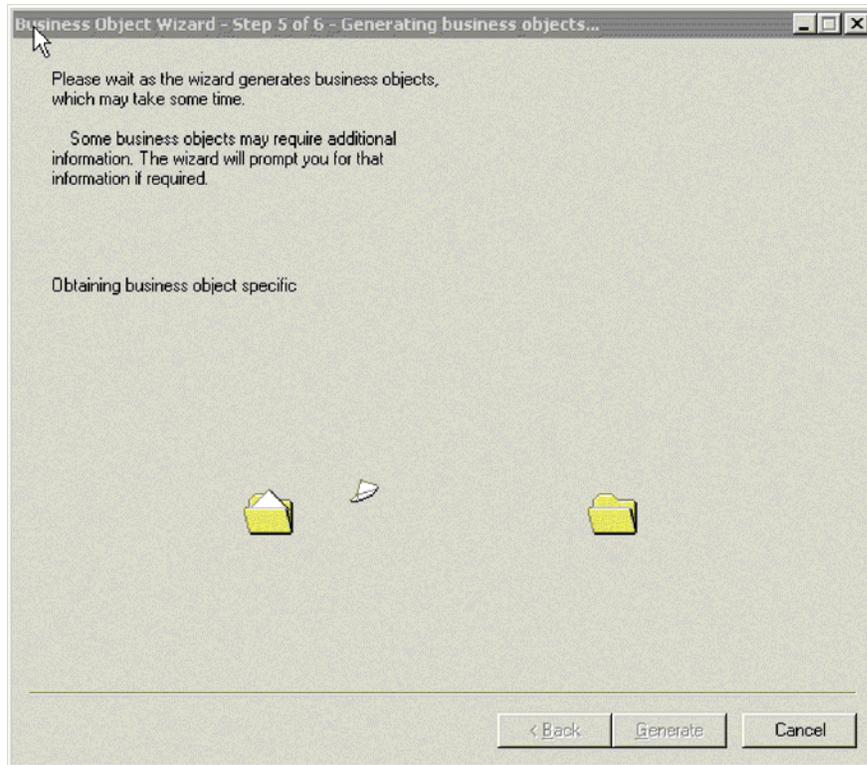


Figure 10. Generating the business object definitions

Provide additional information

Because SiebelODA needs additional information about the verbs, Business Object Designer displays the BO Properties window for each of the generation types you chose (business objects, integration objects, and application services), which prompts you for the information. Figure 11 on page 69 illustrates these screens.

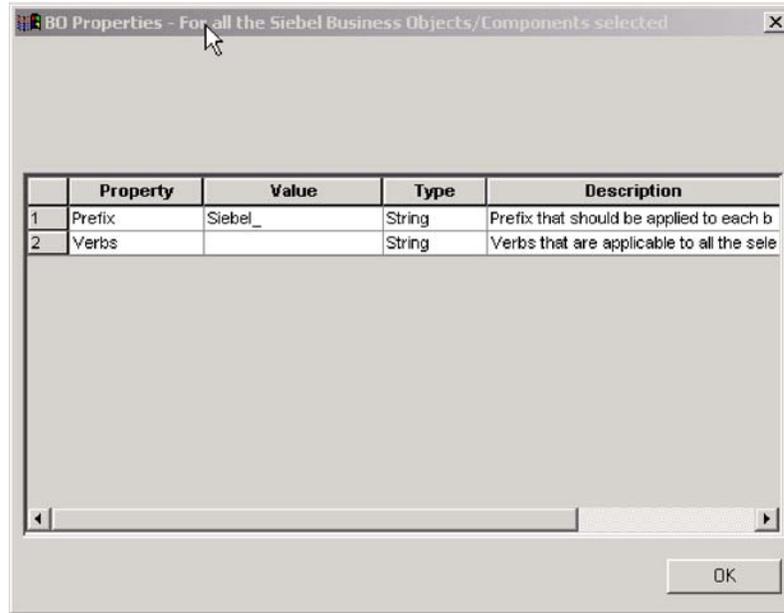


Figure 11. Providing additional information for business object

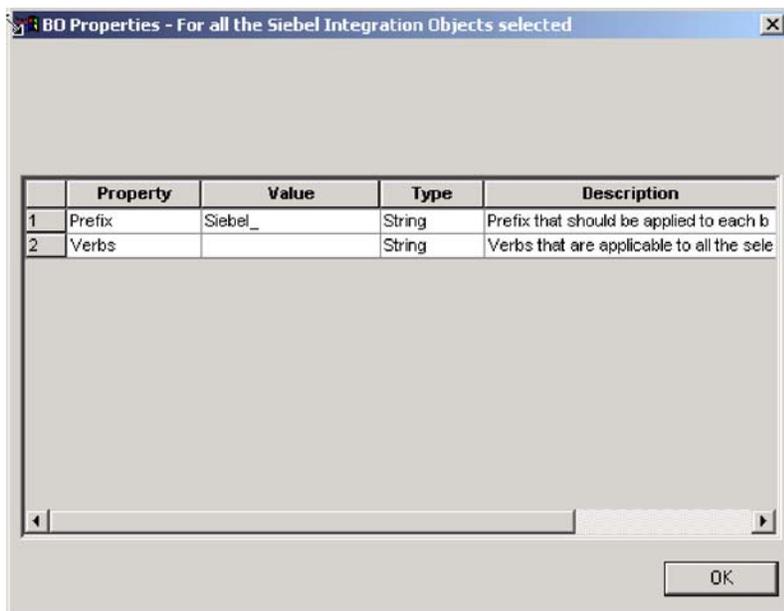


Figure 12. Providing additional information for integration object

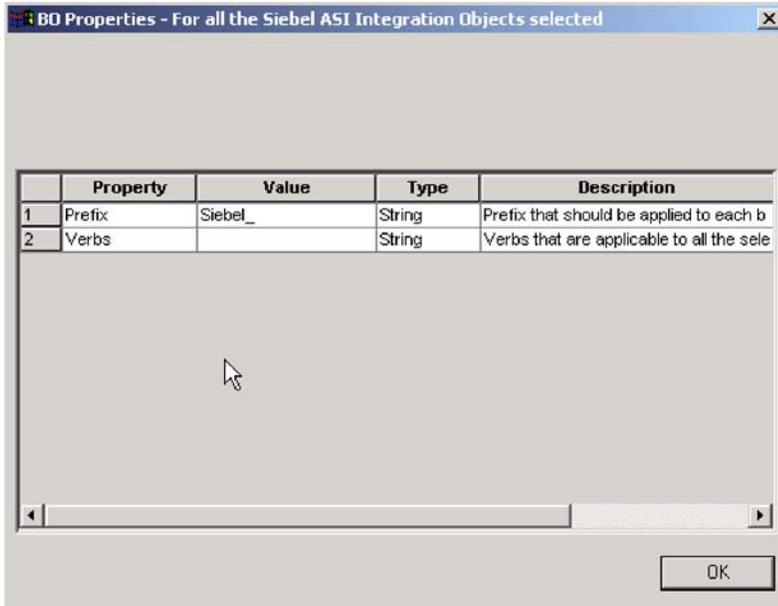


Figure 13. Providing additional information for application service

In the BO Properties window, enter or change the verb information. Click in the *Value* field and select one or more verbs from the pop-up menu. These are the verbs supported by the business object.

Note: If a field in the BO Properties dialog box has multiple values, the field appears to be empty when the dialog box first displays. Click in the field to display a drop-down list of its values.

If your Siebel document has a schema document that contains an `anyAttribute` element, SiebelODA displays an additional BO Properties window, as shown in Figure 14.

