

WebSphere Business Integration Server
Express and Express Plus



Adapter for Siebel eBusiness Applications User Guide

Version 4.3

WebSphere Business Integration Server
Express and Express Plus



Adapter for Siebel eBusiness Applications User Guide

Version 4.3

Note!

Before using this information and the product it supports, read the information in "Notices" on page 99

14May2004

This edition of this document applies to IBM WebSphere Business Integration Server Express Plus, version 4.3, 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 2004. 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	vi
New in this release	vii
New in release 4.3	vii
Chapter 1. Overview of the Adapter	1
Definition of the connector	1
Siebel application architecture	1
Connector architecture	2
How the connector works	2
Chapter 2. Installing and configuring the connector	7
Prerequisites	7
Installing the adapter	8
Installed files the connector uses	8
Event and archive tables	8
Configuring the connector	14
Creating multiple connector instances	18
Starting the connector	19
Stopping the connector	20
Chapter 3. Understanding business objects	21
Business object structure and relationships	21
Business object application-specific information	23
Using business object samples	32
Chapter 4. Generating business object definitions using SiebelODA	37
Installation and usage	37
Using SiebelODA in Business Object Designer Express	39
Contents of the generated definition	51
Adding information to the business object definition	53
Chapter 5. Using the Siebel connectivity DLL	55
Connector architecture and processes	55
Configuring the Siebel connectivity DLL	57
Communicating with the Siebel connectivity DLL	59
Generating IBM business objects	59
Configuring the IBM generic business services	60
Chapter 6. Supporting Siebel business services	63
Understanding business services	63
Verb processing with business services	65
Events detection with business services	66
Custom business service support	66
Chapter 7. Troubleshooting	69
Handling errors	69
Decreasing the size of the Siebel log file	70
Appendix A. Standard configuration properties for connectors	71

Configuring standard connector properties	71
Summary of standard properties	72
Standard configuration properties	75
Appendix B. Connector Configurator Express	85
Overview of Connector Configurator Express	85
Starting Connector Configurator Express	86
Running Configurator Express from System Manager	86
Creating a connector-specific property template	86
Creating a new configuration file	89
Using an existing file	90
Completing a configuration file.	91
Setting the configuration file properties	91
Saving your configuration file	96
Completing the configuration	96
Using Connector Configurator Express in a globalized environment	96
Notices	99
Programming interface information	100
Trademarks and service marks	100

About this document

The products IBM^(R)WebSphere Business Integration Server Express and IBM^(R)WebSphere Business Integration Server Express Plus are made up of the following components: InterChange Server Express, the associated Toolset Express, CollaborationFoundation, and a set of software integration adapters. The tools in Toolset Express help you to create, modify, and manage business processes. You can choose from among the prepackaged adapters for your business processes that span applications. The standard processes template--CollaborationFoundation--allows you to quickly create customized processes.

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

Audience

This document is for WebSphere Business Integration Server Express Plus 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

Related documents

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

You can download, install, and view the documentation at the following site:
<http://www.ibm.com/websphere/wbiserverexpress/infocenter>

Note: Important information about this product may be available in Technical Support Technotes and Flashes issued after this document was published. These can be found on the WebSphere Business Integration Support Web site, <http://www.ibm.com/software/integration/websphere/support/>. Select the component area of interest and browse the Technotes and Flashes sections.

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

This is the first release of this guide.

Chapter 1. Overview of the Adapter

This chapter provides an overview of the Adapter for Siebel eBusiness Applications component of the IBM WebSphere Business Integration Server Express Plus and contains the following sections:

- “Definition of the connector”
- “Siebel application architecture”
- “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 6

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 InterChange Server Express and the application-specific component. The application-specific component contains code tailored to a particular application. The connector framework provides the following services between InterChange Server Express 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 Server Express Plus Adapter 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.

Connector architecture

The connector has been designed following the meta-data design principles. 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. The integration broker is InterChange Server Express.

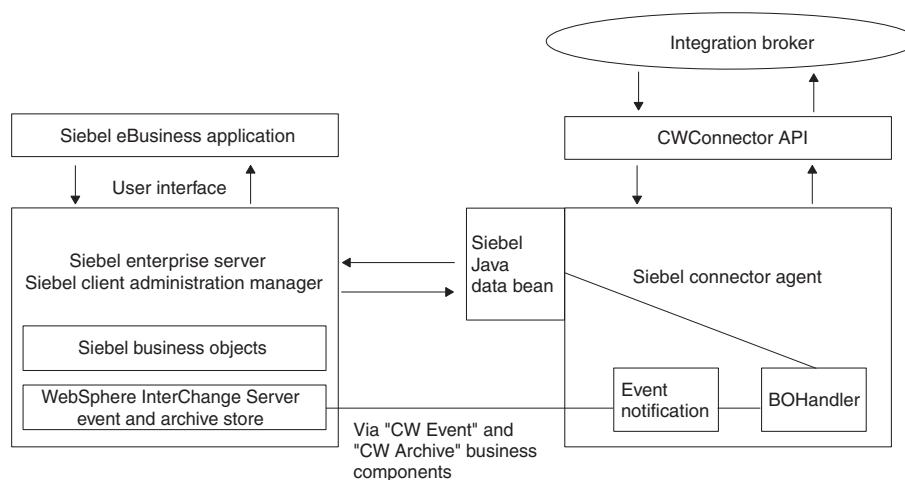


Figure 1. Siebel connector architecture

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, "Supporting Siebel business services," on page 63

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. The integration broker for WebSphere Business Integration Server Express Plus is InterChange Server Express.

Processing InterChange Server Express 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 InterChange Server Express asks the connector to retrieve a hierarchical business object from the Siebel application, the connector attempts to return a business object to InterChange Server Express 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 InterChange Server Express 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 InterChange Server Express to recursively descend through the corresponding data in the database.

Business object RetrievalByContent: When InterChange Server Express 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 InterChange Server Express 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 InterChange Server Express 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 Event and 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
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

Table 1. Events business component structure (continued)

Fields	Description
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 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 InterChange Server Express.

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 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 InterChange Server Express.

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 Event business component), and sends it with the Delete verb to InterChange Server Express. 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 Event business component and a 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 Event business component until it has been inserted it into the Archive business component.

The connector polls the 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 Archive business component after updating its status. For more information on the ArchiveProcessed property, see "Configuring the connector" on page 14

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 Events business component and archived with status of Success
	Unsuccessful No subscription for business object	Archived with status of Error Event is deleted from the 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 Events business component with a status of Success
	Unsuccessful	Event is not archived and remains in the 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 Event and Archive business components
- Retrieve the business object that is related to the event
- Create or update a record pertaining to a business object.

Chapter 2. Installing and configuring the connector

This chapter describes how to install and configure the Adapter for Siebel eBusiness Applications component of the WebSphere Business Integration Server Express Plus. The following topics are covered:

- “Prerequisites”
- “Installing the adapter” on page 8
- “Installed files the connector uses” on page 8
- “Event and archive tables” on page 8
- “Configuring the connector” on page 14
- “Starting the connector” on page 19

Note: This chapter includes references to Event and Archive business components, business objects, and tables. These references are synonymous with references to CW Event and CW Archive that appear in earlier versions, and with references to IBM Event and IBM Archive that appear in Siebel 7.5

Prerequisites

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

- Install the Siebel 6.2.x, Siebel 7.x, or Siebel 7.5 .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 7x or 7.5

- SiebelJI_Common.jar
- SiebelJI_enu.jar

Installing the adapter

For information on installing the adapter, see the discussion of installing from the Adapter Capacity Pack for WebSphere Business Integration Server Express Plus, in the *Installation Guide for WebSphere Business Integration Server Express*, located in the WebSphere Business Integration Server Express Infocenter at the following site:

<http://www.ibm.com/websphere/wbiserverexpress/infocenter>

Installed files the connector uses

The following section describes the file structures of an installed connector on a Windows 2000 system.

Installed files on a Windows 2000 system

Table 3 describes the Windows file structure used by the connector.

Table 3. 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
\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.
\bin	Contains the WBIA.jar file.
	Contains the CWConnEnv.bat file.

Installer adds an icon for the connector file to the WebSphere Business Integration ServerExpress Plus menu. For a fast way to start the connector, create a shortcut to this file on the desktop.

Note: For more information on installing adapters, refer to the *Installation Guide for WebSphere Business Integration Server Express*.

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 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 in Siebel, version 7.5

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

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

2. Apply the six patch files in the following order to your local database:
 - `ibmtable.sif`
 - `ibmview.sif`
 - `ibmapplet.sif`
 - `ibmbo.sif`
 - `ibmbc.sif`
 - `ibmcreen.sif`

When you apply WebSphere Business Integration Server Express Plus 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.

