IBM WebSphere Business Integration Adapters

Adapter for Clarify CRM User Guide

Version 4.8.x

IBM WebSphere Business Integration Adapters

Adapter for Clarify CRM User Guide

Version 4.8.x

Note!
Before using this information and the product it supports, read the information in Appendix C, "Notices," on page 85."Notices."

25June2004

This edition of this document applies to IBM WebSphere Business Integration Adapter for Clarify CRM (5724-G95), version 4.8.x.

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

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

Audience

This document is for WebSphere Business Integration Adapter consultants and customers. Users of this document should be familiar with the WebSphere Business Integration Adapter system, with business object and collaboration development, and with the Clarify CRM application.

Related documents

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

You can install related documentation from the following sites:

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

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

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

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, filename, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
italic, *`italic`*	Indicates a cross-reference or a variable name..
blue text	Blue text, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click any blue text to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
`[ ]`	In a syntax line, square brackets surround an optional parameter.
`...`	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.
`< >`	In a naming convention, angle brackets surround individual elements of a name to distinguish them from each other, as in `<server_name><`*`connector_name`*`>tmp.log`.
/, \	In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes. All WebSphere business integration system product pathnames are relative to the directory where the product is installed on your system.
`%text%` and *`$text`*	Text within percent (%) signs indicates the value of the Windows `text` system variable or user variable. The equivalent notation in a UNIX environment is *`$text`*, indicating the value of the *`text`* UNIX environment variable.
ProductDir	Represents the directory where the product is installed.

New in this release

Version 4.8.x

The following are new in this relaease:

- Support for Clarify 12.0 has been added. When run against Clarify 12.0, the adapter is supported on the HP-UX platform as well as the Windows and Solaris platforms.
- An ″in progess″ status has been added to the event table for event polling.
- Beginning with this release, the adapter is no longer supported on the Solaris 7 platform.

Version 4.7.x

Beginning with the 4.7 version, the adapter for Clarify CRM is no longer supported on Microsoft Windows NT.

Adapter installation information has been moved from this guide. See Chapter 2, “Installing the Clarify CRM adapter and other files” on page 9, for the new location of that information.

Prior Releases

Features and changes in prior releases.

Version 4.6.x

Support for Clarify 11.5 has been added for this release.

A new connector specific property has been added, `RestartCount`.

Beginning with the x.x version, the adapter for <<AdapterName>> does not run on Microsoft Windows NT.

The adapter can now use WebSphere Application Server as an integration broker. For further information, see “Compatibility” on page 7. The adapter now runs on the following platforms: Solaris 7 and 8

Version 4.5.x

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

The IBM WebSphere Business Integration Adapter for Clarify CRM is being released with the same functionality as in previous releases.

Version 4.4.x

The IBM WebSphere Business Integration Adapter for Clarify CRM is being released with the same functionality as in previous releases.

Version 4.3.x

The IBM WebSphere Business Integration Adapter for Clarify CRM includes the connector for Clarify CRM. This adapter supports two integration brokers: InterChange Server (ICS) and WebSphere MQIntegrator. An integration broker is an application that performs integration of heterogeneous sets of applications; it provides services such as data routing.

The IBM WebSphere Business Integration Adapter for Clarify CRM includes the following:

- An application component specific to Clarify CRM
- A sample business object (located in the \connectors\Clarify\Samples directory)
- IBM WebSphere Adapter Framework, which consists of the following:
  - Connector Framework
  - Development tools (including Business Object Designer and Connector Configurator)
  - APIs (including ODK, JCDK, and CDK)

This manual provides information about using the adapter with both the ICS and WebSphere MQIntegrator integration brokers.

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

Chapter 1. Overview of the connector

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 or technology (in this case, Clarify CRM). 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

For more information about the relationship of the integration broker to the connector, see the *IBM WebSphere InterChange Server System Administration Guide* or the *IBM WebSphere Business Integration Implementation Guide for WebSphere MQ Integrator Broker*.

This chapter describes the connector component of the IBM WebSphere Business Integration Adapter for Clarify CRM. Note that this document contains information about both the connector framework and the application-specific component. It refers to both of these as the connector. It contains the following sections:

- "The connector for Clarify CRM"
- "How the connector works" on page 2

The connector for Clarify CRM

The adapter for Clarify CRM allows the integration broker to exchange business objects with Clarify 8.0, 8.5, 9.0, 10.0, (version 10 is only available on UNIX), 10.1, 10.2, 11.1, 11.2, 11.5, and 12.0 applications. The connector supports Clarify eFrontOffice 8.0 (CeFO8) and Clarify eFrontOffice 9.0 (CeFO9) on Microsoft SQL Server or Oracle.