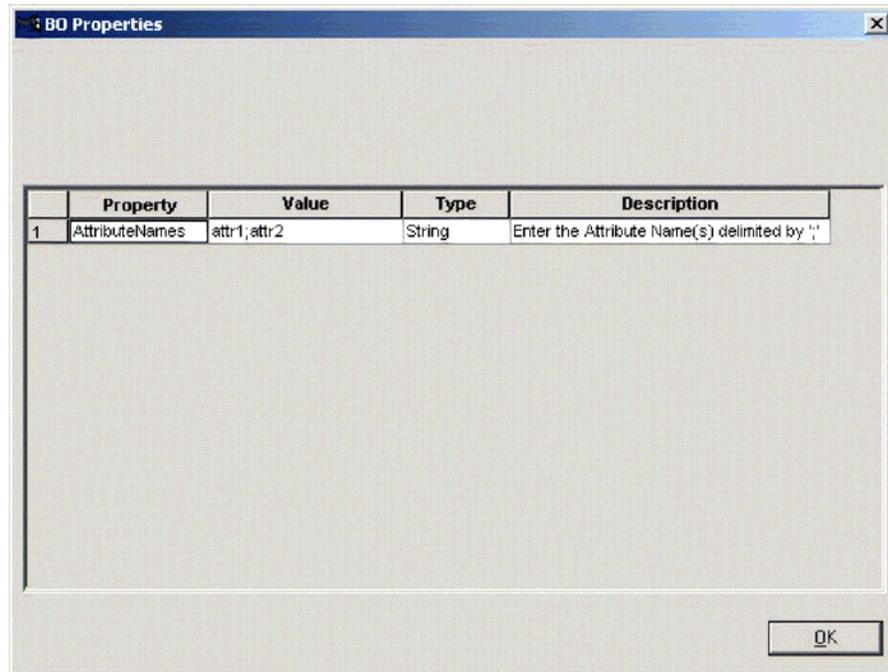


Figure 14. Providing additional information - attribute names

In this BO Properties window, enter the names of the business object attributes you want SiebelODA to create. Separate each attribute with a semicolon (;).

Save the business object definition

After you provide all required information in the BO Properties dialog box and click OK, Business Object Designer displays the final dialog box in the wizard. In this dialog box, you can take any of the following actions:

- Save the business object definition to the server (if InterChange Server is the integration broker).
- Save the business object definition to a file (for any integration broker).
- Open the business object definition for editing in Business Object Designer.

For more information, and to make further modifications, see the *Business Object Development Guide*.

Figure 15 on page 72 illustrates this dialog box.

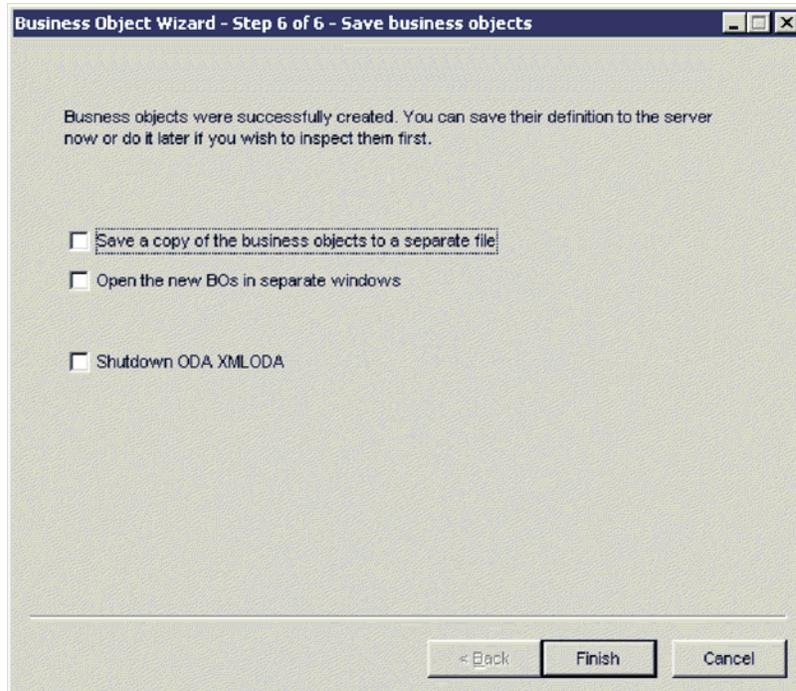


Figure 15. Saving the business object definition

Contents of the generated definition

The business object definition that SiebelODA generates contains:

- An attribute for each column in the specified database tables and views
- The verbs specified in the BO Properties window
- Application-specific information:
 - At the business-object level
 - For each attribute
 - For each verb

When generating business objects by getting the information from the Siebel application server, the application-specific information generated is for simple attributes only. The exception to this rule is if the container attribute is a multi-value link. In all other cases, the user must enter the application-specific information as described in Chapter 6, “Using business services,” on page 83.

This section describes:

- “Business-object-level properties” on page 72
- “Attribute properties” on page 73
- “Verbs” on page 74

Business-object-level properties

SiebelODA generates the following information at the business-object level:

- Name of the business object
- Version—defaults to 1.0.0
- Application-specific information

Application-specific information at the business-object level contains the name of the corresponding Siebel business object or business component.

Attribute properties

This section describes the properties that SiebelODA generates for each attribute.

Important: Any user edits described in the following sections refer to business object generation only, not to business object conversion.

Name property

SiebelODA obtains the value of the attribute's name from the corresponding attribute in the Siebel business component.

Data type property

When setting the type of an attribute, SiebelODA converts the data type of the attribute in the Siebel business component and converts it to the corresponding data type, as shown in Table 17. This is only in the case of business object generation, since business object conversion is for existing business objects.

Table 17. Correspondence of data types

Application	WebSphere business integration system	Length
DTYPE_BOOL	BOOLEAN	
DTYPE_ID, DTYPE_PHONE	STRING	Length of corresponding attribute in the Siebel application server
DTYPE_TEXT DTYPE_NOTE		
DTYPE_DATE DTYPE_TIME DTYPE_DATETIME DTYPE_UTCDATETIME	DATE	
DTYPE_INTEGER DTYPE_NUMBER	INTEGER	
DTYPE_CURRENCY	DOUBLE	

Note: If an attribute's data type is not one of those shown in Table 17, SiebelODA skips the column and displays a message stating that the column cannot be processed.

Cardinality property

SiebelODA sets the cardinality of all simple attributes to 1 and the container attributes to n. The user should change the cardinality of the container attributes wherever it is needed. For example, if the container attribute turns out to be a PickList, the user needs to set the cardinality to 1.

MaxLength property

SiebelODA obtains the length of the attribute from the Siebel application server.

IsKey property

If the column is a primary key in the table or view, SiebelODA marks it as a key attribute. In the case of business object generation, the Id attribute is the only one marked as key by default.

IsRequired property

If a field is designated not null in the table or view, SiebelODA marks it as a required attribute. However, SiebelODA does not mark the key field as required because the Siebel application generates its own Id values while creating a record.

AppSpecificInfo Property

The user should edit this property if container attributes have not been generated and ensure the correctness if container attributes have been generated.

PollQuantity

Number of rows in the database table that the connector retrieves per polling interval. Allowable values are 1 to 500.

The default is 1.

Verbs

SiebelODA generates the verbs specified in the BO Properties window. It creates an AppSpecificInfo property for each verb but does not populate it.

Adding information to the business object definition

Since Siebel business objects and business components may not have all the information that a business objects requires, it may be necessary to add information to the business object definition that SiebelODA creates, especially when generating new business objects.

To examine the business object definition or reload a revised definition into the repository, use Business Object Designer. Alternatively, if ICS is the integration broker, you can use the `repos_copy` command to load the definition into the repository; if WebSphere MQ Integrator Broker is the integration broker, you can use a system command to copy the file into the repository directory.

Note: Because the calculated fields in Siebel Application do not correspond to a column, they are not being generated by the ODA. These fields can be manually added to the Business Object Definition.

Chapter 5. Using the Siebel connectivity DLL

The Siebel Enterprise Application Integration (EAI) framework provides transports that allow Siebel applications to exchange data with external applications. To do this, the Siebel EAI Dynamic Link Library (DLL) transport invokes the WebSphere Integration Adapter (WIA) Siebel connectivity DLL which invokes the access framework interface provided by the ICS Integration Broker. The WebSphere Integration Adapter Suite consists of adapters for various applications, hence, ICS and the appropriate WBI adapters can process the Siebel generated request sent by the DLL. The DLL is unidirectional; requests sent from Siebel to the broker (ICS) are processed, and not vice versa.