3. When you are prompted, lock the IBM Audit project on your local database.
4. Ensure that the following have been created:
 - Two new tables, CX_IBM_Archive_Q and CX_IBM_Event_Q
 - One new business object, IBM Events
 - One new business object, Schema version
 - Two new business components, IBM Archive and IBM Events
 - One new view, IBM Event List View
 - Two new applets, IBM Archive List Applet and IBM Event List Applet
 - One new screen IBM Events and one new screen view, IBM Event List view
5. 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 IBM 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.
 - f. Go to the Page tab locale and create a new record for IBM Events. Add ENU for the Language Code and IBMEvents for text, if it does not exist.
6. 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 IBM 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.
 - f. Go to the screen menu item locale and create a new record for IBMEvents. Add ENU for language and IBMEvents for text, if it does not exist.
7. 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 multiple connectors, set and activate the Connector Id in the VB scripts.
8. Apply the physical schema for the new tables to your local database. You can do this by querying for the two new tables, CX_IBM_ARCHIVE_Q and CX_IBM_EVENT_Q, and selecting the current query to create a physical schema. Make sure that you leave the table space and index space blank.
9. Activate the new schema using the activate button.

10. Compile the updated and locked projects on your local database to create a new Siebel repository (.srf) file.
11. Open Siebel Sales Enterprise on your local database. You must have administrative privileges to perform the following:
 - a. .Create a new view called IBM Event List View. Tip: Copy the view name from tools and paste it into the View Name field.
 - b. .Create a new responsibility called IBM Responsibility for IBM Event List View.
 - c. .Add the employees or teams who are responsible for reviewing events to the newly created IBM Responsibility.
 - d. .Create the CWCONN user and add it to IBM Responsibility and Administrative Responsibility.
12. Test the application in your local environment. Ensure that you have visibility to IBM 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 IBM Event List View.
13. Check in the following updated and locked projects to your development server.
 - IBM Audit
 - 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.

14. Apply the physical schema to your development database. You can do this by querying for the two new tables, CX_IBM_ARCHIVE_Q and CX_IBM_EVENT_Q, and select the current query to create a physical schema. Make sure that you leave the table space and index space blank.
15. Activate the queried tables in the development database.
16. Move to test and production environments accordingly.
17. 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.

Creating the event and archive tables in Siebel, versions below 7.5

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
 - 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:

- cwtable.sif
- cwview.sif
- cwapplet.sif
- cwbo.sif
- cwbc.sif
- cwdo.sif
- cwscreen.sif
- schemabo.sif

When you apply WebSphere Business Integration Server Express Plus 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, Events
 - One new business object, schema version
 - Two new business components, Archive and Events
 - One new view, Event List View
 - Two new applets, Archive List Applet and Event List Applet
 - One new screen Events and one new screen view, 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 on page 13.

- 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 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 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 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 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 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 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 if it is idle for more than 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 Express, which is accessed from System Manager. For detailed configuration information, see Appendix B, "Connector Configurator Express," on page 85.

Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, "Standard configuration properties for connectors," on page 71 for documentation of these properties.

Table 4 provides information specific to this connector about configuration properties in the appendix.

Table 4. Property information specific to this connector

Property	Note
Character Encoding	This connector does not use this property.
Locale	Because this connector has not been internationalized, you cannot change the value of this property.

Note: Ensure that every component of your installation (for example, all adapters, applications, and InterChange Server Express itself) is set to the same locale.

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

Table 5. 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
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
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 connector.	
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.	

Table 5. Connector-specific configuration properties (continued)

Name	Meaning	Possible values	Default value
DataBeanPoolSize	Indicates the maximum number of beans in the data bean pool.	An integer determining the bean pool size.	
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	
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
EventProcessingSupport	Indicates whether the adapter processes the event or not. Can be used to switch off subscription services if necessary.	Boolean	True
SiebelVersion	Allows the adapter to run against a specified version of Siebel without accessing the SchemeVersion Siebel business object to obtain the version. Use of the default value is recommended.	6, 7, or NONE	NONE

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 Archive business component after they are deleted from the 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 Event business component.
- If the connector does not subscribe to the event's business object, the connector leaves the event in the 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.

EventProcessingSupport

If EventProcessingSupport is set to true, the adapter processes an event. If EventProcessingSupport is set to false, the adapter does not process the event.

The default value is true.

SiebelVersion

Enables the adapter to run against specified versions of Siebel without accessing the Siebel business object Schema Version to obtain the version. Set to 6 for Siebel version 6, or to 7 for Siebel version 7.

The default value is NONE. When you use the default value, the adapter obtains the Siebel version from Schema Version. Using the default value is recommended.

Creating multiple connector instances

Creating multiple instances of a connector is in many ways the same as creating a custom connector.

Note: Each additional instance of any adapter supplied with WebSphere Business Integration Server Express will be treated as a separate adapter by the function that limits the total number of adapters that can be deployed.

You can set your system up to create and run multiple instances of a connector by following the steps below. You must:

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

Create a new directory

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

`ProductDir\connectors\connectorInstance`

where connectorInstance uniquely identifies the connector instance.

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

`ProductDir\repository\connectorInstance`

Create business object definitions

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

1. If you need to modify business object definitions that are associated with the initial connector, copy the appropriate files and use Business Object Designer

Express to import them. You can copy any of the files for the initial connector. Just rename them if you make changes to them.

- Files for the initial connector should reside in the following directory:
ProductDir\repository\initialConnectorInstance

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

Create a connector definition

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

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

Create a start-up script

To create a startup script:

- Copy the initial connector's startup script and name it to include the name of the connector directory:
dirname
- Put this startup script in the connector directory you created in "Create a new directory" on page 18.
- Create a startup script shortcut.
- Copy the initial connector's shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

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

Starting the connector

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

ProductDir\connectors\connName

where *connName* identifies the connector. The name of the startup script depends on the operating-system platform, as Table 6 shows.

Table 6. Startup scripts for a connector

Operating system	Startup script
Windows	start_connName.bat

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

- On Windows systems, from the **Start** menu

Select **Programs>IBM WebSphere Business Integration Express>Adapters>Connectors>your_connector_name**.

By default, the program name is "IBM WebSphere Business Integration Express". However, it can be customized. Alternatively, you can create a desktop shortcut to your connector.

- From the command line
 - On Windows systems:
`start_connName connName ICS [-cconfigFile]`
where *connName* is the name of the connector.
- From System Monitor
You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector starts when the Windows system boots (for an Auto service) or when you start the service through the Windows Services window (for a Manual service).

For more information on how to start a connector, including the command-line startup options, refer to the *System Administration Guide*.

Stopping the connector

The way to stop a connector depends on the way that the connector was started, as follows:

- If you started the connector from the command line, with its connector startup script:
 - On Windows systems, invoking the startup script creates a separate “console” window for the connector. In this window, type “Q” and press Enter to stop the connector.
- From System Monitor
You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector stops when the Windows system shuts down.

Chapter 3. 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 23
- “Using business object samples” on page 32

Note: This chapter includes references to Event and Archive business components, business objects, and tables. These references are synonymous with references to CW Event and CW Archive that appear in earlier versions, and with references to IBM Event and IBM Archive that appear in Siebel 7.5

Business object structure and relationships

The connector supports Create, Retrieve, Update, Delete, Exists, Retrieve By Content, and DeltaUpdate 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 correspond 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 7. 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 8. 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 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.

