IBM WebSphere Business Integration Adapters

# Adapter for Siebel 2000 User Guide

*Version 4.1.x*

IBM WebSphere Business Integration Adapters

# Adapter for Siebel 2000 User Guide

*Version 4.1.x*

**Note!**
Before using this information and the product it supports, read the information in "Notices" on page 69

**19December2003**

This edition of this document applies to IBM WebSphere Business integration Adapter for Siebel 2000, version 4.1.x, and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

# Contents

# About this document

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

This document describes the installation, configuration, and business object development for the IBM WebSphere Business Integration Adapter for Siebel 2000.

## Audience

This document is for WebSphere business integration system consultants and customers who are located in Japan and who are using Siebel 2000. 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 describes the features and components common to all WebSphere Business Integration Adapters installations, and includes reference material on specific components.

This document contains many references to two other documents: the *System Installation Guide for Windows* or *for UNIX* and the *System Implementation Guide for WebSphere InterChange Server*. If you choose to print this document, you may want to print these documents as well.

You can install related documentation from the following sites:

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

  http://www.ibm.com/websphere/integration/adapters/infocenter
- For using adapters with InterChange Server:

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

  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- For more information about message brokers (WebSphere MQ Integrator Broker, WebSphere MQ Integrator, and WebSphere Business Integration Message Broker):

  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- For more information about WebSphere Application Server:

  http://www.ibm.com/software/webservers/appserv/library.html

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

## Typographic conventions

This document uses the following conventions:

| | |
|---|---|
| `courier font` | Indicates a literal value, such as a command name, file name, information that you type, or information that the system prints on the screen. |
| **bold** | Indicates a new term the first time that it appears. |
| *italic*, *`italic`* | Indicates a variable name or a cross-reference. |
| *blue outline* | A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference. |
| *ProductDir* | Represents the directory where the product is installed. |

# New in this release

## New in release 4.1.x

Updated in December, 2003.

- Adapter installation information has been moved from this guide. See Chapter 2 for the new location of this information.
- The adapter Siebel 2000 is no longer supported on Microsoft Windows NT.

Updated in July, 2003.

The adapter can now use WebSphere Application Server as an integration broker. For further information, see "Compatibility" on page 5.

Updated in March, 2003.

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

The IBM WebSphere Business Integration Adapter for Siebel 2000 includes the connector for Siebel 2000. This adapter operates with both the InterChange Server (ICS) and WebSphere MQ Integrator integration brokers. An integration broker, which is an application that performs integration of heterogeneous sets of applications, provides services that include data routing. This adapter includes:

- A sample business object
- IBM WebSphere Adapter Framework, which consists of:
  - Development tools (including Business Object Designer and Connector Configurator)
  - APIs (including ODK, JCDK, and CDK)

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

## New in release 4.0.x

The connector has been internationalized. For more information, see "Processing locale-dependent data" on page 4 and Appendix A, "Standard configuration properties for connectors," on page 37

# Chapter 1. Overview of the connector

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

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

This document contains information about the connector framework and the application-specific component, which it refers to as the connector. The following topics are covered:



## COM support

The connector supports COM communication architecture. In the COM configuration the connector communicates with the Siebel COM data server. This configuration allows the connector to directly access and manipulate both Siebel business objects and Siebel business components.

## Supported operating systems

The connector is supported on Windows NT 4.0 Japanese only.

## Meta-data-driven design

The connector is meta-data driven. Meta-data, in the IBM WebSphere Business Integration Adapter environment, is application-specific data that is stored in business objects and that assists a 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 hard-coded in the connector.

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

## Architecture

The Siebel application architecture can be viewed in terms of three layers:

- User interface object layer
- Business object layer
- Data object layer

These layers provide a structure for user interaction with application data and insulate the application from the database. Application data is stored in a third-party database management system.

Siebel provides the Siebel object interface to enable external programs, such as a connector, to integrate with Siebel applications. The Siebel object interface is composed of methods on Siebel business objects that expose their data and functions to external programs. Siebel business objects organize application data and business logic using virtual tables, joins between tables and various relationships between tables.

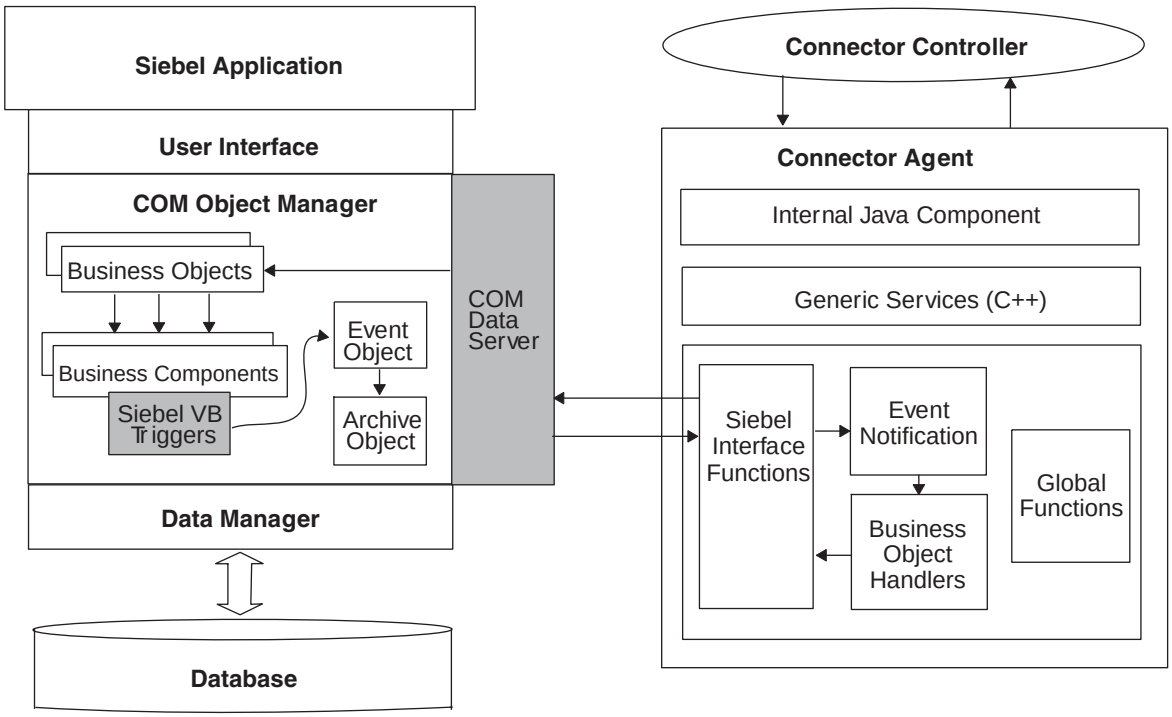
The Siebel object interface includes the Siebel COM data server which provides access to the Siebel application data. Using this interface, the connector can access and manipulate business object and business component data without interfering with user actions in the Siebel application.

Siebel business objects are customizable, object-oriented arrays of child business objects in Siebel applications. Business objects define the relationships between different business components. A Siebel business object groups one or more business components into a logical unit of information. A business component defines the structure, the behavior, and the actual data of a particular entity, such as an account.

Siebel business objects are hierarchical in structure. Each business object contains one or more Siebel business components. In the IBM WebSphere Business Integration Adapter system a top-level business object corresponds to a Siebel business object, and child business objects correspond to Siebel business components.

## Communication between the COM connector and Siebel

The illustrations below shows the Siebel COM connector components and their relationship to the Siebel application.




## Supported verbs

The connector supports Create, Retrieve, Update, and Delete verbs.

## Event notification

The connector relies on Siebel Visual Basic scripts embedded in business component event handlers to populate an event queue table. When an event occurs in Siebel, the Visual Basic script places the event in the event queue table. The connector polls the event queue table at configurable intervals and processes the events sequentially.

The connector uses the Siebel_CW_Event and Siebel_CW_Archive tables for event notification. You create these tables as part of the connector installation, and reside within the Siebel application. They are represented by the `Siebel_BO_CWEvent` and `Siebel_BO_CWArchive` business objects. To retrieve an event from Siebel, the connector calls the business object handler method `doRetrieve()` on the business object Siebel_BO_CWEvent. This business object queries the Siebel CW Events object and retrieves all events with a Status value of `0`. The order in which the connector retrieves events is specified by the priority and sort properties on the Retrieve verb in the Siebel_BC_CWEvent business object. For every event the connector retrieves from the event table, it then retrieves the actual business object that triggered that event.