The following topics are covered in this chapter:

- “Connector architecture and processes”
- “Configuring the Siebel connectivity DLL” on page 77
- “Communicating with the Siebel connectivity DLL” on page 79
- “Generating WBIA business objects” on page 79
- “Configuring the IBM generic business services” on page 80

Connector architecture and processes

In the context of the Siebel EAI, data is transported in the form of an integration message. In this context, a message typically consists of header data that identifies the message type and structure, and a body that contains one or more instances of data, such as orders, accounts, or employee records.

The following diagram illustrates the Siebel connector architecture and processes.

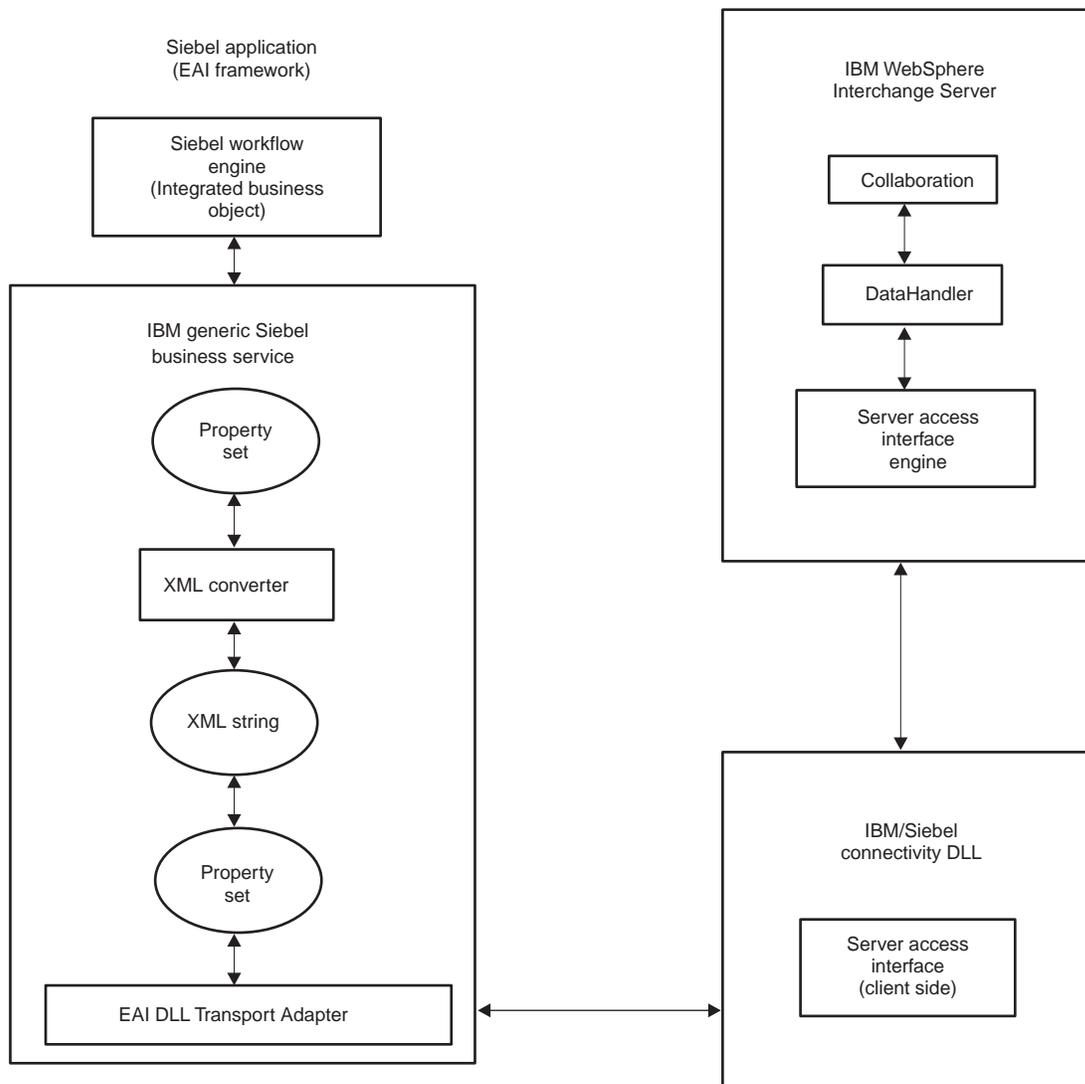


Figure 16. Siebel connector and architecture process

The IBM generic Siebel business service is provided as part of the Siebel connectivity DLL solution. A business service is an object that encapsulates and simplifies the use of some set of functionality, such as the task of moving data and converting data formats. Business components and business objects are typically tied to specific data and tables in the Siebel data model. Alternately, business services are not tied to specific objects; they operate or act upon objects to achieve a particular goal.

IBM generic business service can be invoked by a workflow or other business services. The generic business service receives requests in the form of a property set. The XML converter, converts data from property sets to XML string. During conversion, the string is stripped of the tags that the WBI DataHandler does not support. This stripped down XML string is sent as a property set to the EAI DLL transport to invoke the send or receive method. The EAI DLL transport adapter sends and receives integration messages to and from the IBM Siebel connectivity DLL.

The EAI DLL transport adapter:

- Allows Siebel applications to exchange data with external applications using standard technologies for synchronous communication protocols
- Supports only outbound messages
- Invokes the IBM Siebel connectivity DLL
- Is synchronous and waits for a response back after messages are sent before continuing

To execute the collaboration, the IBM Siebel connectivity DLL makes server access interface calls. The collaboration represents the business process that can represent multiple applications. The XML request from Siebel is then converted to WBI business objects using the XML DataHandler.

The interchange server invokes the XML DataHandler through the server access interfaces. In this instance, the DataHandler converts the XML input from Siebel into a generic WBI business object. The resultant message is passed back by the interchange server to the Siebel connectivity DLL. The connectivity DLL passes the message to the IBM Siebel generic business service. In the generic business service, the Siebel-specific tags stripped earlier are added back to the message and are converted back to a Siebel property set using the XML converter service. The message is set to the generic Siebel business service's output property set. At this point, the Siebel workflow will pickup the message.

Configuring the Siebel connectivity DLL

The WBI Siebel connectivity DLL is installed as part of the Siebel suite of products. When you are running the WBI CD installation program, choose Siebel for your installation. The installation puts the Siebel connectivity DLL in the following folders, depending on which version of Siebel you are using:

Siebel 7.0.x

ProductDir\connectors\Siebel\SiebelEAIAdapter\Siebel7.0.x

Siebel 7.5.x

ProductDir\connectors\Siebel\SiebelEAIAdapter\Siebel7.5.x

The other components are installed under the following folder:

ProductDir\connectors\Siebel\SiebelEAIAdapter

This section describes how to configure and set up the WBI Siebel connectivity DLL.

Editing the IBMCrossWorlds.cfg file

To add the [collabdetails] section (the Collaboration Name, Generic BO Name, PortName etc.), you must edit IBMCrossWorlds.cfg file. You can find a copy of the file in the %*ProductDir*%\connectors\Siebel\SiebelEAIAdapter\Samples directory. The file should contain the items listed in Table 18.

Table 18. IBMCrossWorlds.cfg file contents

Item	Description
SERVERNAME	Name of your interchange server.
USERNAME	Siebel application user name.
PASSWORD	Siebel application password.