Parameter	Description
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 Server Express Plus 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_BCOppportunity 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 Server Express Plus 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=Oppportunity;PL=true;From=OppportunityId;To=OppportunityId

```

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.

Multiple record retrieval

The adapter processes the RetrievebyContent verb similarly to the way it processes the Retrieve verb, except that it does not check to see that all keys are set on the inbound IBM business object.

The Adapter for Siebel eBusiness Applications supports wrapper business objects. See “Business object structure and relationships” on page 21. The adapter can retrieve multiple records during request processing using the RetrievebyContent verb and a wrapper business object. For a wrapper business object with a multiple cardinality container, all matching records are returned. For a single cardinality child, only one record is processed and MULTIPLE_HITS is returned.

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 32
- “Siebel_BCQuote” on page 33
- “Siebel_BCContact” on page 34
- “Siebel_BCInternalProduct” on page 34
- “Siebel_BCAsset” on page 35

Siebel_BCAccount

1. Locate the following Siebel VB files:

Siebel 6

Account_Write.svb, Account_PreDelete.svb, Business_Address_PreDelete.svb, 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 Server Express Plus system object.)
 - Select Name and set the value to Not Required.
 - Select Product Id and set the value to Required.
- Note:** WebSphere Business Integration Server Express Plus components track 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 Express” on page 39
- “Contents of the generated definition” on page 51
- “Adding information to the business object definition” on page 53

Installation and usage

This section discusses the following:

- “Installing SiebelODA” on page 37
- “Before using SiebelODA” on page 37
- “Launching the SiebelODA” on page 38
- “Running SiebelODA on multiple machines” on page 38
- “Working with error and trace message files” on page 38

Installing SiebelODA

To install SiebelODA, use the WebSphere Business Integration Server Express Adapter Installer. Follow the instructions in the *Installation Guide for WebSphere Business Integration Server Express*. 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
- bin\CWODAEV.bat

Note: Except as otherwise noted, this document uses backslashes (\) as the convention for directory paths. All WebSphere Business Integration Server Express Plus 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.x

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 Express.
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 this startup script.

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 Express > Adapters > Object Discovery Agents.

You configure and run SiebelODA using Business Object Designer Express. Business Object Wizard, which Business Object Designer Express 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 runs on a unique port.

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 Express as part of ODA configuration. Business Object Designer Express 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 Express 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 10 on page 41.

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 9 describes these values.

Table 9. 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	<ul style="list-style-type: none"> • 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 10 on page 41.

Using SiebelODA in Business Object Designer Express

This section describes how to use Business Object Designer Express to generate business object definitions using SiebelODA. For information on launching Business Object Designer Express, see the *Business Object Development Guide*. Business Object Designer Express 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 Express 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 Express.
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 illustrates this window.

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

1. "Select the ODA"
2. "Specify configuration properties" on page 41
3. "Selecting the source" on page 42
4. "Confirm selection of objects" on page 46
5. "Generate the business object definition" on page 47 and, optionally, "Provide additional information" on page 48