The connector implements business object handling, event polling, and event notification. The application-specific component of the connector generates business objects that it sends to the integration broker; it also responds to business object requests from the integration broker. It generates logging and tracing messages that it writes to a file or the connector console, or sends to the integration broker.

Along with event notification and business object request processing, the connector allows you to specify the following functionality:

- Specify the type of transactionality for a business object. The connector can wrap an entire business object request in a single transaction, and if a failure is detected during the transaction, the entire transaction is rolled back. Alternately, for hierarchical business objects, the connector can commit changes to each child object as a set of incremental transactions. This ensures that child business objects are processed in the intended order.
- Specify that the connector preprocess an attribute to obtain its value before executing a business object request.

- Specify whether a connector responds to a Retrieve request by retrieving the business object's entire hierarchy (a deep retrieve) or by retrieving only the top-level business object (a shallow retrieve).
- Specify whether a Retrieve operation succeeds for a hierarchical business object if one or more child objects are missing.
- Perform a Retrieve operation based on the keys of a record or based on non-key values. A retrieve using non-key values is also called RetrieveByContent. A RetrieveBy Content operation can use one or more designated attributes to query for a record.
- Specify how a Clarify CRM ID is created on a Create operation. A connector property can be set to specify that either Clarify CRM create the ID or that the connector pass in the ID to Clarify CRM in the business object.
- Specify on a Update operation that the connector keep existing relations between tables as well as create relations for new child business objects.

Figure 1 shows the connector components and their relationships within the WebSphere Business Integration Adapter system. In this figure, InterChange Server is used as the integration broker.

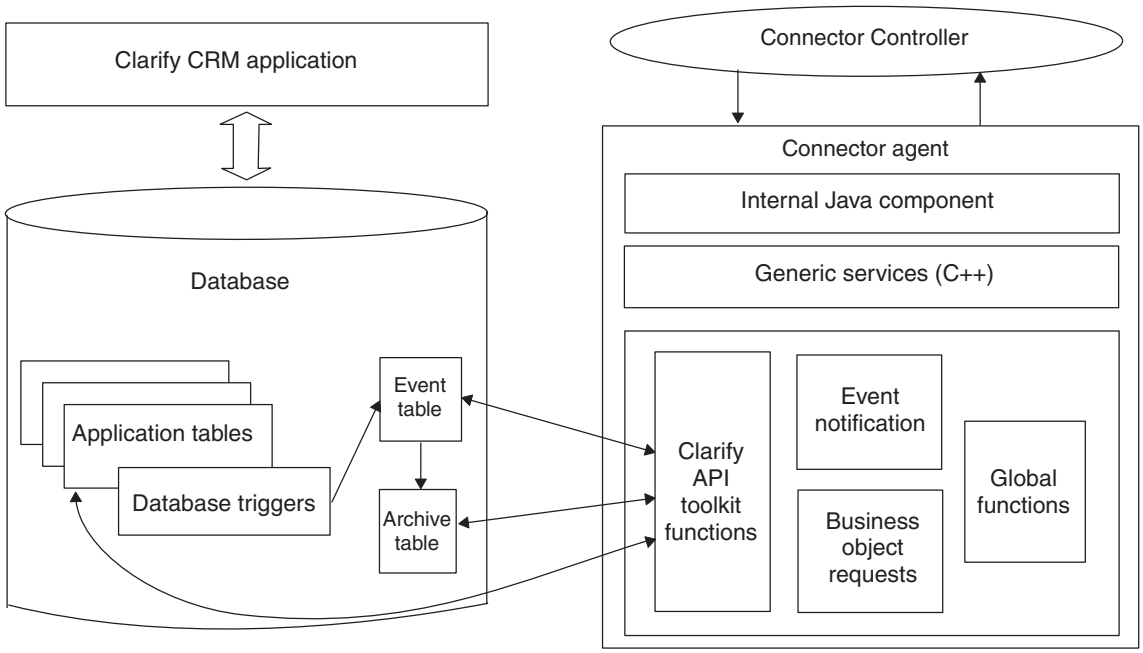

Figure 1. Architecture of the connector

How the connector works

The following sections describe how the connector processes business object requests and event notifications.

Business object processing

When the connector receives a request to perform an application operation, it communicates with the application using the API Toolkit. The connector uses the meta-data in the business object definition and the values in the business object instance to generate functions calls that access the Clarify CRM application database tables. These function calls perform the required operations in the Clarify CRM database for the business object and verb that the connector is processing.

Figure 2 illustrates business object request processing. (In this figure, InterChange Server is used as the integration broker.) When a business object is sent to the connector's business object handler, the handler generates an API call that modifies the data in the appropriate database table.