Table 18. *IBMCrossWorlds.cfg* file contents (continued)

Item	Description
[CollabDetails]	<p>Details for each collaboration that is being used. For example, an account object would include:</p> <pre>[CollabDetails] IntObjName=Account BOName=NewSiebel_Account CollabName=SiebelConnectivityObject PortName=From [EndCollabDetails]</pre>

Copying the files

Copy the following files from the install folder to the %Siebel%/Client/bin directory:

- SiebelConnectivity.DLL
- IBMCrossWorlds.cfg
- SiebelConnectivityMessages.txt

Generating persistent .ior files

When the InterChange Server is started, it generates an *.ior* file, the persistent object reference to the interchange server access engine object. The name of the file is *[Server Name] InterChangeServer.ior* and can be found in %WBIA%. Copy this file to the %Siebel%/Client/bin directory. Each time the ICS starts, it generates a new *.ior* file. To keep the contents of this *.ior* file the same for every boot of interchange server, a persistent *.ior* file is generated. To have the server generate a persistent *.ior* file, edit the *InterchangeSystem.cfg* file, and add a subsection for CORBA, if one does not already exist.

The CORBA subsection contains the port number in the format: *OAport=portnumber*. For example, if the port number is 15000, add the following lines to the *InterchangeSystem.cfg* file:

```
[CORBA]USERNAME=Siebel Application user name
OAport=15000
```

If the access client is in the DMZ and InterChange Server is on a different subnet, make sure that the port number for the *OAport* is open.

Note: For more information on the system guide, copy the *[Server Name] InterChangeServer.ior* file to the %Siebel%/Client/bin directory.

Generating and copying the WBIA business object

Generate a WBIA business object definition for every integration object using XMLborgen or XMLODA. The object generated will be used by the workflow process. The business objects generated are automatically loaded into the interchange server repository.

Configuring the XML DataHandler

The XML DataHandler is a data-conversion module whose primary role is to convert business objects to and from XML documents. An XML document is serialized data with the text/xml MIME type. This XML document is called from within the server access interface. To configure the XML DataHandler, the

configuration information must be provided in the XML child meta-object. For the XML DataHandler, WebSphere business integration system delivers the default meta-object `MO_DataHandler_DefaultXMLConfig`. Table 19 shows attributes contained in the meta-objects that must be configured.

Table 19. Attributes contained in the meta-object

Attribute	Description
BOPrefix	The WBIA Business Object Name with the appropriate prefix is given in the configuration file. Hence, this attribute should be left blank.
ClassName	The ClassName attribute, by default, will point to XML DataHandler class that is invoked by the server access interface. The default value is <code>com.crossworlds.DataHandlers.text.xml</code> .

Defining the collaboration template

Define a collaboration template and create a collaboration object for the business process. For more details on defining collaboration templates and creating collaboration objects, refer to the “Collaboration development Guide”.

Importing the business service

To import the business service:

1. Start the Siebel client.
2. Select View—>Site Map—>Business Service Administration—>Business Service Scripts.
3. Import the business service from the file, `IBMSiebelConnectivityService.xml`, found under the following folder:
`%ProductDir%connectors\Siebel\SiebelEAIAdapter\dependencies.`

Configuring the Siebel workflow

Follow the workflow process set up described in the Siebel documentation and add the business service trigger. This trigger will call the `IBMsiebelConnectivityService` into action. The method name needs to be set by the workflow according to the process that needs to be performed.

Communicating with the Siebel connectivity DLL

As the Siebel Connectivity DLL is a server access client, it uses CORBA to communicate with ICS. To ensure that the access client can locate the interchange server instance, access to an interoperable object reference file for that interchange server instance needs to be provided. For access details, refer to “User setup” on page 9.

Generating WBIA business objects

Siebel integration objects represent integration metadata for the Siebel business object. XML is a common structure that the Siebel EAI infrastructure understands. Because these integration objects adhere to a set of structural conventions, they can be traversed and transformed programmatically, using Siebel eScript objects, methods, and functions or transformed declaratively using Siebel Data Mapper. For more information, see the *Siebel eBusiness Application Integration, volume IV*.

Siebel tools provide the ability to generate the schema or DTD for the integration objects. The DTDs generated become the input for the WBI utility, XMLborgen or XMLODA. The WebSphere business integration system business object is generated from the XMLborgen and XMLODA DTD. While executing the XMLborgen, give the top-level object name as the one which has the string, *ListOf*, prepended to its name.

As an example, ListOfChargesData in the DTD, can be found in the file, ListofChargerData.dtd, under the following directory:
`%ProductDir%connectors/Siebel/Siebel/SiebelEAIAdapter/Samples.`

Note: For more information on generating schemas or DTDs, see *Siebel Integration Platform Technologies*.

Business objects generated by XMLborgen, can be found in the `%ProductDir%connectors/Siebel/SiebelEAIAdapter/Samples` directory.

Note: Remember to generate the DTD without the Siebel message envelope. However, for this, you need to select the *No Envelope* option. While passing in the property set, which would have the actual data, you need to send it with the Siebel message tags, `<SiebelMessage>` and `</SiebelMessage>`, and the `<?Siebel-Property-Set...>` tag.

Configuring the IBM generic business services

IBM provides the generic Siebel business service, IBMSiebelConnectivityService as part of the WBI Siebel EAI Adapter. This business service needs to be part of the Siebel workflow process. The Siebel workflow passes the Siebel property set containing the integration object instance (data) to the IBMSiebelConnectivityService. This property set should contain the `<SiebelMessage>` tag. The Siebel integration object name is obtained from this tag. The IBMSiebelConnectivityService, in turn, will use an instance of the Siebel business service, XML converter, to convert the property sets to XML. After the integration object name is obtained from the converted XML string, it is stripped of the two tags, `<SiebelMessage>` and `</SiebelMessage>`, that are not supported by the WebSphere business integration system Data Handler.

Siebel connectivity DLL supports one method for each of the following verbs:

- Create
- Retrieve
- Update
- Delete

The IBMSiebelConnectivityService takes a parameter called `MethodName` in the `Service_PreInvokeMethod`. This methodname maps to a method in the Siebel connectivity DLL. Table 20 shows the mapping between the `MethodName` parameter and the verb supported by the DLL.

Table 20. Mapping supported by the Siebel connectivity DLL

Method name	Verb
Insert	Create
Update	Update
Retrieve	Retrieve

Table 20. Mapping supported by the Siebel connectivity DLL (continued)

Method name	Verb
Delete	Delete

Once the value of the `MethodName` is verified by the generic business service, the Siebel request is processed as follows:

- The edited XML string is set as part of another property set.
- The EAI DLL transport business service is invoked and the following parameters are set:
 - `MethodName` is set to the value of `ExternalFunction` property
 - `SiebelConnectivity` DLL's name is set to the `DLLName` property of the same service
- The `SendReceive` method of the EAI DLL transport service is then called and the property set is passed as input. The input property set is passed on to the Siebel connectivity DLL in the form of a XML string. The Siebel connectivity DLL will then perform the needed operation through the server access interface calls. It will then return the result in the form of XML string back to the generic Siebel business service.

In the generic Siebel business service, the Siebel-specific tags are added back to the resultant XML string that comes back from the connectivity DLL. The edited XML string is then converted back to a property set using the instance of XML converter service again. The result is stored as part of the output property set parameter of the `IBMSiebelConnectivityService`. The Siebel workflow would obtain the same from the `IBMSiebelConnectivityService` and perform the required operation.

Chapter 6. Using business services

This chapter describes how to use Siebel business services and covers the following topics:

- “Understanding business services”
- “Verb processing with business services” on page 85
- “Events detection with business services” on page 86

Understanding business services

This section explains what a business service is and describes how to create business objects that support business services. The following topics are covered:

“Description of business services”

“Creating business objects that support business services”

Description of business services

A business service is an object that encapsulates and simplifies the use of some sets of functionality, such as moving and converting data formats between the Siebel application and external applications. Business components and business objects are objects that are typically tied to specific data and tables in the Siebel data model. Business services, on the other hand, are not tied to specific objects, but rather operate on objects to achieve a particular goal.

The adapter supports generic business services and ten application-specific interfaces. The business services take the integration object instances (or property sets) as the input. The integration object instance is created by building the definition for the integration object, then building the property set based on the values of the integration object definition. Once the integration object instance is created, it can be sent to the business services. Existing integration object definitions can be used, as can existing business services, such as EAI Siebel Adapter.