6. "Save the business object definition" on page 51

Select the ODA

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

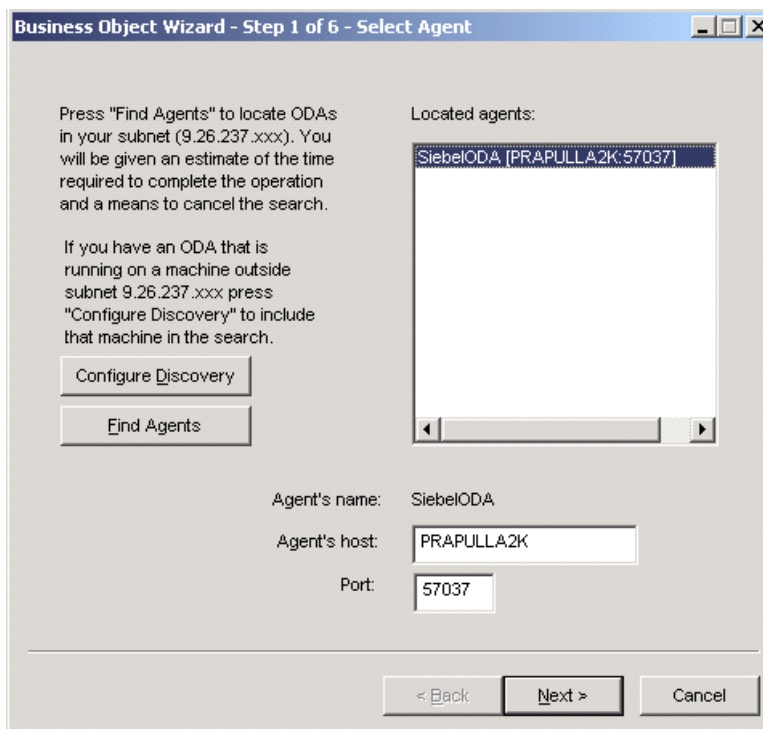


Figure 2. Business Object Wizard, Select ODA screen

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 4.

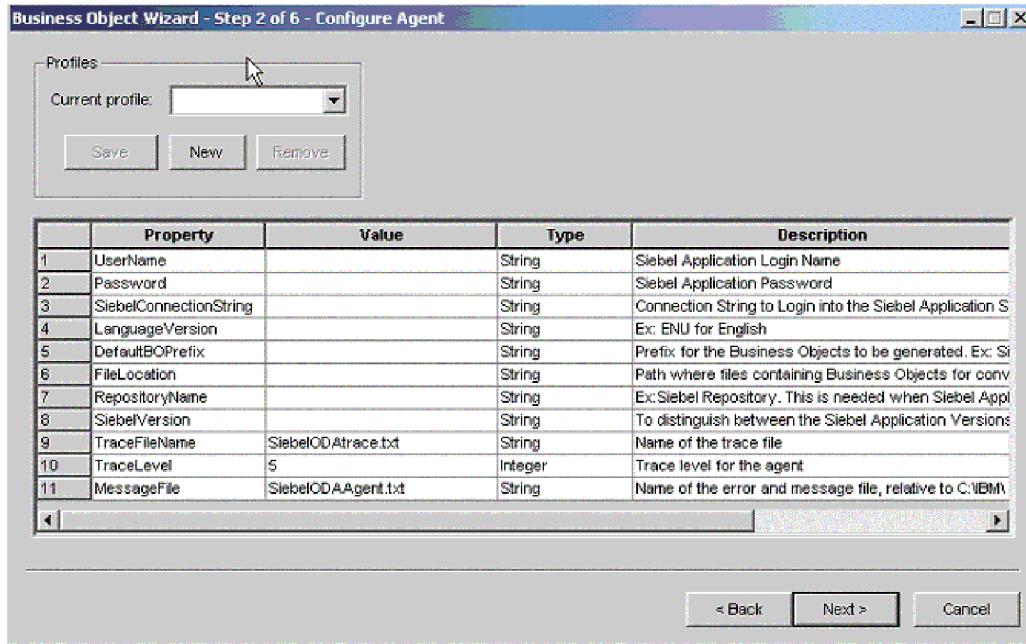


Figure 3. Business Object Wizard, Configure Agent screen

Configure the SiebelODA properties described in Table 10.

Table 10. 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.x: //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.
6	FileLocation	String	The absolute path containing the files with previous versions of business object definitions. For example, in Windows, the path is C:\SiebelBos.
7	RepositoryName	String	The name of the Siebel repository in the Siebel application.

Table 10. SiebelODA configuration properties (continued)

Row number	Property name	Property type	Description
8	SiebelVersion	String	Identifies the Siebel Application version. For Siebel version 6.x, this property must be set to 6.x. For Siebel version 7.x, do not set.
9	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.
10	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 38.
11	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 Express represents a non-existent file. If the name is not correct when you move forward from this dialog box, Business Object Designer Express displays an error message in the window from which the ODA was launched. This message does not pop up in Business Object Designer Express. 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 43).

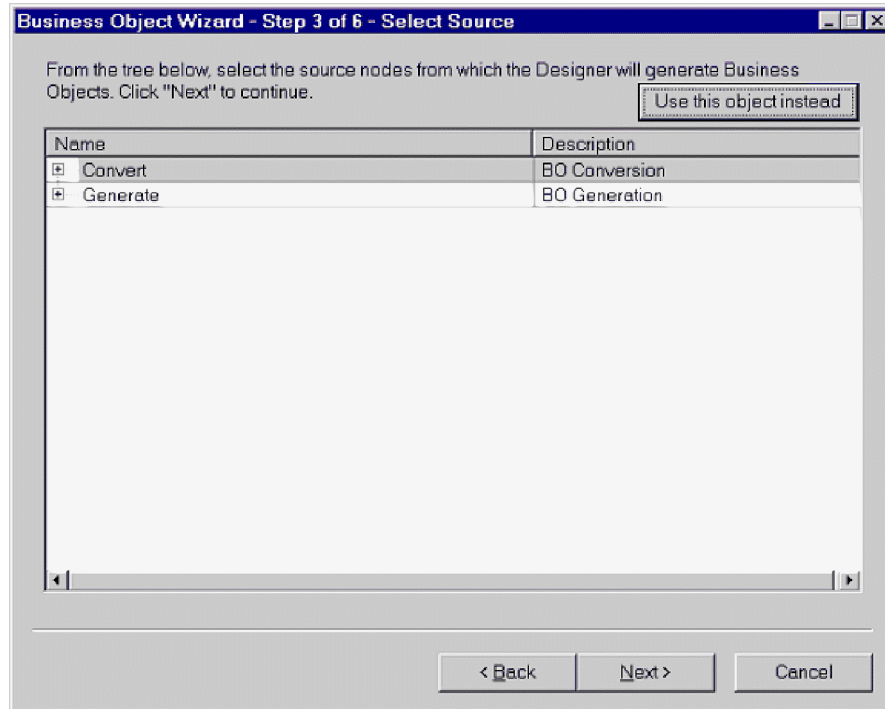


Figure 4. Business Object Wizard, Select Source screen

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).

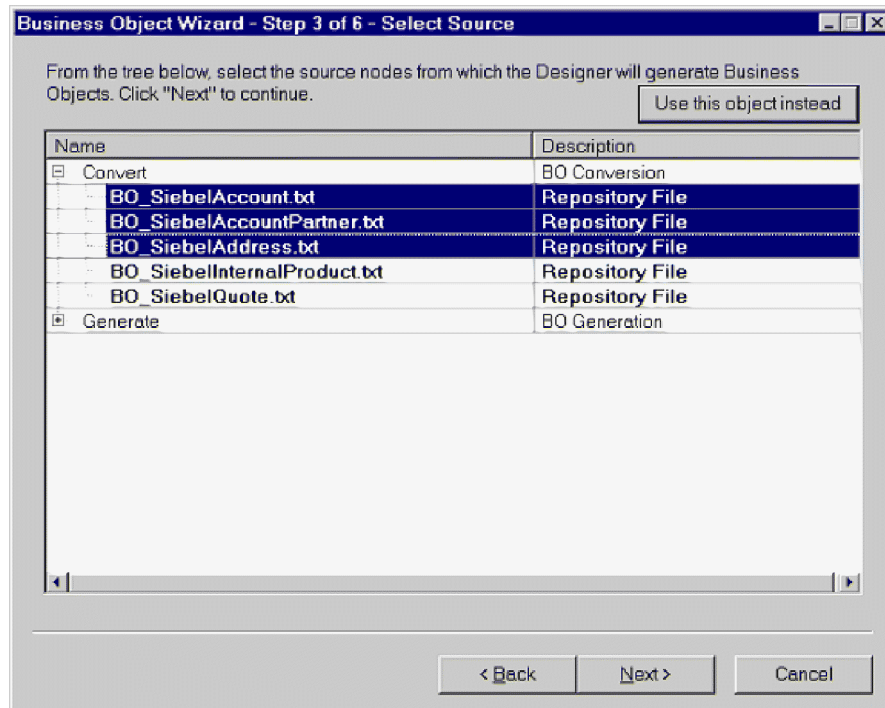


Figure 5. Business Object Wizard, screen displaying business objects to be converted

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 45, and Figure 8 on page 45. 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.

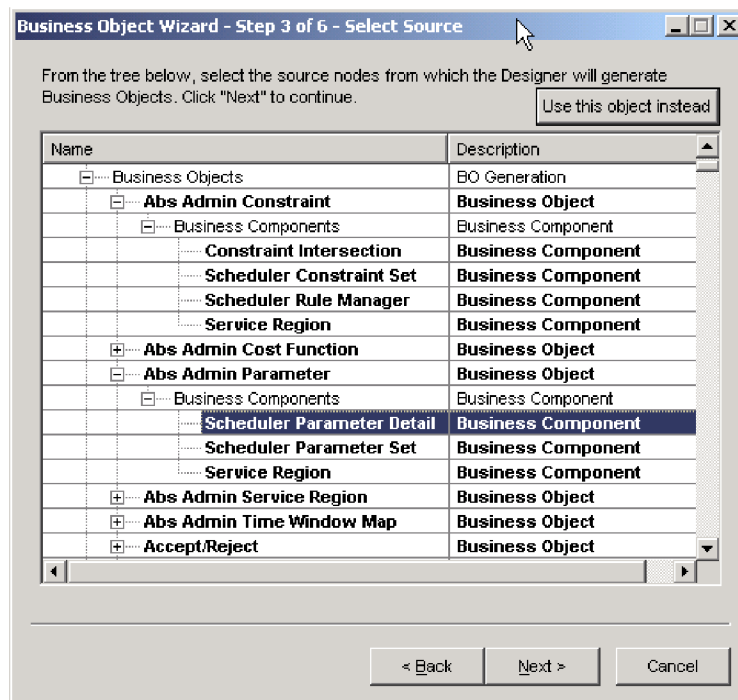


Figure 6. Business Object Wizard, displaying Business Objects expanded

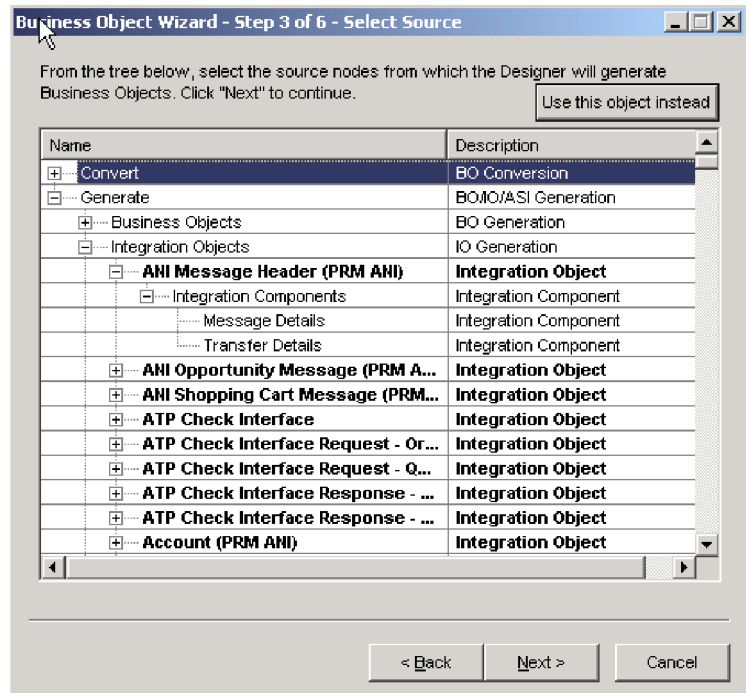


Figure 7. Business Object Wizard, displaying Integration Objects expanded

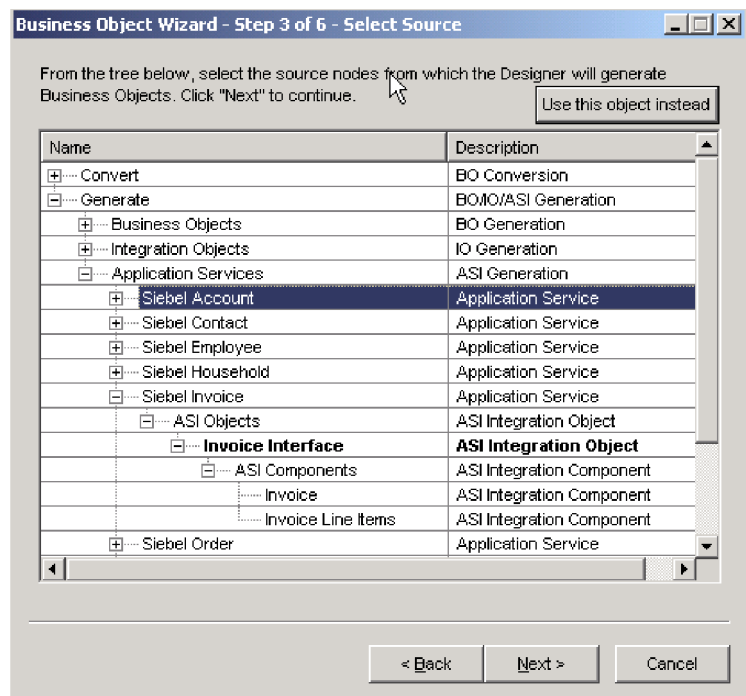


Figure 8. Business Object Wizard, displaying Application Services expanded

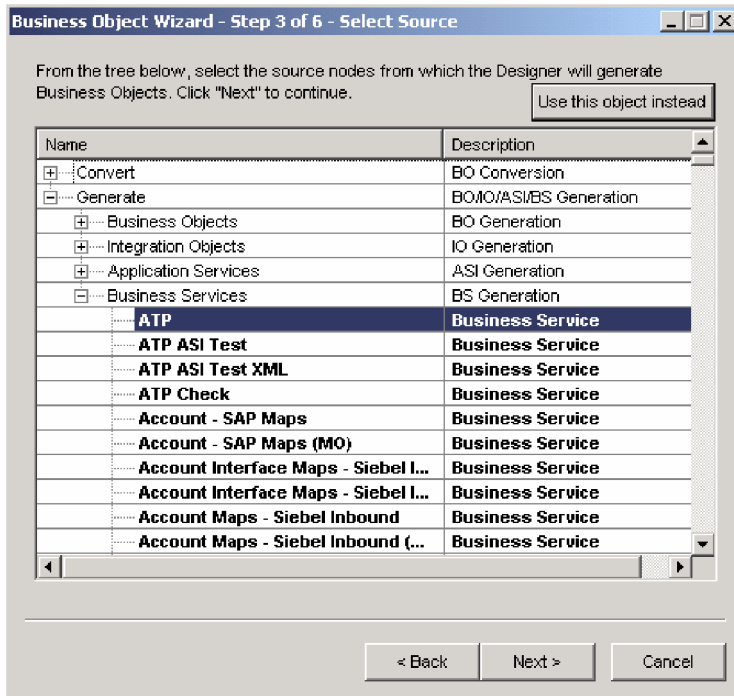


Figure 9. Business Object Wizard, displaying Business Services

Confirm selection of objects

After you identify all the Siebel elements to be associated with the generated business object definitions, Business Object Designer Express displays the dialog box with only the selected objects and components. Figure 10 on page 47 illustrates this dialog box.

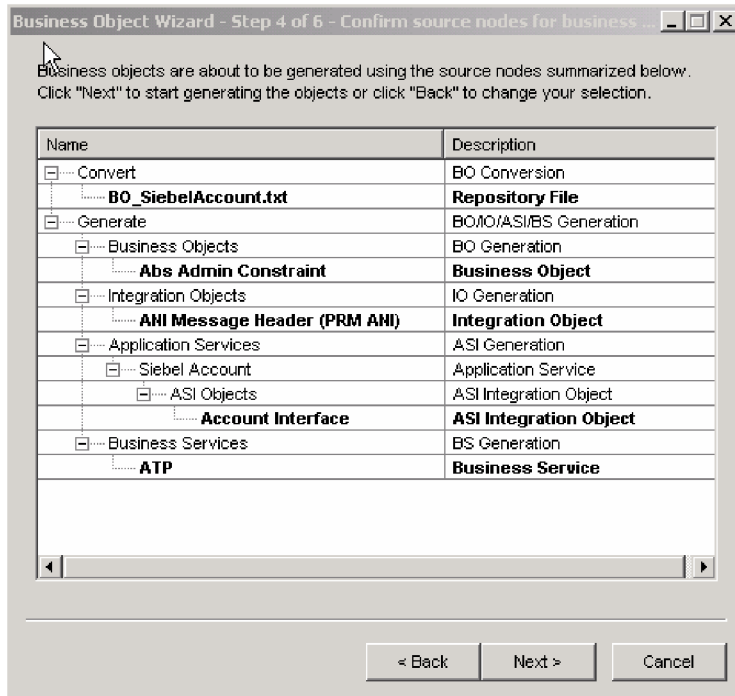


Figure 10. Business Object Wizard, confirming selecting of objects and components

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 Express is generating the business object definition. If a large number of Component Interfaces has been selected, this generation step can take time.

Figure 11 on page 48 illustrates this dialog box.

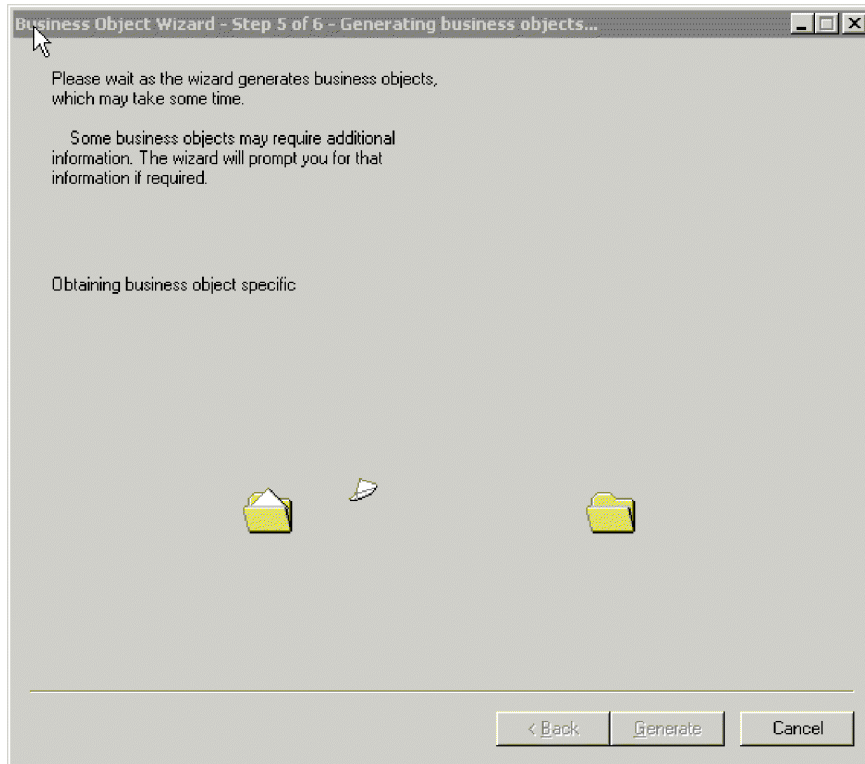


Figure 11. Generating the business object definitions

Provide additional information

Because SiebelODA needs additional information about the verbs, Business Object Designer Express 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 12 on page 49 illustrates these screens.

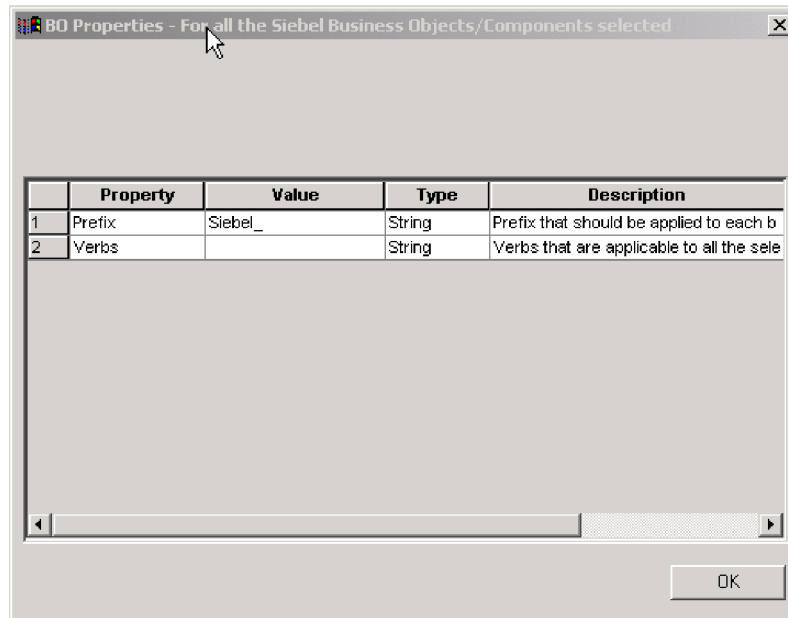


Figure 12. Providing additional information for business object

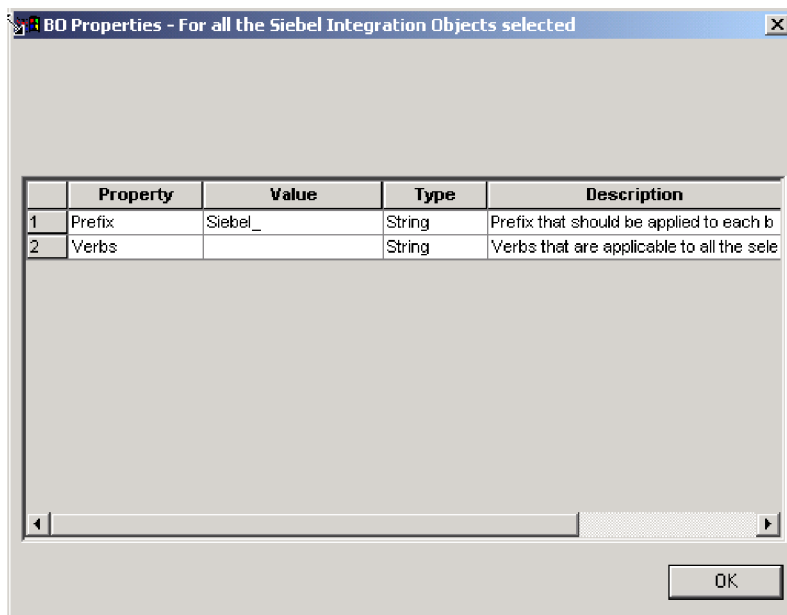


Figure 13. Providing additional information for integration object

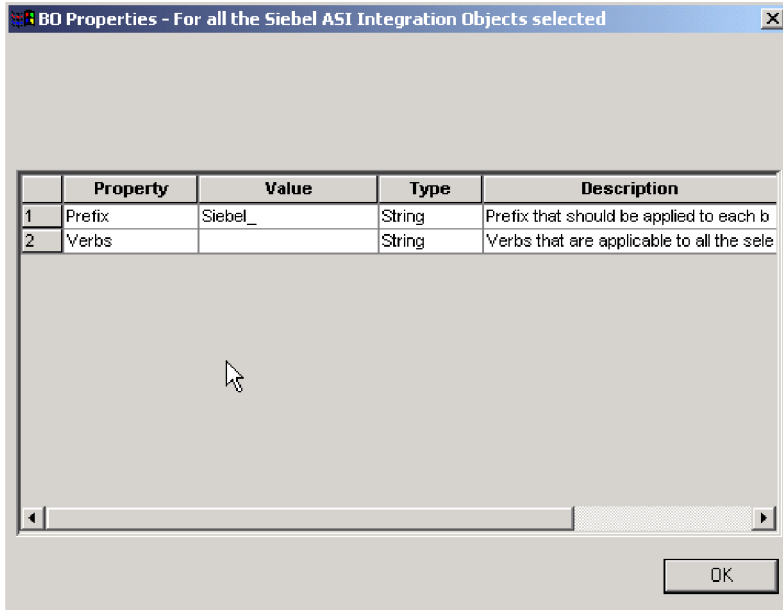


Figure 14. Providing additional information for application service

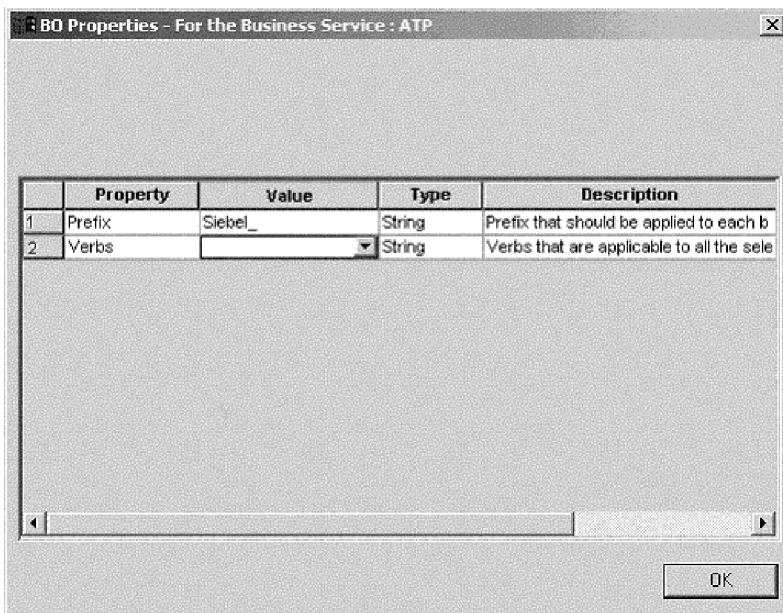


Figure 15. Providing additional information for a custom-written business service

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.

Save the business object definition

After you provide all required information in the BO Properties dialog box and click OK, Business Object Designer Express 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.
- Save the business object definition to a file.
- Open the business object definition for editing in Business Object Designer Express.

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

Figure 16 illustrates this dialog box.