When the connector has processed an event, it inserts the event into the archive table by calling the business object handler `doCreate()` method, and it removes the event from the event table by calling the `doDelete()` method. When an event is processed successfully, the status field is updated to `Success`. If an event is not processed because of an error, the status field is updated to `Failure`. If an event is unsubscribed, the status field is updated to `Skipped`.

See “Creating the event and archive tables” on page 7 for instructions on creating the queue tables.

## Processing locale-dependent data

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

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

Because this connector is written in C++, the connector framework converts application data from native encoding to the Unicode code set. The connector's application-specific component passes the data in native encoding to the connector framework, which converts the data to Unicode before sending it to ICS.

When the connector framework receives a request business object from ICS, it determines the appropriate locale information from the connector's `Locale` and `CharacterEncoding` configuration properties. It uses that information to convert the data to native encoding before passing it to the connector's C++ application-specific component.

Figure 1 illustrates where the character conversion occurs.

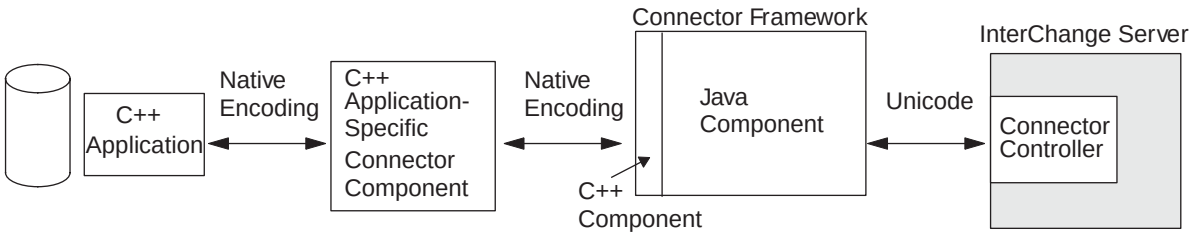



*Figure 1. How the connector handles character encoding*

For more information on the locale properties, see Appendix A, “Standard configuration properties for connectors,” on page 37

# Chapter 2. Installing and configuring the connector

This chapter describes how to install and configure the connector and how to configure the Siebel application to work with the connector. The following topics are covered:



## Compatibility

The adapter framework that an adapter uses must be compatible with the version of the integration broker (or brokers) with which the adapter is communicating. Version 2.4.x of the adapter for i2 Active Data Warehouse is supported on the following versions of the adapter framework and with the following integration brokers:

- **Adapter framework:**
  WebSphere Business Integration Adapter Framework versions 2.1, 2.2, 2.3.x, and 2.4
- **Integration brokers:**
  - WebSphere InterChange Server, versions 4.2.x
  - WebSphere MQ Integrator, version 2.1.0
  - WebSphere MQ Integrator Broker, version 2.1.0
  - WebSphere Business Integration Message Broker, version 5.0
  - WebSphere Application Server Enterprise, version 5.0.2, with WebSphere Studio Application Developer Integration Edition, version 5.0.1.

See *Release Notes* for any exceptions.

**Note:** For instructions on installing the integration broker and its prerequisites, see the following documentation.

For WebSphere InterChange Server (ICS), see the *System Installation Guide for UNIX* or *for Windows*.

For message brokers (WebSphere MQ Integrator Broker, WebSphere MQ Integrator, and WebSphere Business Integration Message Broker), see *Implementing Adapters with WebSphere Message Brokers*, and the installation documentation for the message broker. Some of this can be found at the following Web site:
http://www.ibm.com/software/integration/mqfamily/library/manualsa/

For WebSphere Application Server, see *Implementing Adapters with WebSphere Application Server* and the documentation athttp://www.ibm.com/software/webservers/appserv/library.html.

# Prerequisites

This section describes the software components you must install and the tasks you must perform before installing and running the connector.

## Prerequisite software

Siebel client software must be installed on the same machine where the connector runs. However, there is a Borland VisiBroker (runtime) incompatibility issue that does not allow running the Siebel client software on the same machine as the IBM WebSphere Business Integration Adapter software. This incompatibility can be addressed by removing the file *%ProductDir%*\bin\orb_r.dll. Note, however, that removing this file disables all the graphical tools on that machine's installation of IBM WebSphere Business Integration Adapter.

## 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.

**Note:** Before you use the connector, you must 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.

**Note:** The relevant connector message file needs to be copied to the *ProductDir*\connectors\messages directory from the *ProductDir*\connectors\messages\Siebel\COM directory.

# Installing the adapter and related files

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

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

# Required modifications

After you have installed the connector, you must copy the Siebel files to a new location, create the event and archive tables, and modify the settings in the Siebel configuration file (siebeljpn.cfg). This section contains the following topics:



## Copying Siebel files

After installing the Siebel connector, use the information in Table 1 on page 7 for copying the Siebel files to the correct location.

*Table 1. Where to copy Siebel files*

| Directory in which files are installed | Directory to which files must be copied |
|---|---|
| *ProductDir*\connectores\Siebel\COM\Siebel.dll | *ProductDir*\connectores\Siebel\Siebel.dll |
| *ProductDir*\connectores\Siebel\COM\repository\CN_SiebelCOM.txt | *ProductDir*\connectores\Siebel\repository\CN_SiebelCOM.txt |
| *ProductDir*\connectores\Siebel\COM\messages\SiebelCOMConnector.txt | *ProductDir*\connectores\Siebel\messages\SiebelCOMConnector.txt |
| *ProductDir*\connectores\Siebel\COM\repository\Siebel_BO_CWArchive.txt | *ProductDir*\connectores\Siebel\repository\Siebel_BO_CWArchive.txt |
| *ProductDir*\connectores\Siebel\COM\repository\Siebel_BO_CWEvent.txt | *ProductDir*\connectores\Siebel\repository\Siebel_BO_CWEvent.txt |

## Creating the event and archive tables

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

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

1. Ensure that all current projects have been checked in.
2. On your local database, check out and lock the following files:
   - New Table project
   - 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. See the Siebel Tools Guide, Chapter 5, for information on locking projects.
3. Apply the seven IBM WebSphere Business Integration Adapter patch files in the following order to your local database:
   - `cwtable.spf`
   - `cwview.spf`
   - `cwapplet.spf`
   - `cwbo.spf`
   - `cwbc.spf`
   - `cwdo.spf`
   - `cwscreen.spf`
4. The `cweventmanager.spf` patch adds the capability to re-queue or delete events from the CW archive table. If you require this functionality, see "CW Event Manager" on page 11 for more information before applying the patch. If you are using VB script, apply the `cweventmanagervbscript.spf` patch. If you are using e-script, apply the `cweventmanagerescript.spf` patch.
5. When you are prompted, lock the CW Audit project on your local database. See the Siebel Tools Guide, Chapter 5, for more information on locking projects.
6. Ensure that the following have been created:

- Two new tables, CX_CW_Archive_Q and CX_CW_Event_Q
- One new business object, CW Events
- Two new business components, CW Archive and CW Events
- One new view, CW Event List View
- Two new applets, CW Archive List Applet and CW Event List Applet
- One new screen CW Events and one new screen view, CW Event List view
- Two new dock objects, CX_CWArchive and CX_CWEvent

7. 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 and 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.

   See the Siebel Tools Guide, Chapter 5, for more information on page tabs.
8. Create a screen menu item as follows:
   a. Access the Application>Siebel Sales Enterprise>Screen Menu Item.
   b. Right-click and select New Record.
   c. Enter CW Events as the screen and 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.
9. Add or modify the Siebel VB scripts for the business components that correspond to IBM WebSphere Business Integration Adapter 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. See "Assigning priority to events" on page 12 for more information.
   - If you are installing is multiple connectors, set and activate the Connector Id in the VB scripts. See "Installing multiple connectors" on page 14 for more information.
   - If you want to use the Additional Object Key field, you must set it in the VB script. See "Setting multiple attributes in an event business object" on page 13 for more information.
10. 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. See the Siebel Tools Guide, Chapter 11, for instructions on applying changes to your local database.
11. Activate the new schema using the activate button. See the Siebel Tools Guide, Chapter 11, for instructions.
12. Compile the updated and locked projects on your local database to create a new Siebel repository (.srf) file. See the Siebel Tools Guide, Chapter 5, for instructions on compiling projects.

13. Open Siebel Sales Enterprise on your local database. You must have administrative privileges to perform the following:
    a. Create a new view called `CW Event List View`. Tip: Copy the view name from tools and paste it into the View Name field.
    b. Create a new responsibility called `CW Responsibility` for `CW Event List View`.
    c. Add the employees or teams who are responsible for reviewing events to the newly created `CW Responsibility`.
    d. Create the `CWCONN` user and add it to `CW Responsibility` and `Administrative Responsibility`.