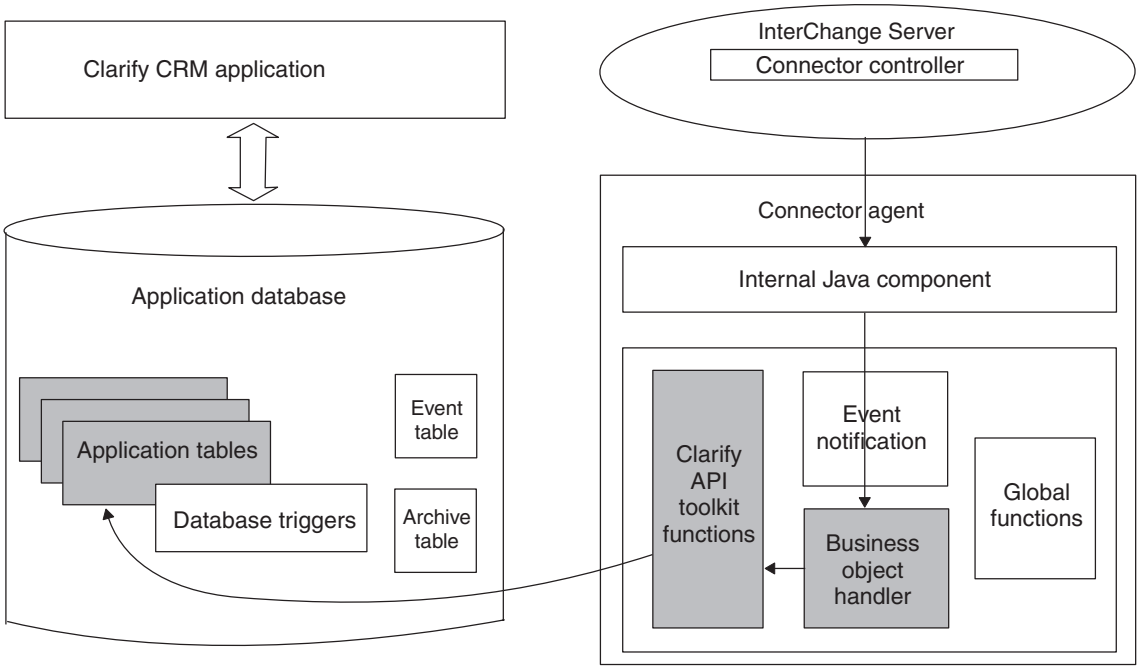

Figure 2. Business object request architecture

Delete processing

For Delete operations, the Clarify CRM application performs logical deletes rather than physical deletes. In logical deletes, rows in the database are marked as inactive rather than being physically deleted.

Because logical deletes are handled as Update operations, the connector responds to business objects with a Delete verb by converting the verb to Update. Note, however, that verb conversion in the connector may be deprecated. If ICS is used as the integration broker, use native mapping to convert Delete verbs to Update verbs for the connector.

Logical delete operations: The connector deletes a non-hierarchical business object by marking it as inactive. When the connector receives a request to delete a hierarchical business object, it logically deletes only the top-level business object but not the children.

The recommended way to logically delete a record in Clarify CRM is to change a value in a status field to an inactive value. In some tables, Clarify CRM provides a status field, and a business object can use that field to specify logical deletes. However, when a Clarify CRM table does not provide a status field but the field is needed for delete operations, you must customize the table to add a status field. If you add delete functionality to an existing business object, you will also need to edit the business object to provide a status attribute and the correct inactive value. See the Clarify CRM documentation for information on customizing Clarify CRM tables.

Physical delete operations: Neither the Clarify CRM application nor the connector performs physical deletes. If your site needs to physically delete records from a table in response to business object requests, you can add update triggers to Clarify CRM tables. As an example, the update trigger might perform a physical delete when it detects an update of a specific field to a specific value by the connector. However, keep referential integrity in mind when doing physical deletes on existing Clarify CRM tables.

Event notification

For event notification, WebSphere Business Integration Adapter provides database triggers that you add to the Clarify CRM database during the installation procedure. These triggers populate an event table whenever an event of interest occurs in Clarify CRM. The connector polls this table at a regular interval, retrieves the events, and processes the events first by priority and then sequentially. When the connector has retrieved an event, the event record is removed from the event table and stored in the archive table.

Figure 3 shows the components in the event notification architecture. (In this figure, InterChange Server is used as the integration broker.) When a relevant database table column changes, a trigger populates the event table with a record about the event. The connector polls the event table using an API call, retrieves the event, and generates a business object if the business object has subscribing collaborations. The event record is moved to the archive table after the connector retrieves the record from the event table.