Note: If an object contains the application-specific information BSN, then it will be treated as a business service.

Creating business objects that support business services

The adapter constructs the property set for the incoming business object, which is the representation of the integration object. The following example describes how to construct the property set.

Example:

```
//instantiate a new property set for type Siebel Message  
  
//set the type as Siebel MMessage and properties such as IntObject Format,  
MessageId, IntObjectName  
  
//instantiate a new property set for type ListOf<Parent Component>
```

```
//set the type as ListOf<Parent componet name>
//instantiate a new property set for type <Parent Component>
//set type as <parent object type>
//set different properties
//do the same for child components
```

You can create business objects that support business services using one of the following methods:

- Create integration objects that relate to the Siebel Application Specific Interface
- Create separate integration objects

The following examples show how to create each type of integration object.

Siebel Integration object

```
Account (PRM ANI)          (Integration object)
+Account (Integration component)
+Business address (Integration component)
```

Integration object

```
Siebel <IntObjectName> (ParentIntegrationComponent)
BSN=<name>;IO<Name>;IC=<Name>
Attribute1 FN=<fieldname>
Attribute2 FN=<fieldname>
Attribute3 FN=<fieldname>
+ChildIntegrationComponent
IO=<Name>;IC=<Name>
childAttribute1 FN=<fieldname>
childAttribute2 FN=<fieldname>
Object level ASI:
for Parent Integration Component
BSN=<name>;IO=<Name>;IC=<Name>
for Child Integration Component:
IO=<Name>;IC=<Name>
```

The following tables describe the Business object level application text and the Simple attribute level application text used when creating integration objects.

Table 21. Business object level application text

Parameter	Description
I0=	The name of the Siebel integration object corresponding to this business object.
IC=	The name of the Siebel integration component corresponding to this business object.
BSN=	The name of the business service used by this business object. When using application specific information, such as Siebel Account or Siebel Contact, the specific business service must be present. When using other integration objects, the EAI (put explanation of acronym here) must be present.
SiebASI=	When a business object represents the ASI integration object, it contains SiebASI=true

Table 22. Simple attribute level application text

Parameter	Description
FN=	The field name of the field in the Siebel integration component corresponding to this attribute

Verb processing with business services

The following verbs are supported by business services.

Note: The returned code for all the verbs in Table 23 is VALCHANGE.

Table 23. Verbs supported by business services

Verb	Description
Delete	Parent object keys are used to delete the Siebel object. The adapter verifies that all primary keys are present.
Insert	The complete incoming business object is used for the Insert verb.
InsertOrUpdate (Upsert)	If an object with the same keys as the input object exists, merge the specified input object with the existing object. Otherwise, create a new object in Siebel based on the input object. The adapter verifies the existence of all the primary keys before processing the object.
QueryByExample (or Query in the case of EAI Siebel Adapter)	Queries for objects based on the example object provided. This operation can be treated as a Retrieve by content operations.
QueryById	If the object with the keys exists, it is queried or retrieved. These operations can be treated as Retrieve operations.
Update	If an object with the same keys as the input object exists, merge the specified input object with the existing object. Otherwise, error out.

Table 23. Verbs supported by business services (continued)

Verb	Description
Synchronize	If n object with the same keys as the input object exists, make it look like the input object. Otherwise, create a new object in Siebel based on the input object.

The following example describes a process flow for using any of the verbs in Table 23 on page 85.

Example process flow:

```
//Get the business service name,  
  
//construct the property set based on the input,  
  
//invoke the verb on the specified business service passing in the input  
property set, then  
  
//construct the business object from the output property set.
```

Events detection with business services

The scripts for triggers remain the same when using business services, except that the name of the business object and the verb change. The triggers should be written on the business object on which the integration object is based. The trigger should populate the new verb and the corresponding business object to the integration object when creating an event.

Because the adapter overrides the default `getB0()` method, the verb `RetrieveByContent` must be set before calling the `doVerbFor` method. In this scenario, if the business object is an integration object, the verb `QueryByExample` will be set, whereas if the business object is an application-specific interface, the verb `Query` will be set. The corresponding method for `QueryByExample` (which is equivalent to `RetrieveByContent` in the generic business service, EAI Siebel Adapter), is `Query`.

Chapter 7. Troubleshooting

This chapter describes error messages that you may encounter when running the connector and possible fixes for those errors.

MAX_LOG_FILE_SIZE error on UNIX

When running the connector on UNIX, you may receive the following error message:

Using default value UNLIMITED for configuration parameter MAX_LOG_FILE_SIZE in subsystem LOGGING.

This error message may result from the following conditions:

- The OS agent is not running.
- Incorrect information in the InterchangeSystem.cfg file.
- Incompatible connector and InterChange Server versions.

Handling errors

The Siebel connectivity DLL component of the WBI Siebel EAI adapter returns an error in the following cases:

- Inability to connect to the interchange server
- Inability to open log or trace files for writing
- Errors occurring when creating the WebSphere business integration system business object from input XML messages. This could be the result of incorrect data formatting in the incoming XML data. In the case of a data error in a request, the workflow does not stop processing the batch request. To find the failed requests, the workflow log needs to be looked at to find the failed requests. After fixing the errors, the request must be sent again. This scenario can also occur if the Interchange Server goes down while processing the request. The user will have to look at the workflow log to check and see if a series of requests have failed. Then, the user will have to verify that the Interchange Server is up, and the failed requests will need to be resent.
- Errors in executing the collaboration can be caused due to a wrong verb setting, attribute values not having the correct formatting, or if the external application or Interchange Server goes down. This could also happen if the collaboration does not exist or is not up. The user will have to ensure that the external application is up and the Interchange Server is up prior to analyzing the cause of the error. The user will have to look at the DLL log and trace files to find the cause of the error, fix the error and send the request again.
- Unable to open log or trace files for writing.
- There are error scenarios in which the Siebel Connectivity DLL has no control. For example, consider the scenario in which the WebSphere business integration system business object has been built and sent to the collaboration for processing, the results have been returned to the DLL and the Siebel application triggering the workflow is down. In addition, the request has already been sent out and processed by the external application. In this case, the user will have to check for the requests processed prior to the application shut down. Also, the

user needs to verify that the request processing has succeeded in the external application. If anything in the batch needs to be sent again, the same needs to be handled on an individual request basis.

For all the above cases excluding the last one, the connectivity DLL returns an error message in an XML message. The error message is enclosed in the tags <Error> and </Error>. The generic business adds the Siebel specific tags to the error message, then, converts it to a property set by calling the XML converter service. All error messages from the generic business service needs to be handled by the workflow from which it was invoked.

The Siebel connectivity DLL logs the trace and error messages to the SiebelConnectivityTrace.txt file created in the Siebel\Client\bin folder.

Decreasing the size of the Siebel log file

Seibel JAVABean allows you to change the logging timeout value.

To reset the logging timeout value in Siebel JAVABean:

1. Select Site Map > Server Admin > Components (Sales Object Manager).
2. In the lower applet, go to Component Parameter and enter a timeout value.

Note: The Request Timeout current value is set to 600. This means that the connector will die after ten minutes. Based on Siebel, you can change this value to be as large as you want.

Appendix A. Standard configuration properties for connectors

This appendix describes the standard configuration properties for WebSphere Business Integration adapter connectors. The information covers connectors running on the following brokers:

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator Broker (WMQI)
- WebSphere Application Server (WAS)

Not every connector makes use of all these standard properties. When you select a 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 either added or deleted in the 2.3 release of the adapters.

New properties

- RFH2Message Domain
- ListenerConcurrency
- RestartCount
- WsifSynchronousRequestTimeout

Deleted properties

None

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 the WebSphere MQ Integrator Broker, 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 24 on page 91 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.