    See the Siebel Applications Administration Guide, Chapter 2, for more information on responsibilities.
14. Test the application in your local environment. Ensure that you have visibility to `CW Event List View` and that an event is generated in the view after you create an IBM WebSphere Business Integration Adapter supported object. For example, create a new account in Siebel and check that a new account event appears in the `CW Event List View`.
15. 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. See the Siebel Tools Guide, Chapter 5, for more information.
16. 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.
17. Activate the queried tables in the development database. See the Siebel Tools Guide, Chapter 8, for more information.
18. Move to test and production environments accordingly. See the Siebel Tools Guide, Chapter 5, for information on migrating the Siebel repository between databases.

## Points to remember

- Distribute the new repository file to remote users via diskette or e-mail. This file replaces the existing `.srf` file in the Siebel object directory. The new repository file was created on your local database when you compiled your projects.
- When configuring the production environment, change the data source in the configuration file from local to server. This specifies that the connector should obtain data from the server instead of from each remote user. Under connector properties, determine which Siebel configuration file is in use and change the `DataSource` property in that file from `Local` to `Server`.
- Remote users must synchronize in order to update their systems with the event and archive tables.

## Changing configuration file settings

You may need to change one or more of the configuration file settings described in this section.

### Setting the DataSource property

In the `siebeljpn.cfg` configuration file on the same machine as the connector, set the value of the `DataSource` configuration property to `Server`.

### Immediate event display

To display new inbound events in Siebel as soon as they occur, set the `MaxCachedCursors` and `MaxCachedDataSets` properties in the `siebeljpn.cfg` file to `0`.

### Microsoft application conflicts

When the connector is running on the same machine with Microsoft applications, the Microsoft applications might hang. Also, if you have existing Visual Basic code for your business components in Siebel, the connector may fail to process an event. To avoid these situations you can do one of the following:

- If you are not running a Siebel client application on the same machine where the connector is running, open the `siebeljpn.cfg` file on the machine where the connector is running, and change the `EnableBasic` property to `false`.
- If you are running the Siebel client application on the same machine where the connector is running, you can create a second configuration file named `cwsiebeljpn.cfg` by copying and renaming the `siebeljpn.cfg` file in the same directory, and then changing the `EnableBasic` property in the new file to `false`. Make sure both configuration files point to the same `siebel.srf` repository file. Make sure that you change the `SiebelConfigFile` property in the System Manager so that it points to the renamed configuration file. This allows the Siebel client, connector, and Microsoft Applications to co-exist and function normally on the same machine.

# Configuring the connector

The following sections describe how to configure the connector:



## Common configuration tasks

This section contains some general points to consider to ensure a successful implementation.

### Setting the -t option

Since the connector is single threaded, you must set the `-t` option in the connector shortcut before running the connector.

### Using the Row_Id attribute

In order for the connector to insert data into the Siebel database, the Row_Id attribute must be in the CW_Siebel business object. Also, in order to perform an update or delete you must do one of the following:

- Cross-reference the Row_Id (recommended)
- Modify the collaboration to do the following:
  - Create a "dummy" send object with a unique value in one or more attributes and pass this object to the send call.

– Take the Row_Id values from the retrieve call and insert them into the original send object. Pass this object to the send call.

**Note:** For this option to work, you must hard code the attribute names into the collaboration. This adds overhead and another element to consider when modifying the system.

## Multi-value links, multi-value fields, multi-value groups, and picklists

Before attempting to access Siebel with the connector, ensure that all multi-value links, multi-value fields, multi-value groups and picklists exist in Siebel as defined in the CW_Siebel business objects.

A multi-value link exists between two Siebel business components that have a one-to-many or many-to-many relationship and reside in the same screen or applet. However, the connector assumes that a multi-value link exists between any two business components that have a one-to-many or a many-to-many relationship, regardless of whether they reside in the same screen or applet.

## Processing multiple events

If the connector must process 5000 or more multiple events, use the RefreshLogonCycle connector property. This enables the connector to disconnect and reconnect to the client after processing a specified number of events. For more information, see "Connector configuration properties" on page 15

## Changes to the Siebel application

The connector is dependent on the `Siebel.srf` file in the `objects` directory. Therefore, when changes are made in the Siebel application, the `Siebel.srf` file must be replaced with the updated version.

## CW Event Manager

`CW Event Manager` is a view in the `CWEvents` screen. It is used to re-queue events from the CW archive table to the CW event table for reprocessing by the connector. `CW Event Manager` can also be used to delete processed events from the CW archive table. Events that have been re-queued will have "Requeued Event" added to their description in the CW archive and CW event tables. This makes it easier to identify and delete duplicate events from the CW archive table.

This view provides two options. After entering the event criteria in the appropriate fields, you can either delete or re-queue the selected events. If you are using the start time and end time fields to select events, use the format: `mm/dd/yy hh:mm:ss AM/PM`. For example 3/28/01 9:45:32 PM. CW Event Manager converts this to military time.

`CW Event Manager` is script-dependent. You must apply the appropriate `CW Event Manager` patch depending on whether you are using VB or e-script. After applying the patch, perform the following:

1. In Siebel Tools, add the `CW Event Manager` business component to the `CW Events` business object.
2. Add the `CW Event Manager` screen view to the `CW Events` screen. Enter `CW Event Manager` in the Menu Text and View Bar Text fields.
3. In the Siebel Application, create the new view named `CW Event Manager`.
4. Add the view to `CW Responsibilities` and `Admin Responsibilities`.

### Performance

The Siebel dll avoids performance overhead in retrieving and comparing the application image with the sent business object. The child verb must be set manually when `SkipRetrieve` is set to `true`, otherwise the children will inherit the parent verb; because there is no retrieve, that might not be the result needed.

`SkipRetrieve` is an ASI, which can have values of `true` or `false`. This new ASI can be used only at the BC Parent level, which is immediately inside the parent business object. When this ASI is being used, the name of the business object must be set after `nm=` and must be separated from `SkipRetrieve` ASI by a semicolon.

For example, in `Siebel_BCAccount` the ASI will be `nm=Account;SkipRetrieve=true`.

## Configuring the event delivery mechanism

You can configure the following behaviors of the event delivery mechanism:

- Number of events processed per poll
- Priority assigned to events
- Number of attributes set by business object triggers

### Processing multiple events per poll

The connector can be configured to retrieve a specified number of `Siebel_BC_CWEvent` business objects and process them in a single poll. Processing multiple events per poll can improve performance when the application generates large numbers of events.

The number of events per poll is specified using application-specific information on the Retrieve verb of the `Siebel_BC_CWEvent` business object. You can change the application-specific information for this business object using the System Manager just as you would for any application-specific business object.

To specify the number of events processed on each poll, use the attribute-value pair `max=n`. For example, to specify that the connector retrieve 25 event business objects per poll, use the following application-specific information on the Retrieve verb:

```
[Verb]
Name = Retrieve
AppSpecificInfo =max=25
[End]
```

**Note:** If you do not set a maximum number of events per poll, the connector retrieves all events. This may cause system performance problems. Set this value based on the maximum number of events the connector can retrieve without impeding system performance. The default value is 25 events per poll.

### Assigning priority to events

You can assign priority values to events to specify the order in which the connector selects events to process. Event priority enables the connector poll method to handle situations where the number of events in the event store exceeds the maximum number of events the connector retrieves in a single poll.

Event priority is set in the Siebel VB script for each business component. By default, priority for all business objects is set to 1, and events are sorted by creation date.

To enable sorting by priority, first edit the VB scripts to change the priority to a value between 0 (highest) and 5 (lowest). Then, add the name-value pair `sort=Priority, EventTs(ASCENDING)` to the application-specific information of the Retrieve verb on the `Siebel_BC_CWEvent` object. This text specifies that events are sorted first by priority and then by creation date in ascending order.

Always specify a maximum number of events to retrieve per poll. If you also specify sorting by priority, the maximum events attribute-value pair precedes the priority attribute-value pair. For example, to specify sorting by priority, use the following application-specific information on the Retrieve verb of the `Siebel_BC_CWEvent` object:

```
[Verb]
Name = Retrieve
AppSpecificInfo =max=25;sort=Priority, EventTs(ASCENDING)
[End]
```

If `sort=Priority` is not specified, sorting defaults to the creation date, and the priority is not checked.