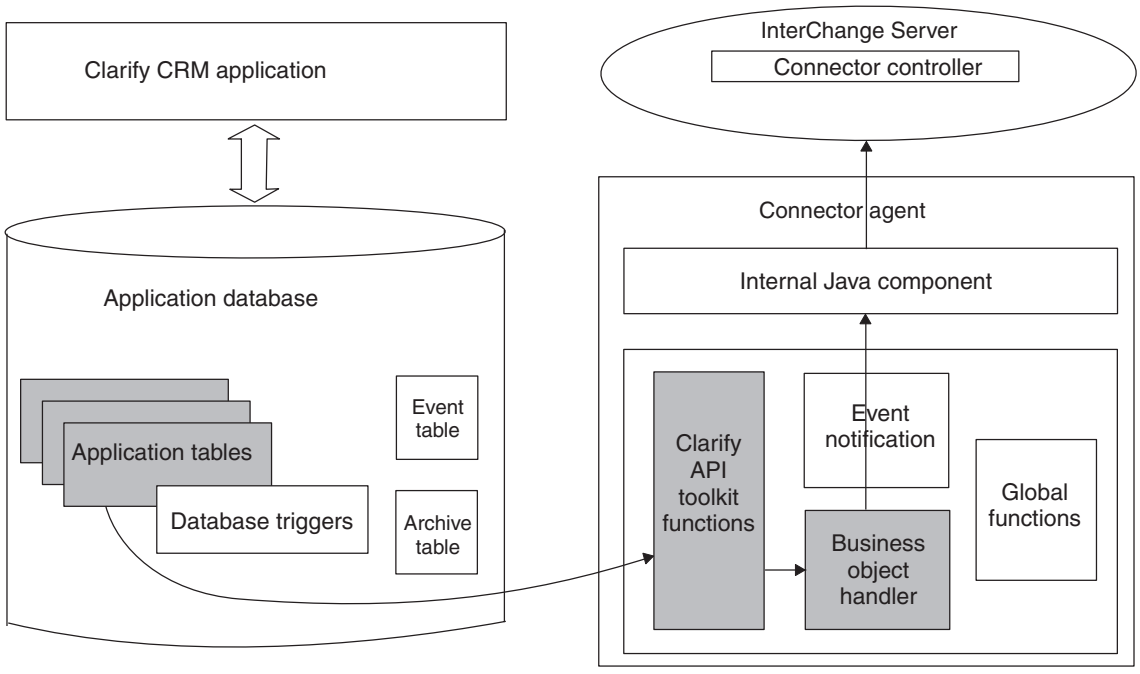

Figure 3. Event notification architecture

Archiving events

The archive table contains a copy of all events that are picked up by the connector. After the connector reads an event, the event is deleted from the event table whether the connector has successfully processed it, not processed it because of an error, or skipped it because there is no subscription for it.

When an event is deleted from the event table, the connector automatically inserts a copy of the event into the archive table with a status of Error. When the connector processes the event successfully, the status field is updated to Success. If the connector determines that there is no subscription for the event, the status field is updated to Skipped.

Delete processing in the event notification mechanism

The connector has been designed to respond to Delete and Physical Delete verbs in the event table. If your Clarify CRM application has been customized to perform physical delete operations, you can modify the Delete trigger script so that it creates business objects with separate Delete and Physical Delete verbs.

The Delete verb indicates a soft delete, which means that a record was marked for delete (inactive) but still remains in the database. If a Delete event occurs, the connector attempts to retrieve and return the entire record for the deleted entity.

The Physical Delete verb indicates a hard delete; in other words, the record was actually removed from the database. If the connector detects a Physical Delete event, it populates a business object with the values contained in the event table but does not attempt to retrieve the deleted entity's entire record.

To take advantage of this feature, you must set up Delete triggers so that they generate the appropriate verbs. Soft delete events should generate a Delete verb in the event table, while hard delete events should generate a Physical Delete verb in the event table. See "Installing database triggers for event notifications" on page 12 for information on the database triggers provided with the connector.

Chapter 2. Installing and configuring the Clarify CRM adapter

This chapter describes how to install and configure the connector and how to configure the Clarify CRM application to work with the connector. It contains the following sections:

Note: Before beginning the installation, determine whether the application has been customized. Existing application customization may affect how you enable the application for the connector. In addition, since the connector setup requires the installation of triggers on database tables, determine whether the relevant tables have existing triggers, and make sure that WebSphere Business Integration Adapter-specific triggers do not overwrite the existing triggers.

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 4.8 of the adapter for Clarify CRM 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

Additionally the adapter is supported under version 4.2.2 of the Connector Development Kit.

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

Prerequisites

Prerequisite software for Oracle

If you run the Clarify application on an Oracle database, install the Oracle client before running the connector. The `tnsname.ora`entry should be updated with the database service name.