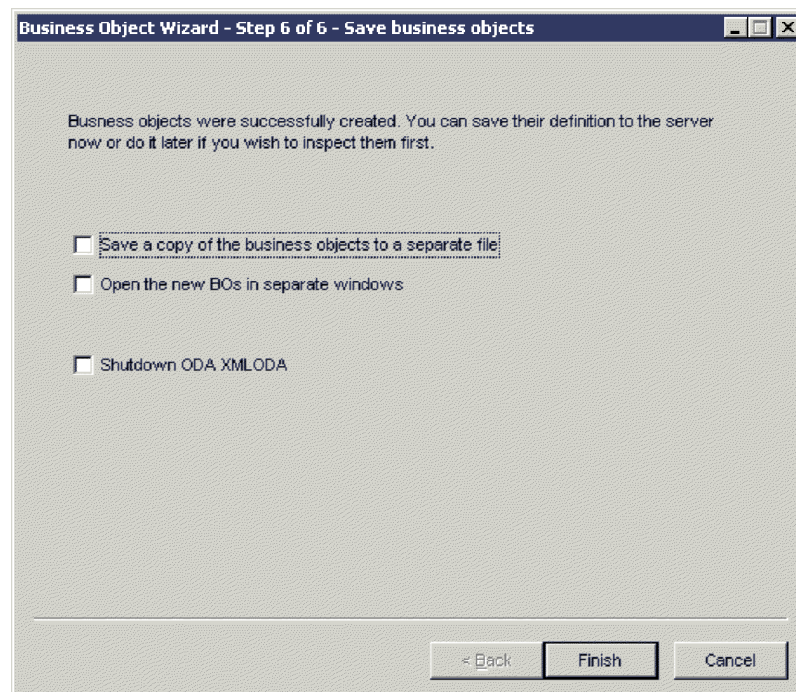


Figure 16. Saving the business object definition

Contents of the generated definition

The business object definition that SiebelODA generates contains:

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

Note: When the SiebelODA generates business objects for Siebel business objects or components, the application-specific information may or may not be generated for container attributes. You must check the generated business objects to see if they contain the application-specific information.

When the SiebelODA generates business objects for Siebel integration objects or components, or business services, the generated business object does contain application-specific information for all attributes, including container attributes.

This section describes:

- “Business-object-level properties” on page 52
- “Attribute properties” on page 52
- “Verbs” on page 53

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 11. This is only in the case of business object generation, since business object conversion is for existing business objects.

Table 11. 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 11, 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 Express. You can also use the `repos_copy` command to load the definition into the repository.

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 InterChange Server Express integration broker. The WebSphere Integration Adapter Suite consists of adapters for various applications, hence, InterChange Server Express and the appropriate WebSphere Business Integration Server Express Plus adapters can process the Siebel generated request sent by the DLL. The DLL is unidirectional; requests sent from Siebel to the broker (InterChange Server Express) 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 57
- “Communicating with the Siebel connectivity DLL” on page 59
- “Generating IBM business objects” on page 59
- “Configuring the IBM generic business services” on page 60

Note: This chapter may include references to Event and Archive business components, business objects, and tables. These references are synonymous with references to CW Event and CW Archive that appear in earlier versions, and with references to IBM Event and IBM Archive that appear in Siebel 7.5

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.

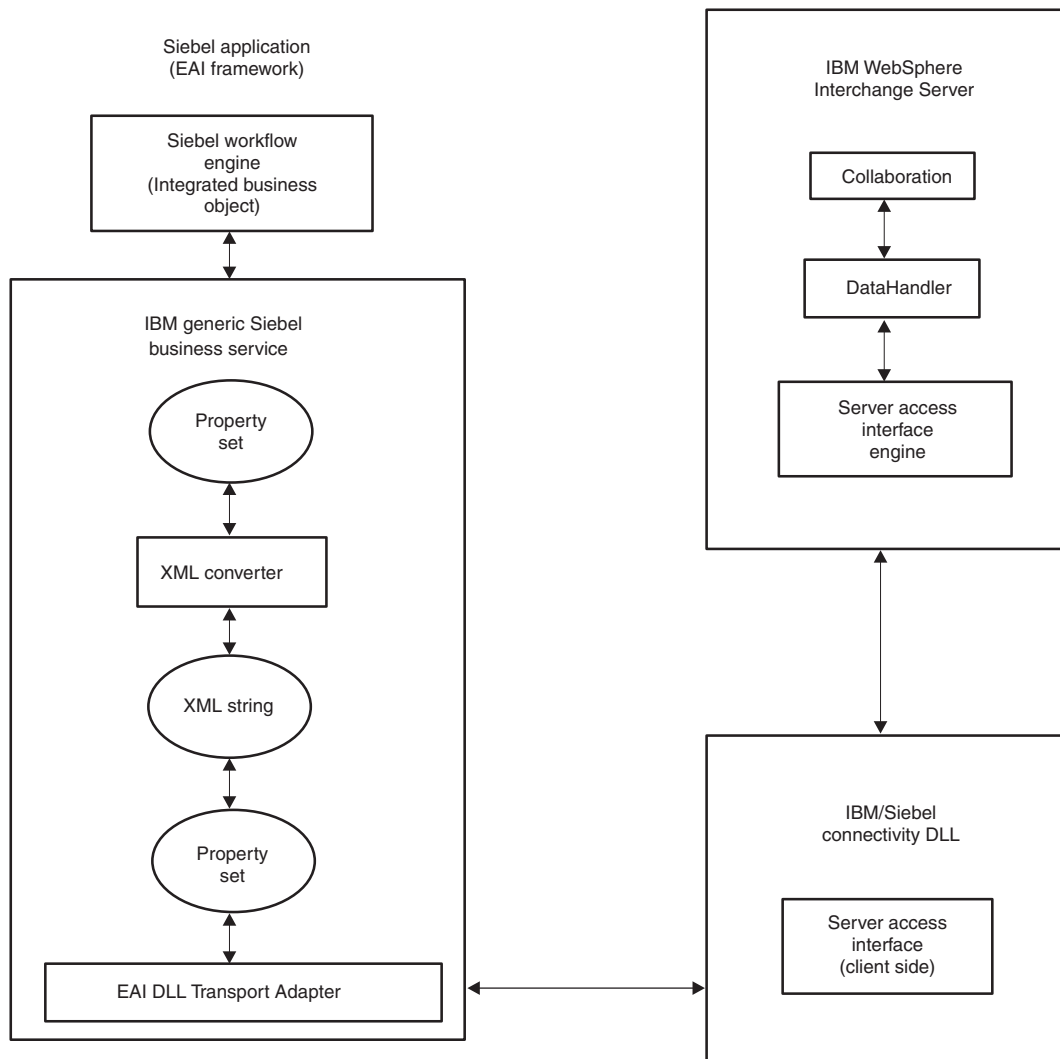


Figure 17. Siebel connector and architecture process

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 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 business objects using the XML DataHandler.

InterChange Server Express invokes the XML DataHandler through the server access interfaces. In this instance, the DataHandler converts the XML input from Siebel into a generic business object. The resultant message is passed back by InterChange Server Express 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 WebSphere Business Integration Server Express Plus Siebel connectivity DLL is installed as part of the Siebel suite of products. When you are running the WebSphere Business Integration Server Express Plus 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 WebSphere Business Integration Server Express Plus 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 12.

Table 12. IBMCrossWorlds.cfg file contents

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

Table 12. IBMCrossWorlds.cfg file contents (continued)

Item	Description
PASSWORD	Siebel application password.
[CollabDetails]	Details for each collaboration that is being used. For example, an account object would include: [CollabDetails] IntObjName=Account BOName=NewSiebel_Account CollabName=SiebelConnectivityObject PortName=From [EndCollabDetails]

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 InterChange Server Express is started, it generates an .ior file, the persistent object reference to the InterChange Server Express 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 InterChange Server Express 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: OApport=portnumber. For example, if the port number is 15000, add the following lines to the InterchangeSystem.cfg file:

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

If the access client is in the DMZ and InterChange Server Express is on a different subnet, make sure that the port number for the OApport 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 IBM business object