## Setting multiple attributes in an event business object

You can set one or more attributes in a subscription delivery business object by specifying them in the `Object Key` field of a Siebel VB script. If the total length of the attributes is greater than the length of the `Object Key` field (80 characters), use the `Additional Object Key` field which, supports up to 250 characters. `Object Key` is a required value. Therefore, when using the `Additional Object Key` field, you must set a dummy value for the `Object Key` field.

To set the `Additional Object Key` field:

1. In the `Siebel_BC_CWEvent` and `Siebel_BC_CWArchive` objects, set the Application-Specific Info for `Object Key` to `Additional Object Key`.
2. In `Siebel_BC_CWEvent`, create a dummy key attribute with the `AppSpecificInfo` property set to `Object Key`.
3. Add a non-key attribute with the same name as the dummy key to `Siebel_BC_CWArchive`. Make sure that the order of the `Object Key` and dummy key attributes is the same in both objects.
4. Set the `Additional Object Key` in the Siebel VB scripts for all objects.
5. Hard-code `Object Key` with a dummy value in the Siebel VB scripts for all objects. This is necessary because `Object Key` is a required value in Siebel.

In the script, you can specify an attribute and value using a name-value pair, or using a fully qualified path beginning with a forward slash (/). The syntax of a composite key is:

```
/attribute/[/attribute]=value[;/attribute[/attribute]=value]
```

You can also specify only the attribute value; in this case, the first key of the first child business object attribute is set.

For example, a VB script that sets only a single key, such as `ContactId` in the `Siebel_BCContact` business object, might include these lines:

```
RowId=GetFieldValue("Id")
Set EventBC = EventBO.GetBusComp("CW Events")
EventBC.SetFieldValue "Object Key" , RowId
```

A script that sets keys for the `ContactId` and `AccountId` attributes in the `Siebel_BCContact` business object might include these lines:

```
RowId = GetFieldValue("Id")
Account = GetFieldValue ("Account Id")
Key = "/Siebel_BCContact/ContactId=" + RowId +
";/Siebel_BCContact/AccountId=" + Account
EventBC = EventBO.GetBusComp("CW Events")
EventBC.SetFieldValue "Object Key" , Key
```

**Note:** This sample code is based on the structure of the standard `Siebel_BOContact` business object.

## Installing multiple connectors

To streamline the processing of certain types of Siebel events, you can install and configure multiple connectors on a Siebel client and configure each connector to handle specific business objects. Because multiple connectors can access the same Siebel application, each connector can process events of a given type and pass them to the integration broker. This can increase throughput and speed the transfer of data into and out of the Siebel application.

To install and set up multiple connectors, follow these steps:

1. For each additional connector that you want to run, copy and paste the existing Siebel folder in *%ProductDir%*`\connectors` to create a new folder. Rename the folder appropriately. For example, `Siebel1`, `Siebel2`, and so on.
2. Rename the connector DLL (`siebel2000.dll`) in each new folder using the same convention. For example, in the `Siebel2` folder, name the connector DLL `siebel2.dll`.
3. For each new connector, make a copy of the connector definition file, `CN_Siebel2000.txt`, in the *%ProductDir%*`\repository\Siebel2000` directory, and name it with the new connector name. For example, `CN_Siebel1.txt`.
4. Use Business Object Designer to copy the business object definition into your repository.

   **Note:** Alternatively, if InterChange Server (ICS) is the integration broker, you can use the `repos_copy` command to load the definition into the repository; if WebSphere MQ Integrator Broker is the integration broker, you can use a system command to copy the file into the repository.
5. Using copy and paste, create new copies of the connector in the System Manager. Apply the same naming convention used for the folder and the `.dll` file. For example, `Siebel1Connector`.
6. In the System Manager, set the `ApplicationName` connector configuration property for each new connector to the new name. For example, `Siebel2Connector`.
7. Set the `ConnectorId` property in each connector to the number of the connector. For example, set the `ConnectorId` property in `Siebel1Connector` to 1. The value of `ConnectorId` must be an integer. For more information see “Connector configuration properties” on page 15
8. Set the `SiebelConfigFile` property so that all connectors poll the same Siebel application for events.
9. In the System Manager, specify the supported business objects for each connector.
10. Assign collaborations to each connector. If collaborations have already been set up, you may need to recreate them.
11. Copy the connector shortcut for each of the new connectors and change the name of each shortcut to the name of the corresponding connector.

Each connector picks up events that match its ConnectorId property value. If you do not assign a value to the ConnectorId property, the connector picks up all events.

**Note:** If your site is licensed to run multiple InterChange Servers, you can provide different ConnectorId configuration properties for the connectors running on different InterChange Servers.

12. Modify the Siebel Visual Basic scripts for the business objects supported by the new connectors. Edit the ConnectorId field in each script to point to the appropriate connector ID. For example, if the Siebel2 connector polls the Account business object, set the ConnectorId field in Account to 2 by adding the following code to the Visual Basic script:

```
.ActivateField "Connector Id"
.SetFieldValue "Connector Id", "2"
```

# Connector configuration properties

You must configure the connector's standard and connector-specific connector configuration properties before you can run it. To configure connector properties, use Connector Configurator, which you access from System Manager. As you enter configuration values, they are saved in the repository.

## Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, "Standard configuration properties for connectors," on page 37 for documentation of these properties.

Table 2 provides information specific to this connector about configuration properties in the appendix.

*Table 2. Property information specific to this connector*

| Property | Note |
|---|---|
| AgentConnections | Because this connector is single-threaded, do not change the default value of this property. |
| CharacterEncoding | The connector uses this property to specify native encoding. |
| Locale | Because this connector has been internationalized, you can change the value of this property. See release notes for the connector to determine currently supported locales. |

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

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

## Connector-specific properties

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

Table 3 lists the connector-specific configuration properties for the connector. See the sections that follow for explanations of the properties.

*Table 3. Quick reference to Siebel-specific connector properties*

| Name | Possible values | Default |
|---|---|---|
| "ApplicationPassword" | | CWCONN. |
| "ApplicationUserName" | | CWCONN. |
| "ConnectorId" | | |
| "SiebelConfigFile" | | siebeljpn.cfg. |
| "SiebelDirectory" | | \sea\\client\\bin |
| "Transactional" | true or false | true |
| "UseDefaults" | true or false | false. |
| "ViewMode" on page 17 | 1 (Sales Rep)2 (Manager)3 (All)4 (Personal) | 3 (All) |
| "ConnectErrors" on page 17 | | An Oracle database error has occurred. Possibly the database name is invalid. This is the message displayed when the connection terminates in Siebel 99.5. |
| "RefreshLogonCycle" on page 17 | | |
| "RestartConnectorCycle" on page 17 | | |

## ApplicationPassword
The Password for the user account in the Siebel application.

## ApplicationUserName
Name of user account in Siebel.

## ConnectorId
If multiple connectors have been configured, identifies the connector used to process particular Siebel events. The value must be an integer.

## SiebelConfigFile
The Name of the Siebel configuration file.

## SiebelDirectory
For Siebel COM support, the directory and filename of the Siebel configuration file must be known. It is typically the client installation directory.

## Transactional
If Transactional is set to `true`, business objects with nomvl=1 are not processed on Request, Update, Create, or Delete operations. If Transactional is set to `false`, all events are processed, but there is no transactional support on objects with nomvl=1 in `CWSiebelConn`.

If the Transactional property is set to `false`, it may be necessary to set the Cascade Delete property in Siebel. Cascade Delete is an optional value in Siebel. The Cascade Delete property indicates what action to perform on detail business component records of the link if the master business component is deleted. See the *Siebel Tools Guide* for more information.

## UseDefaults
On a Create operation, if UseDefaults is set to true or 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 this property is set to false, the connector checks only for value valid values; the Create operation fails if valid values are not provided.

### ViewMode

Sets the visibility level in the system. For example, 0=SalesRep, 1=Manager. See the object interfaces reference in the Siebel documentation for a complete value set.

### ConnectErrors

If this value is set in the connector properties, the connector terminates whenever the specified error message is displayed in the Siebel log. This is used to stop the connector based on the error message. Multiple messages can be specified in this field. Separate the messages using colon ":" delimiters.

The text should contain a section of the error message after the string "Text:".

**Attention:** Use caution when creating the value as the connector will terminate when a matching string is encountered in the Siebel connector log.

The default message value is: `An Oracle database error has occurred. Possibly the database name is invalid.` This is the message displayed when the connection terminates in Siebel 99.5.

**Note:** The connector no longer supports the MaxEvents configuration property. Setting a maximum number of events to poll is supported at the business object level rather than the connector level. For information on setting the number of events to poll, see "Processing multiple events per poll" on page 12

### RefreshLogonCycle

This property indicates the number of requests the connector processes before the agent is disconnected and restarted.