Prerequisite software for Microsoft SQL Server

If you are running the Clarify CRM application on the MicroSoft SQL Server database, you need to install the MS SQL Server tools and utilities before running the connector.

Setting up a user account in Clarify CRM

You must create a user account in the Clarify CRM application for the connector. The account can have any valid Clarify CRM username and password. It must have the privileges to retrieve, insert, update, and delete data from the Clarify CRM database.

To set up a user account, follow these steps:

1. Bring up the Clarify CRM application and log in with system administrator privileges. A typical example of a system administrator login name is ″sa″.
2. Click on the Policies and Customers icon.
3. Click New—>Employee.
4. Fill in the login name, password, first name, and last name fields on the Employee form with the name of the connector user account. Because the database triggers for the event mechanism are coded with the user name “cw”, the name ″cw″ is recommended for all four fields.

   If you use a different user name, you will need to edit the database triggers to reflect the user name. For information, see “Installing database triggers for event notifications” on page 12.

   Note that the name for the Clarify CRM user account is the same name that you should enter for the ApplicationUserName connector configuration parameter. For information on setting the configuration properties for the connector, see “Configuring the connector” on page 13.
5. Fill in the following required fields in the Employee form:
   - Privileges - Set to system administrator.
   - Workgroup - Set to administration.
   - Site Name - Select any name from the list.
   - Email Address - Enter an email address for the user account.
6. Click Add—>Done.

Checking Windows date and time format

The Clarify CRM application uses the date format specified in the Windows short date style that appears under Regional Settings, Date in the Control Panel. If InterChange Server is used as the integration broker, the date format must correspond to the date format used in the maps for the business objects for Clarify CRM. WebSphere Business Integration Adapter native maps assume that the Windows short date style is MM/dd/yyyy. If the short date style is MM/dd/yy or any other format, you must change the map rules for the Clarify CRM maps to reflect the new date format and recompile the maps.

Installing the Clarify CRM adapter and other 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

Enabling the Clarify CRM application for the connector

Before you can use the connector to process application events and send event notification business objects to the integration broker, you must set up the event notification mechanism in the Clarify CRM application. To do this, complete the following tasks:

1. Create the event and archive tables in Clarify CRM.
2. Install database triggers on Clarify CRM tables to support the business objects needed by the collaborations running at your site.

The event table is used to queue events for pickup by the connector. The event table contains enough information so that the connector can determine the name and verb of the business object instance that it will create to represent an event.

The sections that follow provide information on these tasks.

Creating the event and archive tables

To set up event processing, you create an event table in Clarify CRM. An event table is required; you must create this table even if the connector will not poll for events. An archive table is optional; however, if the connector is polling for events and there is no archive table, all events are lost once the connector retrieves them, whether or not they were successfully processed.

Table 1 shows the Clarify CRM information for the event and archive tables.

Table 1. Connector schema information

Table type	Schema file	Default table name
Event Table	`xrdsevts.txt`	`table_xrds_events`
	`xrdsevts10.txt (Clarify version 10)`	
Archive Table	`xrdsarch.txt`	`table_xrds_archive`
	`xrdsarch10.txt (Clarify version 10)`	

Table 2 on page 10 shows the location of the schema files (`xrdsevts.txt, xrdsevts10.txt, xrdsarch.txt, and xrdsarch10.txt`) based on the operating

system.

Table 2. Schema file locations

Operating system	Schem file location
Windows	%*ProductDir*%\connectors\Clarify\dependencies\ddcomp
UNIX	*$ProductDir*/connectors/Clarify/dependencies/ddcomp

To install the event and archive tables, follow these steps:

1. Make sure that type IDs for the event and archive tables are available.
2. Execute the ddcomp command to create the tables.
3. Confirm that the event and archive tables were created.

These tasks are described in greater detail in the following sections.

Note: If your site will not archive events into an archive table, be sure to remove the value for the connector's ArchiveTableName configuration parameter using System Manager.

Type IDs for the event and archive tables

By default, the type IDs of the event and archive tables are 501 and 502. Follow these steps to determine the type IDs for your site.

1. Log into the Data Dictionary Editor and determine whether the type IDs 501 and 502 are available. If these IDs are available, you can skip to the next section. If the IDs are already in use, continue with the next step in this procedure.
2. If type IDs 501 and 502 are not available, choose new type IDs. The range of numbers reserved for customer use is from 480 to 512 and from 2000 to 4999. Select type IDs within these ranges.
3. Once you have chosen new type IDs, change the type IDs in the xrdsevts.txt and xrdsarch.txt scripts. To do this, open the xrdsevts.txt (xrdsevts10.txt for Clarify version 10)and xrdsarch.txt (xrdsarch10.txt for Clarify version 10)files in the directory appropriate for your operating system (see Table 2 on page 10).