Generate a WebSphere Business Integration Server Express Plus 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 Express 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 Server Express Plus system delivers the default meta-object MO_DataHandler_DefaultXMLConfig. Table 13 shows attributes contained in the meta-objects that must be configured.

Table 13. Attributes contained in the meta-object

Attribute	Description
BOPrefix	The WebSphere Business Integration Server Express Plus 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 com.crossworlds.DataHandlers.text.xml.

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 InterChange Server Express. To ensure that the access client can locate the InterChange Server Express instance, access to an interoperable object reference file for that InterChange Server Express instance needs to be provided. For access details, refer to “User setup” on page 7.

Generating IBM 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 WebSphere Business Integration Server Express Plus utility, XMLborgen or XMLODA. The WebSphere Business Integration Server Express Plus 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 Adapter for Siebel eBusiness Applications component of WebSphere Business Integration Server Express Plus. 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 Server Express Plus 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 14 on page 61 shows the mapping between the `MethodName` parameter and the verb supported by the DLL.

Table 14. Mapping supported by the Siebel connectivity DLL

Method name	Verb
Insert	Create
Update	Update
Retrieve	Retrieve
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. Supporting Siebel 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 65
- “Events detection with business services” on page 66

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”

“Processing business objects that support business services” on page 64

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 EAI Siebel Adapter, a generic business service provided by Siebel; Siebel-defined Applications Services Interfaces (ASIs); and custom-written business services.

EAI Siebel Adapter and ASIs are treated similarly with respect to WebSphere Business Integration Server Express Plus business objects. They implement similar methods that are used as verbs for processing the WebSphere business objects. EAI Siebel Adapter can take any integration object that is based on a Siebel business object. The Adapter for Siebel eBusiness Applications therefore supports EAI Siebel Adapter by representing an integration object with a WebSphere business object. Similarly, the adapter supports Siebel ASIs by representing the integration objects and implementing them with a WebSphere business object.

Custom-written business objects are treated differently. Because they can implement any method, the WebSphere Business Integration Server Express Plus business object represents the service itself, not an integration object.

EAI Siebel Adapter and ASIs can be treated by the adapter as custom-written business services, and WebSphere business objects can be created to directly represent these services, although this is not recommended. See Chapter 4, “Generating business object definitions using SiebelODA,” on page 37.

Note: If an object contains the application-specific information BSN, then it will be treated as a business service. See Table 15 on page 64.

Processing 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 the adapter constructs the property set out of the WebSphere Business Integration Server Express Plus business object representing the integration object.

Example:

- instantiates a new property set for type Siebel Message
- property set type is set as SiebelMessage and properties such as IntObject Format = Siebel Hierarchy, MessageType = Integration Object, MessageId = ""
- IntObjectName is obtained from the business object application specific information
- a new property set is instantiated for the Parent Component
- type of the property set is set as ListOf<Parent component name>
- <Parent component name> can be obtained from the application specific information
- instantiate a new property set for type <Parent Component>
- set type as <parent object type>
- set different properties
- do the same for child components

Example: Siebel Integration object

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

Example: IBM business object representing Siebel integration object