### RestartConnectorCycle

If this value is set, the connector will terminate completely in the first poll call after the specified number of objects have been processed. The difference between this property and `RefreshLogonCycle` is that `RefreshLogonCycle` only disconnects the agent from the Siebel API.

## Creating multiple connector instances

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

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

### Create a new directory

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

```
ProductDir\connectors\connectorInstance
```

where `connectorInstance` uniquely identifies the connector instance.

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

```
ProductDir\repository\connectorInstance
```

### Create business object definitions

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

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

   ```
   ProductDir\repository\initialConnectorInstance
   ```

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

### Create a connector definition

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

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

### Create a start-up script

To create a startup script:

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

   dirname

2. Put this startup script in the connector directory you created in "Create a new directory" on page 17.
3. Create a startup script shortcut (Windows only).
4. Copy the initial connector's shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

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

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

## Starting the connector

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

```
ProductDir\connectors\connName
```

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

*Table 4. Startup scripts for a connector*

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

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

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

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

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

    `connector_manager_connName -start`

  where *connName* is the name of the connector and *brokerName* identifies your integration broker, as follows:
  - For WebSphere InterChange Server, specify for *brokerName* the name of the ICS instance.
  - For WebSphere message brokers (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, or WebSphere Business Integration Message Broker) or WebSphere Application Server, specify for *brokerName* a string that identifies the broker.

  **Note:** For a WebSphere message broker or WebSphere Application Server on a Windows system, you *must* include the `-c` option followed by the name of the connector configuration file. For ICS, the `-c` is optional.
- From Adapter Monitor (WebSphere Business Integration Adapters product only), which is launched when you start System Manager

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- From System Monitor (WebSphere InterChange Server product only)

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector starts when the Windows system boots (for an Auto service) or when you start the service through the Windows Services window (for a Manual service).

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

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

## Running the connector as a Windows service

Once the connector is installed, add the `-t` option to the connector shortcut in the IBM WebSphere Business Integration Adapter program group. Since the connector is single threaded, the connector shortcut must have the `-t` option.

To set the -t option on the connector shortcut:

1. Run the NT Services setup program.
2. In the Configure Connectors as NT Services window, select the connector.
3. Add `-t` as an option in the AppEndWrapper field.

The -t option is also required to run the connector as a Windows service. For more information, see the *System Installation Guide for Windows*.

## Stopping the connector

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

- If you started the connector from the command line, with its connector startup script:
  - On Windows systems, invoking the startup script creates a separate "console" window for the connector. In this window, type "Q" and press Enter to stop the connector.
  - On UNIX-based systems, connectors run in the background so they have no separate window. Instead, run the following command to stop the connector:

    `connector_manager_`*`connName`* `-stop`

    where *connName* is the name of the connector.
- From Adapter Monitor (WebSphere Business Integration Adapters product only), which is launched when you start System Manager

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- From System Monitor (WebSphere InterChange Server product only)

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector stops when the Windows system shuts down.

# Chapter 3. Developing 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

## Business object structure and relationships

The connector supports Create, Retrieve, Update, and Delete 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 two 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. An example of a Multi-Value field is the `Street Address` field of the `Account` business component. Business components can be associated in many-to-one relationships by means of Pick Lists. business component methods provide support for searching a Pick List business component for a specific value and placing that value in a field. Finally, business components can be associated in many-to-many relationships through intersection tables.

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

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

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 ID in Siebel.

**Note:** The connector for Siebel 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 for Siebel, you must ensure that there is a 1-to-1 relationship between the business object and the business component.

## Business object verb processing

The connector supports Create, Retrieve, Update, and Delete verbs. This section provides some considerations for setting verbs on Siebel business objects.

### Create

If the Create verb is set on the parent object only, all child objects will inherit the Create verb automatically. However, if the Create verb is explicitly set on one child object, it must be set on all other child objects as well.

### Update

The Update verb should be set only on the parent object.

### Delete

The Delete verb does not consider verbs on child objects.

When the Siebel application performs a delete on an object, it physically deletes the object data from its database. The connector uses a predelete trigger to capture data for the Delete event object. You must ensure that all attributes necessary for the Delete event business object are captured in the predelete operation. For more information, see "Setting multiple attributes in an event business object" on page 13.

### Retrieve

If the verb on a child object is set to Retrieve, the parent object verb must also be set to Retrieve. The Retrieve verb looks only at the first child object in the parent, or container object.

The connector supports Retrieve in business object requests, even when the collaboration does not have the key attribute that uniquely identifies the application record. This kind of Retrieve is called "retrieve by non-key values" or "retrieve by content." The connector performs Retrieve operations differently depending on the cardinality and whether key attributes are set for a child object.

- If a child object is single cardinality and one key attribute is set, the connector returns the matching record.
- If a child object is single cardinality and no key attributes are set, the connector returns the first matching record.
- If a child object is multiple cardinality, the business object container contains only one object, and no keys are set, all of the matching records are returned. The exception is when the Retrieve verb application-specific information defines a maximum number of events to retrieve that is less than the total number of matching events.
- If a child object is multiple cardinality and the business object container contains more than one object, each contained object is handled as a single cardinality

object. In other words, if one key is set, the connector returns the matching record, but if none of the keys are set, the connector returns only the first matching 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_BO_Account` specifies the Siebel Account object, as shown below.

```
[BusinessObjectDefinition]
Name = Siebel_BO_Account
Version = 1.0.0
AppSpecificInfo = Account
```

On a container attribute, the application-specific information at the object level is not used.

## 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 |
|---|---|
| `nm=` | The name of the corresponding field in the Siebel business component. |
| `pick=` | Specifies a many-to-one relationship (Pick List). The default value is false. |
| `set=` | Used only with a Pick List. Specifies the attribute name of a key in the Siebel business component referenced by the Pick List. |

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

## Field names for a simple attributes

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

```
nm=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 = nm=Main Phone Number
```

For backward compatibility, when attribute application-specific information for the Siebel field name does not include the constant nm and there are no other parameters in the application-specific information, the constant nm is the default. If there are additional parameters in the application-specific information, the complete name-value pair is required. In other words, the two application-specific information formats below are the same if the nm=parameter is the only parameter for this attribute:

```
AppSpecificInfo =Main Phone Number
AppSpecificInfo= nm=Main Phone Number
```

## Foreign key relationship using a pick list

In Siebel, a foreign key relationship between two business components is defined by a Pick List. If a field has an associated Pick List, the field's `PickList` property and `PickList` map define the relationship between the two business components. One of the attributes in the `PickList` map 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 Pick List, 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 Pick List 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 Pick List as a field property. Two attributes are required because the Pick List relationship is based on the object name rather than the object Id.

In the application-specific information for the Pick List attribute, specify that this attribute is a Pick List using the text `pick=1`. Then, to identify which record in the Pick List should be selected, use the text `set=`*fieldname*. The `set` parameter specifies which attribute of the linked object is used in the search specification to identify the record you want to pick.

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 Pick List to the `Internal Product`

business component, so you add an attribute for the key and another attribute for the Pick List. The attributes might be defined in the business object as shown below.

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

[Pick List Attribute]
Name = ProductName
Type = string
Cardinality = n
AppSpecificInfo = nm=ProductName;pick=1;set=Id
```

In some cases, Pick List 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 Pick List value. To ensure that the correct data is retrieved, you can restrict the Pick List 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=Id

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

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

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

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

Name=Account
Type=String
Cardinality=1
AppSpecificInfo=nm=Account;pick=1;set=Name;restrict=Location:Site,City:City
```

The AppSpecificInfo for restricting Pick List fields follows this syntax:

`restrict=<column name>:<Siebel GUI name>,<column name>:<Siebel GUI name>`

There is no limit to the number of restricting fields. Do not use spaces between the attributes after the `set` 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 `pick=1` specifies that the ProductName attribute corresponds to a Pick List business component, and the set parameter specifies that the value of the Id identifies which record the connector should pick.