   ```
   OBJECT xrds_events 501
   ```

   to

   ```
   OBJECT xrds_events new_id
   ```

When you have type IDs, continue with the next section.

Creating the event and archive tables

To create the event table and archive table, run the ddcomp command to execute the xrdsevts.txt (xrdsevts10.txt for Clarify version 10)and xrdsarch.txt (xrdsarch10.txt for Clarify version 10)scripts. The ddcomp utility creates the database schema and populates the Clarify CRM data dictionary with the information contained in the scripts. Typically, WebSphere Business Integration Adapter uses the ddcomp utility to create tables, such as the event table, or views as required by some Clarify CRM application-specific business objects.

Note that if you want to change the name of the event table, you can do this by changing the name in the xrdsevts.txt (xrdsevts10.txt for Clarify version 10)script before you run the ddcomp command. In this case, you will also need to set the EventTableName connector configuration parameter to the new name. To change the name of the archive table, change the name in the xrdsarch.txt

(xrdsarch10.txt for Clarify version 10)script before running the ddcomp command, and set the ArchiveTableName configuration parameter to the new name.

Running the ddcomp command: You can execute the ddcomp command from the dbadmin directory and point to the schema-file directory (see Table 2 on page 10). Alternatively, you can put the path for the directory containing the ddcomp executable in your PATH and run the ddcomp command from the schema-file directory.

Running ddcomp on SQL Server on Windows:: For Microsoft SQL Server on Windows, the syntax for the ddcomp command is:

```
ddcomp db_name db_server user_name password filename
```

where:

db_name	is the name of the Clarify CRM database.
db_server	is the name of the Clarify CRM database server.
user_name	is the Clarify CRM system administrator login name.
password	is the Clarify CRM system administrator password.
filename	is the name of the event table or archive table script.

Running ddcomp on Oracle on Windows or Unix:: For Oracle, the syntax for the ddcomp command is:

```
ddcomp oracle_database oracle_alias user_name password filename
```

where:

oracle_database	is the name of the Clarify CRM database.
oracle_alias	is a relation that ties a physical machine name to an Oracle SID. An Oracle SID is a system identifier that points to the database. It can be thought of as the name of a database instance. Clarify CRM treats the oracle_alias as a server name. Whenever the Clarify CRM application asks for a server name, the oracle_alias can usually be substituted.
user_name	is the Clarify CRM system administrator login name.
password	is the Clarify CRM system administrator password.
filename	is the name of the event table or archive table script.

On Oracle, the ddcomp and ddedit utilities also require you to build a public synonym for an object table before you can access that table using the Clarify CRM client or API. For the WebSphere Business Integration Adapter event and archive tables, execute the following SQL statements to create the public synonyms. Use SQL Plus to execute the statements.

```
create public synonym TABLE_XRDS_EVENTS for sa.TABLE_XRDS_EVENTS;
create public synonym TABLE_XRDS_ARCHIVE for sa.TABLE_XRDS_ARCHIVE;
```

Confirming the event and archive tables

To confirm that the tables were created, use the Clarify CRM Data Dictionary Editor. Log into the Data Dictionary Editor and check for the existence of objects with type ID of 501 and 502, or for objects with the type IDs that you have substituted for those IDs.

Descriptions of table schema

The event table contains the following columns:

objid	Internal identifier
object_name	Description of the object
object_verb	Verb associated with the event
object_key	Primary key for the object
event_priority	Event priority. Defined as 0-1, where 0 is the highest priority.
event_time	Time of the event
event_status	Status of the transaction associated with the event; 0 = OK, 1 = ERROR
event_description	Description of the event or error string

The archive table contains the following columns:

objid	Internal identifier
event_id	Value of objid for the event when the event was in the event table
object_name	Description of the object
object_verb	Verb associated with the event
object_key	Primary key for the object
event_priority	Event priority
event_time	Time of the event
event_status	Status of the transaction associated with the event. Status can be Success, Error, or Skipped.
event_description	Description of the event or error string

Installing database triggers for event notifications

You install database triggers in Clarify CRM to support the business objects used by the collaborations running at your site. The database triggers are implemented so that a row is generated to the event table whenever an application object is created, updated, or deleted.

You also install a database trigger that deletes events from the event table. If an archive table is available, the event record is moved to the archive table; otherwise, it is lost. Table 3 shows the name of the event table deletion trigger based on the database server for the Clarify CRM database.