```
Siebel <IntObjectName> (ParentIntegrationComponent)
    Attribute1 FN=<fieldname>
    Attribute2 FN=<fieldname>
    Attribute3 FN=<fieldname>
    +ChildIntegrationComponent
        childAttribute1 FN=<fieldname>
        childAttribute2 FN=<fieldname>
```

Object level ASI for the Parent Integration Component would be
BSN=<name>;IO=<Name>;IC=<Name>

For the Child Integration Component, it would be 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 15. 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.

Table 15. Business object level application text (continued)

Parameter	Description
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 Siebel Enterprise Applications Integration (EAI) must be present.
SiebASI=	(Deprecated) When a business object represents the ASI integration object, it contains SiebASI=true
BSTYPE=	Determines the type of business service. <ul style="list-style-type: none"> • For EAI Siebel Adapter, where the IBM business object represents the Siebel integration object, the application-specific information should contain BSTYPE=GENERIC. • For Siebel ASIs, where the IBM business object represents the Siebel integration object, the application-specific information should contain BSTYPE=ASI. • For custom-written business services, where the IBM business object represents the service, the application-specific information should contain BSTYPE=CUSTOM.

Table 16. 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 17 is VALCHANGE.

Table 17. 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.

Table 17. Verbs supported by business services (continued)

Verb	Description
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.
Synchronize	If an 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 17 on page 65.

Verbs of the WebSphere Business Integration Server Express Plus business object represent the Methods of the business service.

Verb Processing:

- 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 `getBO()` 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`.

Events detection with business services only supports EAI Siebel Adapter and Application Services Interfaces. It can handle only one integration instance.

Custom business service support

When using custom business services, the adapter checks for application specific information. If `BSTYPE=Custom` is found, it gets the Request business object under the top level business object. The simple attributes of the Request business object contained in the application specific information will provide properties, and the `SiebelMessage` container attribute provides the integration object. The adapter does the following:

1. Instantiates a new property set

2. Sets the simple attribute values as the properties of the new property set
3. Takes the Siebel Message object as the child property set

When the business service method is executed, the OutputPropertySet is obtained and populates the Response business object.

WebSphere Business Integration Server Express Plus business object corresponding to a business service:

```
Siebel_BS<Name>  
  +Request  
  +Response
```

The different methods provided on the business service serve as verbs for processing the business object.

The business object will be determined by each method/operation, and inputs/outputs will vary.

WebSphere Business Integration Server Express Plus business object corresponding to an integration object:

```
Siebel_<IntObjectName> (Parent IntegrationComponent)  
  Field1  
  Field2  
  +ChildIntegrationComponent
```

Chapter 7. Troubleshooting

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

Handling errors

The Siebel connectivity DLL component of the Adapter for Siebel eBusiness Applications returns an error in the following cases:

- Inability to connect to InterChange Server Express
- Inability to open log or trace files for writing
- Errors occurring when creating the WebSphere Business Integration Server Express Plus 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 Express 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 Express 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 Express 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 Express 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 Server Express Plus 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

Siebel 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 if it is idle for more than 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 the connector component of the adapters in WebSphere Business Integration Server Express, running on WebSphere InterChange Server Express.

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

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

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 Express

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

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
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.
- **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**
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 Express window, or see the Update Method column in the Property Summary table below.

Summary of standard properties

Table 18 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker, as standard property dependencies are based on RepositoryDirectory.

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

Table 18. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdminInQueue	Valid JMS queue name	CONNECTORNAME /ADMININQUEUE	Component restart	Delivery Transport is JMS
AdminOutQueue	Valid JMS queue name	CONNECTORNAME/ADMINOUTQUEUE	Component restart	Delivery Transport is JMS
AgentConnections	1-4	1	Component restart	Delivery Transport is IDL
AgentTraceLevel	0-5	0	Dynamic	
ApplicationName	Application name	Value specified for the connector application name	Component restart	
BrokerType	ICS	ICS		
CharacterEncoding	ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437 Note: This is a subset of supported values.	ascii7	Component restart	
ConcurrentEventTriggeredFlows	1 to 32,767	1	Component restart	Repository directory is <REMOTE>
ContainerManagedEvents	No value or JMS	No value	Component restart	Delivery Transport is JMS
ControllerStoreAndForwardMode	true or false	true>true	Dynamic	Repository directory is <REMOTE>

Table 18. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
ControllerTraceLevel	0-5	0	Dynamic	Repository directory is <REMOTE>
DeliveryQueue		CONNECTORNAME/DELIVERYQUEUE	Component restart	JMS transport only
DeliveryTransport	IDL or JMS	IDL	Component restart	
DuplicateEventElimination	true or false	false	Component restart	JMS transport only: Container Managed Events must be <NONE>
EnableOidForFlowMonitoring	true or false	false	Component restart	
FaultQueue		CONNECTORNAME/FAULTQUEUE	Component restart	JMS transport only
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory or any Java class name	CxCommon.Messaging.jms.IBMMQSeriesFactory	Component restart	JMS transport only
jms.MessageBrokerName	crossworlds.queue.manager	crossworlds.queue.manager	Component restart	JMS transport only
jms.NumConcurrentRequests	Positive integer	10	Component restart	JMS transport only
jms.Password	Any valid password		Component restart	JMS transport only
jms.UserName	Any valid name		Component restart	JMS transport only
JvmMaxHeapSize	Heap size in megabytes	128m	Component restart	Repository directory is <REMOTE>
JvmMaxNativeStackSize	Size of stack in kilobytes	128k	Component restart	Repository directory is <REMOTE>
JvmMinHeapSize	Heap size in megabytes	1m	Component restart	Repository directory is <REMOTE>
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	
MaxEventCapacity	1-2147483647	2147483647	Dynamic	Repository Directory is <REMOTE>
MessageFileName	Path or filename	InterchangeSystem.txt	Component restart	

Table 18. Summary of standard configuration properties (continued)

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

Table 18. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
SynchronousRequestTimeout	0 - any number (milliseconds)	0	Component restart	Delivery Transport is JMS
SynchronousResponseQueue	Valid JMS queue name	CONNECTORNAME/ SYNCHRONOUSRESPONSEQUEUE	Component restart	Delivery Transport is JMS
WireFormat	CwBO	CwBO	Agent restart	

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

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 that you are using, which is ICS.

CharacterEncoding

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

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

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

ConcurrentEventTriggeredFlows

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.

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

The default value is `No value`.

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

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

You must also configure a data handler with the `MimeType`, `DHClass`, and `DataHandlerConfigMOName` (optional) properties. To set those values, use the **Data Handler** tab in Connector Configurator Express. 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.

ControllerStoreAndForwardMode

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

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

DeliveryQueue

Applicable only if `DeliveryTransport` is `JMS`.

The queue that is used by the connector to send business objects to the WebSphere InterChange Server Express.

The default value is `CONNECTORNAME/DELIVERYQUEUE`.

DeliveryTransport

Specifies the transport mechanism for the delivery of events. Possible values are `IDL` for CORBA IIOP or `JMS` for Java Messaging Service. The default is `IDL`.

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

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 Express. 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 running on WebSphere InterChange Server Express.

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.

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.

EnableOidForFlowMonitoring

When you set this property to true, the adapter framework will mark the incoming **ObjectEventId** as a foreign key for the purpose of flow monitoring.

The default is false.

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).

The default value is 128m.

JvmMaxNativeStackSize

The maximum native stack size for the agent (in kilobytes).

The default value is 128k.

JvmMinHeapSize

The minimum heap size for the agent (in megabytes).

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 (see 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.

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 using Connector Configurator Express in this guide.

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

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

LogAtInterchangeEnd

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.

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

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

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

The default value is false.

OADMaxNumRetry

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

The default value is 1000.

OADRetryTimeInterval

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

The default is 10.

PollEndTime

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

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

PollFrequency

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

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

The default is 10000.

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

PollQuantity

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

PollStartTime

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

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

RequestQueue

The queue that is used by WebSphere InterChange Server Express to send business objects to the connector.

The default value is *CONNECTOR/REQUESTQUEUE*.

RepositoryDirectory

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

This value must be set to <REMOTE> because the connector obtains this information from the InterChange Server Express repository.

ResponseQueue

Applicable only if DeliveryTransport is JMS.

Designates the JMS response queue, which delivers a response message from the connector framework to the integration broker. WebSphere InterChange Server Express sends the request and waits for a response message in the JMS response queue.

RestartRetryCount

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

The default is 3.

RestartRetryInterval

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

The default is 1.

SourceQueue

Applicable only if DeliveryTransport is JMS and ContainerManagedEvents is specified.

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see “ContainerManagedEvents” on page 76.

The default value is CONNECTOR/SOURCEQUEUE.

SynchronousRequestQueue

Applicable only if DeliveryTransport is JMS.

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

The default is CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE

SynchronousResponseQueue

Applicable only if DeliveryTransport is JMS.

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

The default is CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE

SynchronousRequestTimeout

Applicable only if DeliveryTransport is JMS.

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

The default value is 0.

WireFormat

This is the message format on the transport. The setting is CwB0.

Appendix B. Connector Configurator Express

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

The topics covered in this appendix are:

- “Overview of Connector Configurator Express” on page 85
- “Starting Connector Configurator Express” on page 86
- “Creating a connector-specific property template” on page 86
- “Creating a new configuration file” on page 89
- “Setting the configuration file properties” on page 91
- “Using Connector Configurator Express in a globalized environment” on page 96

Overview of Connector Configurator Express

Connector Configurator Express allows you to configure the connector component of your adapter for use with WebSphere InterChange Server Express.

You use Connector Configurator Express to:

- Create a **connector-specific property template** for configuring your connector.
- Create a **connector configuration file**; you must create one configuration file for each connector you install.
- Set properties in a configuration file.
You may need to modify the default values that are set for properties in the connector templates. You must also designate supported business object definitions and maps for use with collaborations as well as specify messaging, logging and tracing, and data handler parameters, as required.

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 Express 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 Express.

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 Express 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 87 to set up a new one.

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

Starting Connector Configurator Express

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

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

Running Configurator Express in stand-alone mode

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

To do so:

- From **Start>Programs**, click **IBM WebSphere Business Integration Server Express>Toolset Express>Development>Connector Configurator Express**.
- Select **File>New>Configuration File**.

You may choose to run Connector Configurator Express 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 91.)

Running Configurator Express from System Manager

You can run Connector Configurator Express from System Manager.

To run Connector Configurator Express:

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 Express**. The Connector Configurator Express window opens and displays a **New Connector** dialog box.

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.
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 87.
- To use an existing file, simply modify an existing template and save it under the new name.

Creating a new template

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

To create a template:

1. Click **File>New>Connector-Specific Property Template**.
2. The **Connector-Specific Property Template** dialog box appears, with the following fields:
 - **Template**, and **Name**
Enter a unique name that identifies the connector, or type of connector, for which this template will be used. You will see this name again when you open the dialog box for creating a new configuration file from a template.
 - **Old Template**, and **Select the Existing Template to Modify**
The names of all currently available templates are displayed in the **Template Name** display.
 - To see the connector-specific property definitions in any template, select that template's name in the **Template Name** display. A list of the property definitions contained in that template will appear in the **Template Preview** display. You can use an existing template whose property definitions are similar to those required by your connector as a starting point for your template.
3. Select a template from the **Template Name** display, enter that template name in the **Find Name** field (or highlight your selection in **Template Name**), and click **Next**.

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

Specifying general characteristics

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

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

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

Specifying values

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

1. Click the **Value** tab. The display panel for Value replaces the display panel for General.

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

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

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

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

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

Setting dependencies

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

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

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

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

Creating a new configuration file

You create a connector configuration file from a connector-specific template or by modifying an existing configuration file.

Creating a configuration file from a connector-specific template

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

1. Click **File>New>Connector Configuration**.

2. The **New Connector** dialog box appears, with the following fields:

- **Name**

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

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

- **System Connectivity**

The default broker is ICS. You cannot change this value.

- **Select Connector-Specific Property Template**

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

Select the template you want to use and click **OK**.

3. A configuration screen appears for the connector that you are configuring. The title bar shows the integration broker and connector names. You can fill in all the field values to complete the definition now, or you can save the file and complete the fields later.

4. To save the file, click **File>Save>To File** or **File>Save>To Project**. To save to a project, System Manager must be running.

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

Using an existing file

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

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 InterChange Server Express repository file.
Definitions used in a previous InterChange Server Express 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 Express, revise the configuration, and then resave the file.

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

1. In Connector Configurator Express, 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)
 - InterChange Server Express Repository (*.in, *.out)
Choose this option if a repository file was used to configure the connector. 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 Express.
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 Express window displays the configuration screen, with the current attributes and values.

Connector Configurator Express requires values for properties described in the following sections:

- “Setting standard connector properties”
- “Setting application-specific configuration properties” on page 92
- “Specifying supported business object definitions” on page 93
- “Associated maps” on page 94
- “Setting trace/log file values” on page 95

Note: For connectors that use JMS messaging, an additional category may display, for special configuration of data handlers that convert the data to business objects. For further information, see “Data handlers” on page 96.

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 Express displays a configuration screen with tabs for the categories of required configuration values.

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.
- You can configure Value fields.
- The **Update Method** displayed for each property indicates whether a component or agent restart is necessary to activate changed values.

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 Express, click **File>Exit** (or close the window), and click **No** when prompted to save changes.
 - To enter values for other categories in Connector Configurator Express, select the tab for the category. The values you enter for **Standard Properties** (or any other category) are retained when you move to the next category. When you close the window, you are prompted to either save or discard the values that you entered in all the categories as a whole.
 - To save the revised values, click **File>Exit** (or close the window) and click **Yes** when prompted to save changes. Alternatively, click **Save>To File** from either the File menu or the toolbar.

Setting application-specific configuration properties

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

1. Right-click in the top left portion of the grid. A pop-up menu bar will appear. Click **Add** to add a property. To add a child property, right-click on the parent row number and click **Add child**.
2. Enter a value for the property or child property.
3. To encrypt a property, select the **Encrypt** box.
4. Choose to save or discard changes, as described for “Setting standard connector properties” on page 91.

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

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

Encryption for connector properties

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

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

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

Update method

Refer to the descriptions of update methods found in the *Standard configuration properties for connectors* appendix, under “Setting and updating property values” on page 71.

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 Express 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, including those for data handler meta-objects, and map definitions should be saved into Integration Component Library (ICL) projects. For more information on ICL projects, see the *User Guide for WebSphere Business Integration Server Express*.

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 chapter on business objects in this guide as well as the *Business Object Development Guide*.

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 Express 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 Express 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 Express window does not validate your Agent Support selections.

Maximum transaction level

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

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

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

Associated maps

Each connector supports a list of business object definitions and their associated maps that are currently active in InterChange Server Express. 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 Express 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 InterChange Server Express 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 Express window, click **Save to Project**.
4. Deploy the project to InterChange Server Express.
5. Reboot the server for the changes to take effect.

Resources

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.

Setting trace/log file values

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

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. Adapters that make use of the guaranteed event delivery enable this tab.

See the descriptions under ContainerManagedEvents in the Standard Properties appendix for values to use for these properties.

Saving your configuration file

After you have created the configuration file and set its properties, you need to deploy it to the correct location for your connector. Save the configuration in an ICL project, and use System Manager to load the file into InterChange Server Express.

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

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

For details about using projects in System Manager, and for further information about deployment, see the *User Guide for IBM WebSphere Business Integration Server Express*.

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 Express in a globalized environment

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

Connector Configurator Express supports non-English characters in:

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

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

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

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

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.

System Manager includes software developed by the Eclipse Project (<http://www.eclipse.org/>)



WebSphere Business Integration Server Express Plus V4.3



Printed in USA