On a Create, you must create a new Product business component for the picklist with a containment relationship to the Asset business component as follows:

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

[Pick List Attribute]
Name = Siebel_BCInternalProduct
Type = Siebel_BCInternalProduct
Relationship = containment
AppSpecificInfo = nm=Siebel_BCInternalProduct;pick=1;set=Id
```

Note the following mapping tips for Pick List attributes:

- On a Retrieve operation, correspond the value of the Pick List attribute to the name of the Pick List business component, and correspond the value of the key attribute to the key.
- On a Create or Update operation, correspond the Pick List attribute to the key, and correspond the key attribute to a null value. Because the Pick List 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 Pick List attribute contains the key value, and a pick operation is performed using the Pick List component, validation of the key is performed. If the Pick operation finds the field, it adds all the attributes in the pick correspond to the new object, and the new object is created.
- To remove the link from the Pick List, correspond the value for the Pick List attribute to null and correspond the value for the key attribute to blank.

## Application specific text 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 |
|---|---|
| `nm=` | The name of the Multi Value Field in the Siebel business component. |
| `mvg=1 pick=1 assc=1` | Specifies the relationship type between Siebel business components. mvg=1 specifies a one-to-many relationship. pick=1 specifies a many-to-one relationship, and assc=1 specifies a many-to-many relationship. The default value if no relationship type is specified is mvg=1. |
| `to=;from=` | Preprocessing instructions for the connector to set the to attribute to the value of the from attribute. The `from` attribute must be populated before the attribute containing this text parameter is processed. The to attribute is set only if it is null. This text can only be used if the objects containing the attributes have a 1-to-1 relationship. |
| `keep=1 delOnly=1` | Specifies constraints on Update operations. |

## 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:

```
nm=multiValueFieldName
```

For example, the 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_BC_BusinessAddress
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
IsKey = false
IsForeignKey = false
AppSpecificInfo = nm=Street Address
```

For backward compatibility, when attribute application-specific information for the Siebel field name does not include the constant nm and there are no other parameters in the application-specific information, the constant nm is the default. If there are additional parameters in the application-specific information, the complete name-value pair is required.

## 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. Use the following values to specify a relationship type.

| Parameter | Description |
|---|---|
| `mvg=1` | Specifies that the relationship is a one-to-many relationship, or a Siebel Multi-Value Link |
| `pick=1` | Specifies that the relationship is a Pick List. For more information on specifying a Pick List relationship between parent and child business objects, see “Specifying pick list relationships” on page 30. |
| `assc=1` | Specifies that the relationship is a many-to-many relationship through an intersection table. 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. |
| `nomvl=1` | Specifies that there is a Link or Multi Value Link relationship between the parent and child object, but there is no multi value field on the parent business object. In this case, you must set the `Transactional` property. See “Connector-specific properties” on page 15 for more information on setting the `Transactional` property. |

If no relationship parameter is specified, mvg=1 is used. 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 mvg=1 to indicate that Street Address is a Multi Value Field that links the Siebel Contact business component to the Siebel Business Address business component.

```
[Example of Container Attribute]
Name = PrimaryAddress
Type = Siebel_BC_BusinessAddress
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
IsKey = false
IsForeignKey = false
AppSpecificInfo = nm=Street Address;mvg=1
```