Table 3. SQL script names for event table deletion triggers

Database server	Event table deletion trigger
MSSQL	MSSQL_event_delete_trigger.sql
Oracle	Oracle_event_delete_trigger.sql Oracle_event_delete_trigger10.sql (Clarify version 10)

The files in Table 3 are located in the directory:

- On a Windows system:
  %ProductDir%\connectors\Clarify\dependencies
- On a UNIX system:
  $ProductDir/connectors/Clarify/dependencies

Note: Before you can execute the trigger scripts, you must have a user account in Clarify CRM. For information on creating a user account, see "Prerequisites" on page 8.

To execute the event table deletion SQL script and install the database trigger, follow these steps:

1. Open a query window.
2. Connect to the server that hosts the Clarify CRM database.
3. Open the Clarify CRM database.
4. Execute the script for the event table delete trigger. See Table 3 on page 12 for the name of the script.
5. Execute the following commands in your SQL processor:

   ```
   grant all on table_xrds_events to username
   grant all on table_xrds_archive to username
   ```

   For SQL Server, use `isql_w` or a similar tool.

If the script executes with no errors, a message indicates that no data was returned.

You might also need to execute scripts for triggers for the business objects required by your business processes. WebSphere Business Integration Adapter provides sample business objects. You can use these business objects or create your own customized business objects. To execute triggers for your business objects, perform the following steps:

1. Identify the WebSphere Business Integration Adapter triggers for the business objects required by your business processes.
2. Connect to the database server and open the Clarify CRM database.

   Follow Steps 1 through 3 on page 13 in the previous list of steps.
3. If the connector user name is the default "cw", skip to the next step. If the user name is not "cw", update the scripts with the correct connector user name before executing them. Setting the user name correctly prevents the trigger from generating an event for an application change that resulted from a business object request.

   To update the scripts, follow these steps:

   a. Edit each script to replace "cw" with the correct user name. Change the line:

      ```
      if (@user <> "cw")
      ```

      to read:

      ```
      if (@user <> "<username>")
      ```

   b. Recompile the scripts.

   Note that the ApplicationUserName connector configuration property should be set to the name for the Clarify CRM user account.
4. Execute the scripts for the business object triggers. You must be the database owner to execute the scripts.

Configuring the connector

You must set the connector's standard and connector-specific configuration properties before you can run it. Use one of the following tools to set a connector's configuration properties:

- Connector Configurator (if ICS is the integration broker)--Access this tool from the System Manager.

- Connector Configurator (if WebSphere MQ Integrator Broker is the integration broker)--Access this tool from the IBM WebSphere Business Integration Adapter program folder. For more information about Connector Configurator, see Appendix B, “Connector Configurator,” on page 69

Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, “Standard configuration properties for connectors,” on page 51 for detailed information about these properties.

Note: This connector is single threaded. It cannot use the AgentConnections property.

Important

Because the connector for Clarify CRM supports both the ICS and WebSphere MQ Integrator Broker, configuration properties for both brokers are relevant to the connector.

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 it.

Table 4 on page 15 lists the connector-specific configuration properties for the connector. See the sections that follow for explanations of the properties.

Table 4. Connector-specific configuration properties

Name	Possible values	Default value	Required
ApplicationPassword	*Password of user account*	cw	Yes
ApplicationUserName	*Name of user account*	cw	Yes
ArchiveTableName	*Name of archive table*	xrds_archive	No
DatabaseName	*Name of Clarify CRM database*	clarify	Yes
EventTableName	Name of event table	xrds_events	Yes
FloatPrecision	Precision of a float value	6	No
ServerName	Name of Clarify CRM server	clarify	Yes
IgnoreMissingChildObject	true or false	true	No
PollQuantity	Number of events per poll	25	No
RestartCount	This is an integer indicating the number of business object requests to be processed before terminating the connector. The connector processes as many incoming business object requests and terminates with the subsequent poll call.	None	No
DoublePrecision	Precision of a double value	15	No
PollAttributeDelimiter	Delimiter for attributes in event table	:(colon)	No
RequestRetrieve	deep or shallow	shallow	No
SQLDumpFileName	Name of file containing SQL statements	C:\\temp\ XrclarifySQL.log	No
StateDumpFileName	Name of file for state report	C:\\temp\ Xrclarify.log	No
UseClarifyID	true or false	false	No
UseDefaults	true or false	false	No

ApplicationPassword
Password for the connector user account. The default value is cw.

ApplicationUserName
Name of the connector user account. The default value is cw.

ArchiveTableName
Name of archive table. The default name is xrds_archive.

DatabaseName
Name of the Clarify CRM database. The default name is clarify.

EventTableName
Name of event table. The default name is xrds_events.

FloatPrecision
Specifies the precision of a float value. The default value (six) is the default precision for floats in the Oracle and MSSQL databases. The value for the parameter should match the precision that the database uses for that data type. If the database has been customized, change the value for the connector to match.