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 24. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdminInQueue	Valid JMS queue name	CONNECTORNAME /ADMININQUEUE		Delivery Transport is JMS
AdminOutQueue	Valid JMS queue name	CONNECTORNAME/ADMINOUTQUEUE		Delivery Transport is JMS
AgentConnections	1-4	1	Component restart	ICS: Delivery Transport is MQ or IDL
AgentTraceLevel	0-5	0	Dynamic	
ApplicationName	application name	The value that is specified for the connector application name	Component restart	Value required
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	No value	Component restart	
ContainerManagedEvents	No value or JMS	JMS		Guaranteed event delivery
ControllerStoreAndForwardMode	true or false	True	Dynamic	ICS only
ControllerTraceLevel	0-5	0	Dynamic	ICS only
DeliveryQueue		CONNECTORNAME/DELIVERYQUEUE	Component restart	JMS transport only
DeliveryTransport	MQ, IDL, or JMS	JMS	Component restart	For WAS or WMQI: JMS only
DuplicateEventElimination	True/False	False	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		CONNECTORNAME/FAULTQUEUE	Component restart	
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory or CxCommon.Messaging.jms.SonicMQFactory or any Java class name	CxCommon.Messaging.jms.IBMMQSeriesFactory	Server restart	JMS transport only
jms.MessageBrokerName	If FactoryClassName is IBM, use crossworlds.queue.manager. If FactoryClassName is Sonic, use localhost:2506.	crossworlds.queue.manager	Server restart	JMS transport only
jms.NumConcurrentRequests	Positive integer	10	Component restart	JMS transport only
jms.Password	Any valid password		Server restart	JMS transport only

Table 24. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
jms.UserName	Any valid name		Server restart	JMS transport only
JvmMaxHeapSize	Heap size in megabytes	128m	Component restart	ICS only
JvmMaxNativeStackSize	Size of stack in kilobytes	128k	Component restart	ICS only
JvmMinHeapSize	Heap size in megabytes	1m	Component restart	ICS only
ListenerConcurrency	1- 100	1	Component restart	ICS only: 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	ICS only
MaxEventCapacity	1-2147483647	2147483647	Dynamic	ICS: Repository Directory must be <REMOTE>
MessageFileName	<i>path/filename</i>	<i>Connectorname.txt</i> or <i>InterchangeSystem.txt</i>	Component restart	
MonitorQueue	Any valid queue name	<i>CONNECTORNAME/MONITORQUEUE</i>	Component restart	JMS transport only: DuplicateEvent Elimination must be True
OADAutoRestartAgent	True or False	False	Dynamic	ICS only: Repository Directory must be <REMOTE>
OADMaxNumRetry	<i>A positive number</i>	1000	Dynamic	ICS only: Repository Directory must be <REMOTE>
OADRetryTimeInterval	<i>A positive number in minutes</i>	10	Dynamic	ICS only: Repository Directory must be <REMOTE>
PollEndTime	HH:MM	HH:MM	Component restart	
PollFrequency	<i>a positive integer in milliseconds</i> no (to disable polling) key (to poll only when the letter p is entered in the connector's Command Prompt window)	10000	Dynamic	

Table 24. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
PollQuantity	1-500	1	Component restart	JMS transport only: DuplicateEvent Elimination must be True
PollStartTime	HH:MM(HH is 0-23, MM is 0-59)	HH:MM	Component restart	
RepositoryDirectory	Location of meta-data repository		Component restart	For ICS: set to <REMOTE> For WMQI and WAS: set to <local directory>
RequestQueue	Valid JMS queue name	CONNECTORNAME/REQUESTQUEUE	Component restart	
ResponseQueue	Valid JMS queue name	CONNECTORNAME/RESPONSEQUEUE	Component restart	
RestartCount	0-100		Dynamic	Connector must be in polling mode
RestartRetryCount	0-99	3	Dynamic	
RestartRetryInterval	A sensible positive value in minutes	1	Dynamic	
RHF2MessageDomain	mrm, xml	mrm	Component restart	Only if Delivery Transport is JMS and WireFormat is CwXML
SourceQueue	Valid WebSphere MQ name	CONNECTORNAME/SOURCEQUEUE	Component restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		CONNECTORNAME/ SYNCHRONOUSREQUESTQUEUE	Component restart	
SynchronousRequestTimeout	0 - any number (milliseconds)	0	Component restart	
SynchronousResponseQueue		CONNECTORNAME/ SYNCHRONOUSRESPONSEQUEUE	Component restart	
WireFormat	CwXML, CwBO	CwXML	Component restart	CwXML for WMQI and WAS; CwBO if Repository Directory is <REMOTE> (ICS)
WisfSynchronousRequest Timeout	0 - any number (milliseconds)	0	Component restart	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

WebSphere ICS only.

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, WMQI 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 `ASCII` for this property. If you previously configured the value of this property to `ascii7` or `ascii8`, you must reconfigure the connector to use either `ASCII` or one of the other supported values.

Important: By default only a subset of supported character encodings display 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 `asci i`.

ConcurrentEventTriggeredFlows

WebSphere ICS only.

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 `JMS`. It can also be set to `no value`.

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

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

You must also configure a data handler with the `MimeType`, `DHClass`, 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

WebSphere ICS only.

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

WebSphere ICS only.

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

DeliveryQueue

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

The default value is `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 WMQI is the broker type, the value may only be `JMS`.
- If WAS is the broker type, the value may only be `JMS`.

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

WebSphere MQ and IDL

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

- Asynchronous communication:
WebSphere MQ allows the application-specific component to poll and persistently store events even when the server is not available.

- Server side performance:
WebSphere MQ provides faster performance on the server side. In optimized mode, WebSphere MQ stores only the pointer to an event in the repository database, while the actual event remains in the WebSphere MQ queue. This saves having to write potentially large events to the repository database.
- Agent side performance:
WebSphere MQ provides faster performance on the application-specific component side. Using WebSphere MQ, the connector's polling thread picks up an event, places it in the connector's queue, then picks up the next event. This is faster than IDL, which requires the connector's polling thread to pick up an event, go over the network into the server process, store the event persistently in the repository database, then pick up the next event.

JMS

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

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

This property can also be set to `false`.

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

FaultQueue

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

The default value is `CONNECTORNAME/FAULTQUEUE`.

JvmMaxHeapSize

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

The default value is 128m.

JvmMaxNativeStackSize

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

The default value is 128k.

JvmMinHeapSize

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

The default value is 1m.

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

The default is `crossworlds.queue.manager`.

jms.NumConcurrentRequests

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

The default value is 10.

jms.Password

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

There is no default.

jms.UserName

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

There is no default.

ListenerConcurrency

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

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

Locale

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

A locale name has the following format:

ll_TT.codeset

where:

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

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

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

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

LogAtInterchangeEnd

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 integration broker is ICS and the Repository Directory is <REMOTE>.

Specifies whether the Object Activation Daemon (OAD) automatically attempts to restart the application-specific component after an abnormal shutdown. This property is required for automatic restart.

The default value is false.

OADMaxNumRetry

Valid only when the integration broker is ICS and the Repository Directory is <REMOTE>.

Specifies the maximum number of times that the OAD automatically attempts to restart the application-specific component after an abnormal shutdown.

The default value is 1000.

OADRetryTimeInterval

Valid only when the integration broker is ICS and the Repository Directory is <REMOTE>.

Specifies the number of minutes for the interval during which the OAD automatically attempts to restart the application-specific component after an

abnormal shutdown. If the application-specific component does not start within the specified interval, the OAD repeats the attempt as many times as specified in “OADMxNumRetry” on page 100.

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 `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 WMQI or WAS, this value must be set to <local directory>.

ResponseQueue

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

RestartCount

Causes the connector to shut down and restart automatically after it has processed a set number of events. You set the number of events in `RestartCount`. The connector must be in polling mode (set `PollFrequency` to "p") for this property to take effect.

Once the set number of events has passed through request processing, the connector is shut down and restarted the next time it polls.

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.

The default is 1.

RHF2MessageDomain

WebSphere MQ Integrator Broker only.