## 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 path;to=attribute path;
```

The attribute path value can be any of the following:

- An attribute name in the current business object
- A fully qualified path beginning with a forward slash (/). Note that the path must start with the attribute in the top-level business object, and it consists of business object attribute names rather than the Siebel or business object names. See Figure 2 for an illustration.
- A short path, which is in the format of business component name/attribute name. See Figure 3 for an illustration.

A full path is traversed from the top-level business object down the hierarchy. The from=;to=; set of parameters can be used on any simple or container attribute, and is acted on before the attribute is processed. Note the following rules:

- 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_BCAccount business object includes a child business object Siebel_BCBusinessAddress, attributes in the Siebel_BCAccount object can specify which address from the Pick List is retrieved. In this example, PrimaryAddressID is the key attribute, and Siebel_BCBusinessAddress is the picked object. The

connector gets the value of the `PrimaryAddressID` attribute, and uses that value to retrieve the specific address. The child attributes are processed after the attributes in the parent business object. The following figure shows the processing flow of attributes from parent to child business objects.

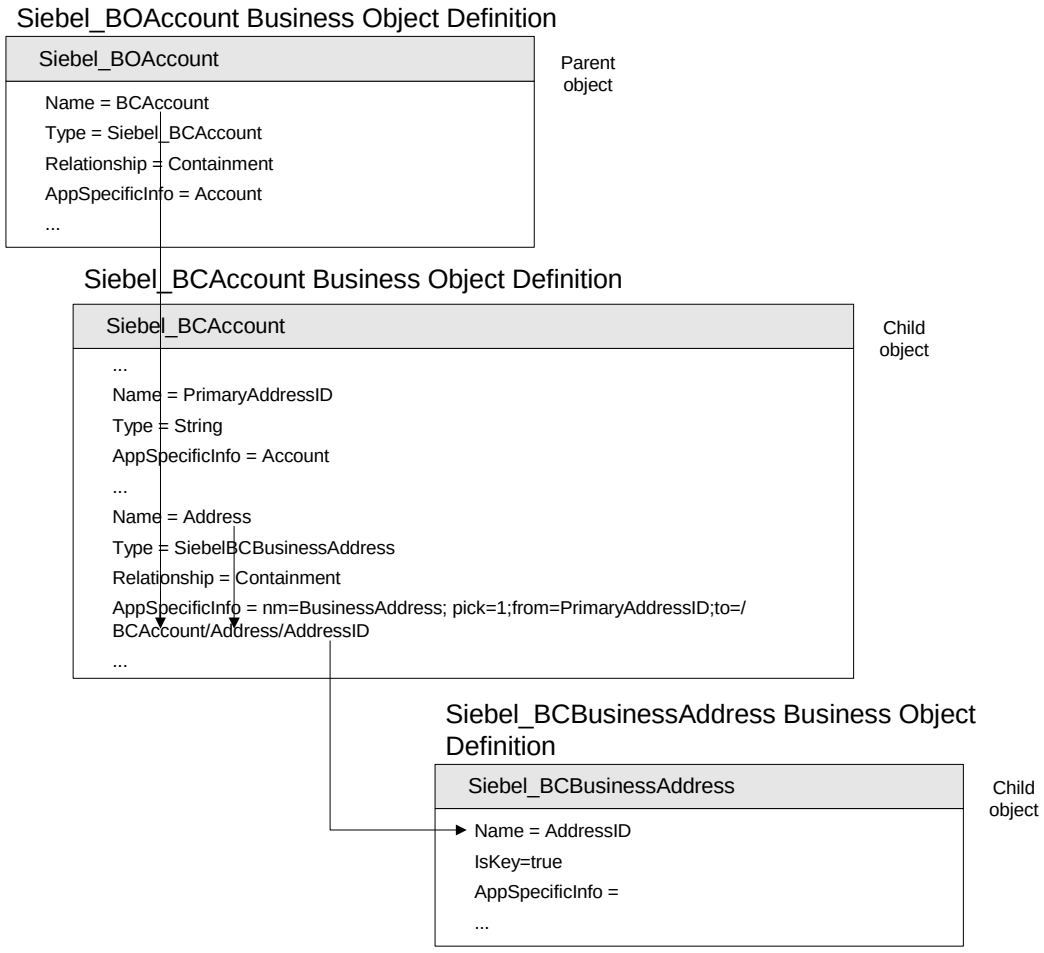



*Figure 2. Using a full path to assign value of one attribute to another*

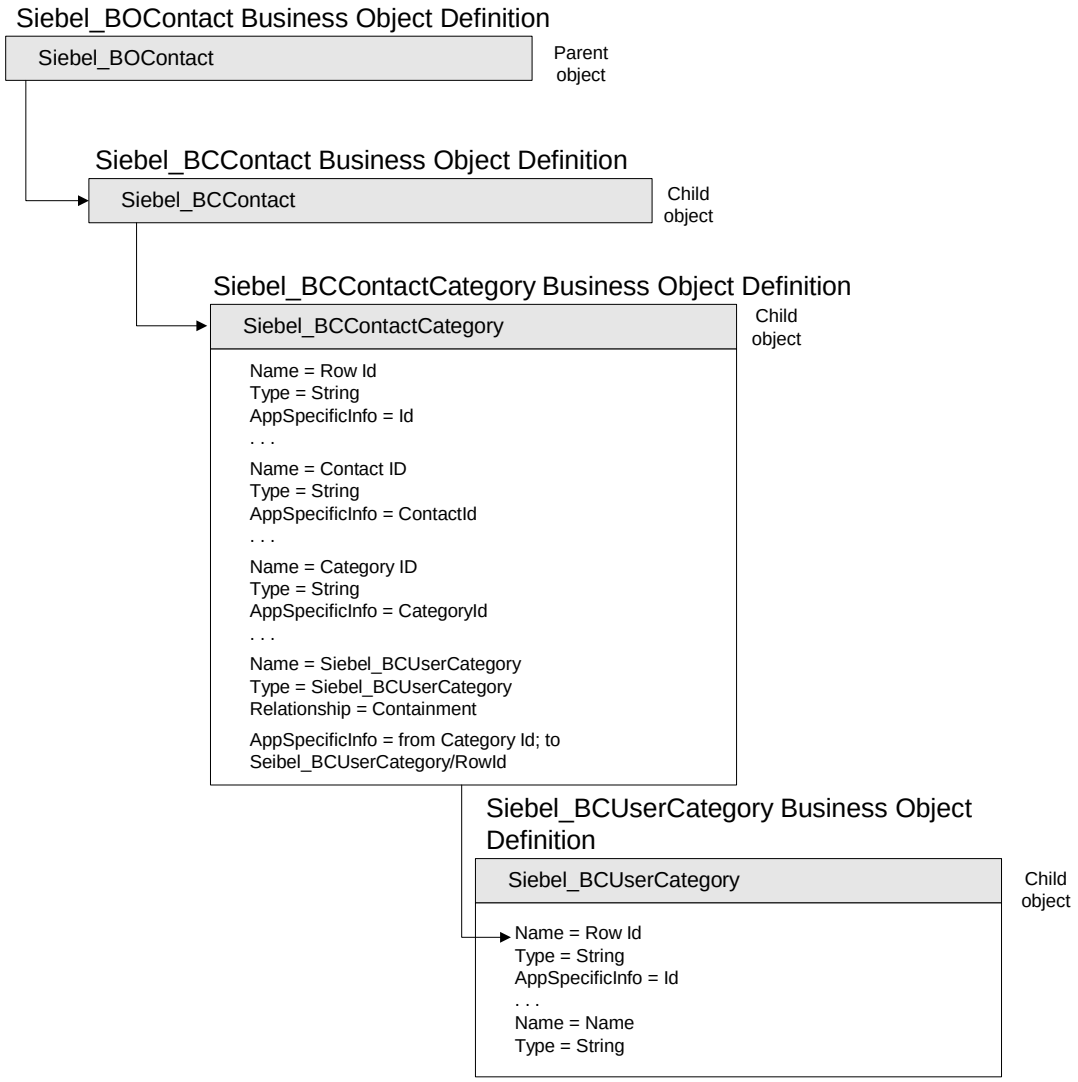
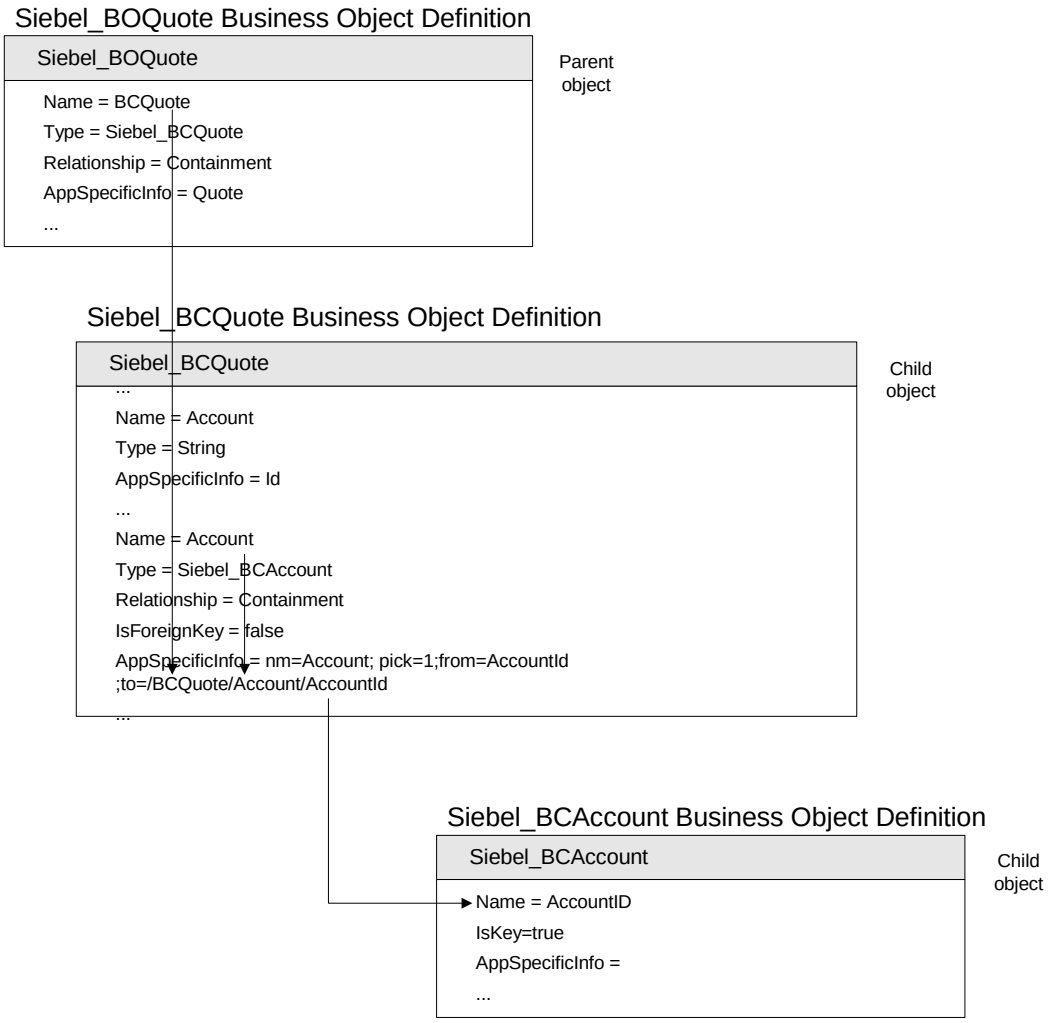



*Figure 3. Using a short path to assign value of one attribute to another*

## Specifying pick list relationships

Some Pick List relationships require the creation of the picked child object in the same transaction. In IBM WebSphere Business Integration Adapter business objects, a Pick List 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 Pick List 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 Pick List relationship between Quote and Account. As shown in the figure below, `AccountId` is the key attribute, and `Account` is the Pick List object.




In the application-specific information, pick=1 indicates that the container attribute represents a Pick List, the from= parameter is the pointer to the key attribute, and the to= parameter points to the key attribute of the Siebel_BCAccount business object.

The order of the attributes is significant in a Retrieve operation, because the value of `AccountId` 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 `AccountId` is a foreign key and is retrieved after the object is created.

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

- If none of the key attributes of the Pick List business object are set, a new object is created in Siebel and picked.
- If any of the key attributes of the Pick List business object are set, the connector searches for the object and picks it. If a Pick List 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 Pick List attributes on container business objects:

- Corresponding 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.
- Corresponding of the container attribute should be keys only if keys are known.
- If the Pick List object will be created, correspond all the required attributes as specified for the Pick List object.
- On a Delete operation, set the key attribute to a space and the Pick List container attribute to `null`.

### Specifying constraints for update operations on multiple cardinality objects

On an Update operation for a hierarchical business object, the connector first retrieves the data from the application and creates a hierarchical business object that contains the current values for the entity. It then compares the business object in the business object request to the application business object, creating records for child business objects not present in the application, updating records that exist in the application, and deleting records that exist in the application but are not present in the collaboration business object.

In addition to this default behavior, you can configure the connector's behavior for updating child objects using the keep and delOnly parameters.

- For attributes that reference child (contained) objects, attribute application-specific information can specify that the connector keep existing relationships and create new relationships for any new child business objects added during an Update operation. The default value is to delete rows representing existing relationships if they are not represented in business object requests.

  To keep existing relationships, edit the application-specific information to contain the string `keep=1;`. This directs the connector to ignore Delete verbs on child objects. For example, suppose that an existing contact is associated with one address, and the connector receives a Update business object with child business objects representing two new addresses but no child business object for the existing address. If the container attribute application-specific information has the value `keep=1;`, the contact is updated to have both the existing and the new addresses. If the application-specific information is empty, the row containing the existing address is deleted, and the new addresses are added.

  Attribute application-specific information can specify that the connector perform only Delete operations. This option is used in the case of wrapper collaborations. To delete existing relationships, edit the application-specific information to contain the string `delOnly=1;`.
- If neither keep=1 or delOnly=1 is specified, the connector processes all child business objects according to their verbs.

## 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 Oject Key attribute and creates its own RowId for that record.

**Note:** It is recommended that you use RowId as the key attribute in Create and Update requests.

# Chapter 4. Troubleshooting

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

## Failed on init() error

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

```
Mesg: Siebel Error: init() failed on Init().
```

This error message may result from the following conditions:

- There are popup windows or message boxes that appear when logging into the Siebel application. The connector cannot process the popup window or message box and fails while connecting to the application.
- There are popup windows or message boxes in the triggers or in the Siebel custom triggers. The connector cannot process the popup window or message box and fails when these triggers are executed.
- The `siebeljpn.cfg` file on the machine where the connector is running is set to a different scripting language, VBScript or Escript, from the `cfg` file for the Siebel application.

## Loss of connection to the application

When the Siebel application connection is lost, Siebel returns error 4096. This error number is associated with different text messages depending on the version of Siebel. Because Siebel does not provide documentation on error messages, the `ConnectError` property has been added to allow you to set the message for your specific version of Siebel. When the specified message is received, the connector stops. The default value for this property is the text error message that is received when Siebel 99.5 terminates. See Table 3 on page 16 for more information.

# Appendix A. Standard configuration properties for connectors

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

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

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

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

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

## New and deleted properties

These standard properties have been added in this release.

**New properties**

- XMLNameSpaceFormat

**Deleted properties**

- RestartCount

## Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties
- Connector-specific configuration properties

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

### Using Connector Configurator

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

**Note:** Connector Configurator and System Manager run only on the Windows system. If you are running the connector on a UNIX system, you must have a Windows machine with these tools installed. To set connector properties

for a connector that runs on UNIX, you must start up System Manager on the Windows machine, connect to the UNIX integration broker, and bring up Connector Configurator for the connector.

## Setting and updating property values

The default length of a property field is 255 characters.

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

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

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

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

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

# Summary of standard properties

Table 5 on page 39 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 5. Summary of standard configuration properties*

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

*Table 5. Summary of standard configuration properties (continued)*

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

*Table 5. Summary of standard configuration properties (continued)*

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

*Table 5. Summary of standard configuration properties (continued)*

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

# Standard configuration properties

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

## AdminInQueue

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

The default value is CONNECTORNAME/ADMININQUEUE.

## AdminOutQueue

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

The default value is CONNECTORNAME/ADMINOUTQUEUE.

## AgentConnections

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

## AgentTraceLevel

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

## ApplicationName

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

## BrokerType

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

## CharacterEncoding

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

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

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

## ConcurrentEventTriggeredFlows

Applicable only if `RepositoryDirectory` is <REMOTE>.

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

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

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

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

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

## ContainerManagedEvents

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

The default value is No value.

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

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

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

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

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

## ControllerStoreAndForwardMode

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

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

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

The default is true.

## ControllerTraceLevel

Applicable only if RepositoryDirectory is <REMOTE>.

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

## DeliveryQueue

Applicable only if DeliveryTransport is JMS.

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

The default value is CONNECTORNAME/DELIVERYQUEUE.

## DeliveryTransport

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

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

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

### WebSphere MQ and IDL

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

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

### JMS

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

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

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

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

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

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

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

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

  This line restricts heap memory usage to a maximum of 768 MB (3 segments * 256 MB). If the process memory grows more than this limit, page swapping can occur, which can adversely affect the performance of your system.
- The IPCCBaseAddress property to a value of 11 or 12. For more information on this property, see the *System Installation Guide for UNIX*.

## DuplicateEventElimination

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

This property can also be set to false.

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

## FaultQueue

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

The default value is CONNECTORNAME/FAULTQUEUE.

## JvmMaxHeapSize

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

The default value is 128m.

## JvmMaxNativeStackSize

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

The default value is 128k.

## JvmMinHeapSize

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

The default value is 1m.

## jms.FactoryClassName

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

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

## jms.MessageBrokerName

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

The default is `crossworlds.queue.manager`.

## jms.NumConcurrentRequests

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

The default value is 10.

## jms.Password

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

There is no default.

## jms.UserName

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

There is no default.

## ListenerConcurrency

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

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

## Locale

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

A locale name has the following format:

*ll_TT.codeset*

where:

| | |
|---|---|
| *ll* | a two-character language code (usually in lower case) |

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

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

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

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

## LogAtInterchangeEnd

Applicable only if RespositoryDirectory is <REMOTE>.

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

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

## MaxEventCapacity

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

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

## MessageFileName

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

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

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

## MonitorQueue

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

The default value is CONNECTORNAME/MONITORQUEUE

## OADAutoRestartAgent

Valid only when the RepositoryDirectory is <REMOTE>.

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

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

The default value is false.

## OADMaxNumRetry

Valid only when the RepositoryDirectory is <REMOTE>.

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

The default value is 1000.

## OADRetryTimeInterval

Valid only when the RepositoryDirectory is <REMOTE>.

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

The default is 10.

## PollEndTime

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

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

## PollFrequency

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

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

The default is 10000.

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

## PollQuantity

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

## PollStartTime

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

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

## RequestQueue

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

The default value is CONNECTOR/REQUESTQUEUE.

## RepositoryDirectory

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

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

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

## ResponseQueue

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

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

## RestartRetryCount

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

The default is 3.

## RestartRetryInterval

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

The default is 1.

## RHF2MessageDomain

WebSphere message brokers and WAS only.

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

A sample header would look like this:

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

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

## SourceQueue

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

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see "ContainerManagedEvents" on page 44.

The default value is `CONNECTOR/SOURCEQUEUE`.

## SynchronousRequestQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default is `CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE`

## SynchronousResponseQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default is CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE

## SynchronousRequestTimeout

Applicable only if DeliveryTransport is JMS.

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

The default value is 0.

## WireFormat

Message format on the transport.

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

## WsifSynchronousRequest Timeout

WAS integration broker only.

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

The default value is 0.

## XMLNameSpaceFormat

WebSphere message brokers and WAS integration broker only.

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

The default value is short.

# Appendix B. Connector Configurator

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

You use Connector Configurator to:

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

**Note:**

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

The topics covered in this appendix are:



## Overview of Connector Configurator

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

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

You use Connector Configurator to:

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

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see "Running Configurator in stand-alone mode" on page 54).

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

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

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

For **connector-specific properties**, however, you need first to define the properties and then set their values. You do this by creating a connector-specific property template for your particular adapter. There may already be a template set up in your system, in which case, you simply use that. If not, follow the steps in "Creating a new template" on page 55 to set up a new one.

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

## Starting Connector Configurator

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

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

### Running Configurator in stand-alone mode

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

To do so:

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

You may choose to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see "Completing a configuration file" on page 59.)

## Running Configurator from System Manager

You can run Connector Configurator from System Manager.

To run Connector Configurator:

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

To edit an existing configuration file:

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

## Creating a connector-specific property template

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

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

- To create a new template, see "Creating a new template" on page 55.
- To use an existing file, simply modify an existing template and save it under the new name.

### Creating a new template

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

To create a template:

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

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

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

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

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

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

## Specifying general characteristics

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

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

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

## Specifying values

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

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

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

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

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

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

### Setting dependencies

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

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

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

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

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

## Creating a new configuration file

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

To select a broker:

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

You can also do this:

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

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

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

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

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

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

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

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

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

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

# Using an existing file

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

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

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

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

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

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

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

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

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

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

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

## Completing a configuration file

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

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

1. Under the **Standard Properties** tab, select the value field for the `BrokerType` property. In the drop-down menu, select the value `ICS`, `WMQI`, or `WAS`.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 62..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

## Setting the configuration file properties

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

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

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

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

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

- Associated Maps
- Resources
- Messaging (where applicable)

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

Standard properties differ from connector-specific properties as follows:

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

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

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

## Setting standard connector properties

To change the value of a standard property:

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

## Setting application-specific configuration properties

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

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

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

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

### Encryption for connector properties

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

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

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

### Update method

Refer to the descriptions of update methods found in the *Standard configuration properties for connectors* appendix, under "Setting and updating property values" on page 38.

## Specifying supported business object definitions

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

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

### If ICS is your broker

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

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

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

To delete a business object from the supported list:

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

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

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

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

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

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

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

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

### If a WebSphere Message Broker is your broker

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

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

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

### If WAS is your broker

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

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

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

## Associated maps (ICS only)

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

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

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

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

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

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

- **Business Object Name**

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

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

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

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

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

  To explicitly bind a map:

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

## Resources (ICS)

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

## Messaging (ICS)

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

## Setting trace/log file values

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

To change the logging and tracing values:

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

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

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

## Data handlers

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

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

# Saving your configuration file

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

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

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

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

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

## Changing a configuration file

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

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

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

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

## Completing the configuration

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

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

## Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

# Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

## Programming interface information

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

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

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

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

## Trademarks and service marks

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

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

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

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

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

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


WebSphere Business Integration Adapter Framework V2.4.0

Printed in USA