This property allows you to configure the value of the field domain name in the JMS header. When data is sent to WebSphere MQ Integrator Broker over JMS transport, the connector framework writes JMS header information, with a domain name and a fixed value of `mrm`. A configurable domain name enables users to track how WebSphere MQ Integrator 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

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 95.

The default value is SOURCEQUEUE.

SynchronousRequestQueue

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.

SynchronousResponseQueue

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.

SynchronousRequestTimeout

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 integration broker is WMQI or WAS, the setting is CwXML.
- if the integration broker is ICS and the value of RepositoryDirectory is <REMOTE>, the setting is CwB0.

WisfSynchronousRequest 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.

Appendix B. Connector Configurator

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

You use Connector Configurator to:

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

Note:

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

The topics covered in this appendix are:

- “Overview of Connector Configurator” on page 105
- “Starting Connector Configurator” on page 106
- “Creating a connector-specific property template” on page 107
- “Creating a new configuration file” on page 109
- “Setting the configuration file properties” on page 112
- “Using Connector Configurator in a globalized environment” on page 119

Overview of Connector Configurator

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

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator Broker (WMQI)
- WebSphere Application Server (WAS)

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see “Running Configurator in stand-alone mode” on page 106).

Each time you install a new adapter, you need to set up a **configuration file** for the connector. This file:

- Sets the standard and application-specific properties for the connector
- Designates which business objects and meta-objects it supports
- Sets the logging and tracing values that the connector will use at run time
- Sets the property values used by messaging and data handlers in the adapter
- Allows you to modify connector properties for an existing connector

You use Connector Configurator to create this configuration file and to modify its settings.

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

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

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

For **connector-specific properties**, however, you need first to define the properties and then set their values. You do this by creating a connector-specific property template for your particular adapter. There may already be a template set up in your system, in which case, you simply use that. If not, follow the steps in “Creating a new template” on page 107 to set up a new one.

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

Starting Connector Configurator

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

- Independently, in stand-alone mode (all brokers).
- From System Manager (ICS and WAS only).

Running Configurator in stand-alone mode

You can run Connector Configurator independently and work with connector configuration files, irrespective of your broker. However, if WMQI is your integration broker, you can only use Connector Configurator in stand-alone mode.

To do so:

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

If you are creating a configuration file for use with ICS or WAS as the broker, you may prefer to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see “Completing a configuration file” on page 111.)

Running Configurator from System Manager

If ICS or WAS is your integration broker, you can run Connector Configurator from System Manager.

To run Connector Configurator:

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

To edit an existing configuration file:

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

Creating a connector-specific property template

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

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

- To create a new template, see “Creating a new template” on page 107.
- To use an existing file, simply modify an existing template and save it under the new name.

Creating a new template

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

To create a template:

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

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

If you do not see any template that displays the connector-specific properties used by your connector, you will need to create one. Connector Configurator provides a template named **None**, containing no property definitions, as a default choice.

Specifying general characteristics

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

- **Edit properties**

Use the buttons provided (or right-click within the **Edit properties** display) to add a new property to the template, to edit or delete an existing property, or to add a child property to an existing property.

A child property is an attribute of another property, the parent property. The parent property can obtain simple values, or child properties, or both. These property relationships are hierarchical. When you create a configuration file from these properties, Connector Configurator will identify hierarchical property sets with a plus sign in a box at the left of any parent property.

- **Property type**

Choose one of these property types: Boolean, String, Integer, or Time.

- **Flags**

You can set **Standard Flags** (IsRequired, IsDeprecated, IsOverridden) or **Custom Flags** (for Boolean operators) to apply to this property.

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

Specifying values

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

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

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

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

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

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

Setting dependencies

When you have made your changes to the **General** and **Value** tabs, click **Next**. The **Dependencies** dialog box appears.

A dependent property is a property that is included in the template and used in the configuration file *only if* the value of another property meets a specific condition. For example, `Pol1Quantity` appears in the template only if `JMS` is the transport mechanism and `DuplicateEventElimination` is set to `True`.

To designate a property as dependent and to set the condition upon which it depends, do this:

1. In the **Available Properties** display, select the property that will be made dependent.
2. In the **Select Property** field, use the drop-down menu to select the property that will hold the conditional value.
3. In the **Condition Operator** field, select one of the following:
 - == (equal to)
 - != (not equal to)
 - > (greater than)
 - < (less than)
 - >= (greater than or equal to)
 - <=(less than or equal to)
4. In the **Conditional Value** field, enter the value that is required in order for the dependent property to be included in the template.
5. With the dependent property highlighted in the **Available Properties** display, click an arrow to move it to the **Dependent Property** display.
6. Click **Finish**. Connector Configurator stores the information you have entered as an XML document, under `\data\app` in the `\bin` directory where you have installed Connector Configurator.

Creating a new configuration file

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

To select a broker:

- In the Connector Configurator home menu, click **File>New>Connector Configuration**. The **New Connector** dialog box appears.
- In the **Integration Broker** field, select `ICS`, `WMQI` or `WAS` connectivity.

Note: To get the choice of WMQI, Connector Configurator must be launched from the Start menu, not from System Manager.

- Complete the remaining fields in the **New Connector** window, as described later in this chapter.

To create a new file for ICS or WAS, you can also do this:

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

Creating a configuration file from a connector-specific template

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

1. Click **File>New>Connector Configuration**.
2. The **New Connector** dialog box appears, with the following fields:
 - **Name**
Enter the name of the connector. Names are case-sensitive. The name you enter must be unique, must end with the word “connector”, and must be consistent with the file name for a connector that is installed on the system. For example, enter `PeopleSoftConnector` if the connector file name is `PeopleSoft.jar`.

Important: Connector Configurator does not check the spelling of the name that you enter. You must ensure that the name is correct.
 - **System Connectivity**
Click ICS or WMQI or WAS connectivity.
 - **Select Connector-Specific Property Template**
Type the name of the template that has been designed for your connector. The available templates are shown in the **Template Name** display. When you select a name in the Template Name display, the **Property Template Preview** display shows the connector-specific properties that have been defined in that template.
Select the template you want to use and click **OK**.
3. A configuration screen appears for the connector that you are configuring. The title bar shows the integration broker and connector names. You can fill in all the field values to complete the definition now, or you can save the file and complete the fields later.
4. To save the file, click **File>Save>to File** or **File>Save>Save to the project**. To save to a project, you must be using ICS or WAS as the broker, and System Manager must be running.
If you save as a file, the **Save File Connector** dialog box appears. Choose `*.cfg` as the file type, verify in the File Name field that the name is spelled correctly and has the correct case, navigate to the directory where you want to locate the file, and click **Save**. The status display in the message panel of Connector Configurator indicates that the configuration file was successfully created.

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

Using an existing file

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

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

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

To use an existing file to configure a connector, you must open the file in Connector Configurator, revise the configuration, and then save the file as a configuration file (`*.cfg` file).

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

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

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

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

Completing a configuration file

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

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

1. Under the **Standard Properties** tab, select the value field for the BrokerType property. In the drop-down menu, select the value ICS, WMQI, or WAS.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 114..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

Setting the configuration file properties

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

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

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

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

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

- Associated Maps
- Resources
- Messaging (where applicable)

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

Standard properties differ from connector-specific properties as follows:

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

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

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

Setting standard connector properties

To change the value of a standard property:

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

Setting application-specific configuration properties

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

1. Right-click in the top left portion of the grid. A pop-up menu bar will appear. Click **Add** to add a property or **Add Child** to add a child property to a property.
2. Enter a value for the property or child property.
3. To encrypt a property, select the **Encrypt** box.

4. Choose to save or discard changes, as described for “Setting standard connector properties” on page 113.

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

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

Encryption for connector properties

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

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

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

Update method

Connector properties are almost all static and the **Update Method** is **Component restart**. For changes to take effect, you must restart the connector after saving the revised connector configuration file.

Specifying supported business object definitions

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

For you to specify a supported business object, the business objects and their maps must exist in the system.

- Business object definitions and map definitions should be saved into System Manager projects if ICS or WAS is your integration broker.
- Business object definitions and MQ message set files should exist if WMQI is your integration broker.

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

If ICS is your broker

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

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

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

To delete a business object from the supported list:

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

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

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

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

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

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

For most connectors, Best Effort is the only possible choice, because most application APIs do not support the Stringent level.

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

If WMQI is your broker

The MQ message set files (*.set files) contain message set IDs that Connector Configurator requires for designating the connector's supported business objects. See *Implementing Adapters with WebSphere MQ Integrator Broker* for information about creating the MQ message set files.

Each time that you add business object definitions to the system, you must use Connector Configurator to designate those business objects as supported by the connector.

Important: If the connector requires meta-objects, you must create message set files for each of them and load them into Connector Configurator, in the same manner as for business objects.

To specify supported business objects:

1. Select the **Supported Business Objects** tab and click **Load**. The **Open Message Set ID File(s)** dialog box displays.
2. Navigate to the directory where you have placed the message set file for the connector and select the appropriate message set file (*.set) or files.
3. Click **Open**. The **Business Object Name** field displays the business object names contained in the *.set file. The numeric message set ID for each business object is listed in its corresponding Message Set ID field. Do not change the message set IDs. These names and numeric IDs are saved when you save the configuration file.
4. When you add business objects to the configuration, you must load their message set files. If you attempt to load a message set that contains a business object name that already exists in the configuration, or if you attempt to load a message set file that contains a duplicate business object name, Connector Configurator detects the duplicate and displays the **Load Results** dialog box. The dialog box shows the business object name or names for which there are duplicates. For each duplicate name shown, click in the **Message Set ID** field, and select the Message Set ID that you wish to use.

If WAS is your broker

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

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

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

Associated maps (ICS only)

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

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

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

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

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

- **Business Object Name**

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

- **Associated Maps**

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

- **Explicit**

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

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

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

To explicitly bind a map:

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

Resources (ICS)

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

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

Configuring messaging (ICS)

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

Setting trace/log file values

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

To change the logging and tracing values:

1. Click the **Trace/Log Files** tab.

2. For either logging or tracing, you can choose to write messages to one or both of the following:

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

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

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

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

Data handlers

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

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

Saving your configuration file

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

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

For ICS:

- From System Manager, as a file with a `*.con` extension in a an ICS User Project, or
- In a directory that you specify.
- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder, if you are using the file as a local configuration file.

For WMQI:

- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder.

For WAS:

- From System Manager, as a file with a `*.con` extension in a WAS User Project, or
- In a directory that you specify.
- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder.

After you have created the configuration file and set its properties, you need to deploy it to the correct location for your connector.

- If you are using ICS as your integration broker, save the configuration in a System Manager project, and use System Manager to load the file into ICS.
- If you are using WMQI as your integration broker, copy the configuration file to the correct location, which must match exactly the configuration file location specified in the startup file for your connector.
- If you are using WAS as your integration broker, save the file in a WAS user project. Use **File>Export** to create .wsdl files that you can then import into WSAD-IE.

You can also export the configuration file as a .jar file to a specified directory.

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

- For ICS: *Implementation Guide for WebSphere InterChange Server*
- For WMQI: *Implementing Adapters with WebSphere MQ Integrator Broker*
- For WAS: *Implementing Adapters with WebSphere Application Server*

Changing a configuration file

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

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

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

- Open the existing configuration file in Connector Configurator.
- Select the **Standard Properties** tab.
- In the **BrokerType** field of the Standard Properties tab, select the value that is appropriate for your broker.

When you change the current value, the available tabs and field selections on the properties screen will immediately change, to show only those tabs and fields that pertain to the new broker you have selected.

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

Appendix C. Connector feature list

This appendix details the features supported by the connector. For descriptions of these features, see “Appendix A: Connector Feature Checklist” in the *Connector Development Guide for Java*.

Business object request handling features

Table 25 details the business object request handling features supported by the connector.

Table 25. Business object request handling features

Category	Feature	Support	Notes
Create	Create verb	Full	
Delete	Delete verb	Full	
	Logical delete	N/A	Siebel applications do not support logical deletes.
Exist	Exist verb	Full	The check is made to see if the primary business component (the first child in a business object) exists in the Siebel database.
Misc	Attribute names	Full	
	Business object names	Full	
Retrieve	Ignore missing child object	N/A	
RetrieveByContent	Ignore missing child object	N/A	
	Multiple results	Full	
Update	RetrieveByContent verb	Full	The non-key values are set only on the parent.
	After-image support	Full	
	Delta support	Partial	This is done using DeltaUpdate verb support.
Verbs	KeepRelations	Full	The DeltaUpdate verb is used in place of KeepRelations.
	Retrieve verb	Full	
	Subverb support	Partial	This support is for the Update operation on the top level business object. On an update, the child verbs are processed independently of the parent verb set.
	Verb stability	Full	

Event notification features

Table 26 details the event notification features supported by the connector.

Table 26. Event notification features

Category	Feature	Support	Notes
Connector Properties	Event distribution	Full	
	PollQuantity	Full	
Event Table	Event status values	Full	READY_FOR_POLL = 0 IN_PROGRESS = 1 UNSUBSCRIBED = 2 SUCCESS = 3 ERROR_PROCESSING_EVENT = -1 ERROR_POSTING_EVENT = -2 ERROR_OBJECT_NOT_FOUND = -3
	Object key	Full	
	Object name	Full	

Table 26. Event notification features (continued)

Category	Feature	Support	Notes
Misc.	Priority	Full	
	Archiving	Full	
	CDK method gotApplEvent	Full	
	Delta event notification	No	
	Event sequence	Full	The events are retrieved using the search criteria on status and connector Id and sorted using timestamp and priority.
	Future event processing	No	
	In-Progress event recovery	Full	With this connector, the event is reprocessed. The connector property InDoubtEvents is set to Reprocess.
	Physical delete event	Full	
	RetrieveAll	Full	
	Smart filtering	Full	
Verb stability	Full		

General features

Table 27 details the general features supported by the connector.

Table 27. General features

Category	Feature	Support	Notes
Business Object Attributes	Foreign key	No	
	Foreign Key attribute property	N/A	
	Key	Full	For simple attributes, if Key is set to true, the attribute is used with Update to update a specific record in Siebel. With Retrieve, these attributes are used in the search specification to get the record from Siebel. During Delete, the keys are set on the top-level business object.
	Max Length	No	
	Meta-data-driven design	Full	
	Required	Full	If any attribute or child object in a WebSphere business integration system business object is marked as required, the value needs to be present.
	Connection Lost		
Connection Lost	Connection lost on poll	Full	
	Connection lost on request processing	Full	Using the connectorErrors connector property.
	Connection lost while idle	No	
Connector Properties	ApplicationPassword	Full	
	ApplicationUserName	Full	
	UseDefaults	Full	
Message Tracing	General messaging	Full	
	generateMsg()	No	
	Trace level 0	Full	
	Trace level 1	Full	
	Trace level 2	Full	
	Trace level 3	Full	

Table 27. General features (continued)

Category	Feature	Support	Notes
Misc.	Trace level 4	Full	
	Trace level 5	Full	
	CDK method LogMsg	Full	
	Java Package Names	Full	
	Logging messages	Unknown	
	NT service compliance	Full	
Special Value	Transaction support	Full	
	CxBlank processing	No	
	CxIgnore processing	Full	In a Create operation, if the attribute is a required attribute, UseDefaults is true, and there is a default value for the attribute, this value is set in the attribute even if the inbound is CxIgnore. Thus, the value in the application changes.

Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

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

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

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

Other company, product or service names may be trademarks or service marks of others. IBM WebSphere InterChange Server V4.2.1, IBM WebSphere Business Integration Toolset V4.2.1, IBM WebSphere Business Integration Adapters V2.3.1, IBM WebSphere Business Integration Collaborations V4.2.





Printed in USA