ServerName
Name of the server running Clarify CRM. The default name is clarify.

IgnoreMissingChildObject
On a Retrieve operation, determines whether the operation succeeds for a hierarchical object if one or more child objects are missing. If the parameter is set

to `true`, the retrieve operation is successful even without all child objects. If the parameter is set to `false`, the retrieve operation fails if all child objects are not retrieved.

The default value is `true`.

PollQuantity

Number of events to process per poll. The connector poll method retrieves the specified number of event records and processes them in a single poll. Processing multiple events per poll can improve performance when the application generates large numbers of events. However, since integration broker requests are blocked while the poll method is processing events, be sure not to set the number of events too high.

As a general guide, set PollQuantity to be ten percent of the average number of events you expect to have in the event table at one time.

There is a relationship between PollQuantity and PollFrequency values. The larger the PollQuantity value, the larger the PollFrequency value should be. As a general guide, set the PollFrequency to 60 Milliseconds * the PollQuantity value.

The default value is 25.

RestartCount

This is an integer indicating the number of poll requests to be processed before terminating the connector.

The default value is ?.

DoublePrecision

Specifies the precision of a double value. The default value, 15, is the default precision for doubles in the Oracle and MSSQL databases. The value for the parameter should match the precision that the database uses for that data type. If the database has been customized, change the value for the connector to match.

The default value is 15.

PollAttributeDelimiter

Specifies the delimiter for multiple attributes in the object name column of the event table. If the Clarify CRM objid is not used as the key field and the key field may contain a colon (:), set this configuration property to a single character that will not be part of the key field.

The default value is a colon (:).

RequestRetrieve

Specifies whether a connector responds to a Retrieve request by retrieving the business object's entire hierarchy (a deep retrieve) or by retrieving only the top-level business object (a shallow retrieve). The possible values are `Deep` and `Shallow`.

Note that the connector also supports the RetrieveAll verb. If the value of `RequestRetrieve` is set to `Deep`, the business object must have support for the RetrieveAll verb.

The default is `shallow`.

SQLDumpFileName
Name of the file containing the SQL statements executed by the Clarify API. Information is appended, so the file may need to be truncated periodically.

In Windows, the default file is `C:\\temp\XrclarifySQL.log`In UNIX, the default file is `$/`*`ProductDir`*`/XrclarifySQL.log`

StateDumpFileName
Name of the file in which the Clarify API reports its state when accessing different objects. Information is appended, so the file may need to be truncated periodically.

In Windows, the default file is `C:\\temp\Xrclarify.log`. In UNIX, the default file is `$/`*`ProductDir`*`/Xrclarify.log`

UseClarifyID
On a Create operation, determines how the Clarify ID is created. If the parameter is set to `true`, Clarify CRM creates the ID. If the parameter is set to `false`, the connector passes in an ID to Clarify CRM.

The default value is `false`.

UseDefaults
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 the parameter is set to `false`, the connector checks only for valid values; the Create operation fails if valid values are not provided.

The default value is `false`.

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

If you are running Clarify 12.0 on a Solaris or HP UX platform, you must complete the following steps before you start the connector:

1. Set the Library Path (LD_LIBRARY_PATH/SHLIB_PATH) to the <Clarify Install>db_admin directory.
2. Change the ORACLE_HOME variable in <AdapterFramework>/connectors/Clarify/start_Clarify.sh to the root directory where Oracle is installed.
3. Copy clarify.env from <Clarify Install>/db_admin to the location from which you are starting the connector.

Note: If you see the following error message: `Unable to load PPOMS Library: : 'Orappoms.DLL'. (0x60004c)`, check the LD_LIBRARY_PATH to ensure that the following files are included:

- libOrappoms.sl (.so)
- libclcrt.sl (.so)
- libclutils.sl (.so)
- libkosapi.sl (.so)

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 5 shows.

Table 5. 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* [-c*configFile*]
  - 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*.

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 for the connector

This chapter describes how the connector processes business objects and describes the assumptions the connector makes. You can use this information as a guide to modifying existing business objects for Clarify CRM or as suggestions for implementing new business objects. It contains the following sections:

Note: In this chapter, the term **hierarchical** business object is used to refer to a complete business object, including all the contained child business objects at any level. The term **individual** business object is used to refer to a single business object without reference to any contained child business objects.

Meta-data-driven connector design

The connector is a meta-data-driven connector. In WebSphere Business Integration Adapter business objects, meta-data is data about the application that is stored in a business object and that assists the connector to interact with an application. A meta-data-driven connector handles each business object that it supports based on meta-data encoded in the business object definition rather than on instructions hardcoded in the connector.

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.

When processing business objects, the connector makes assumptions about the:

- Structure of business objects
- Relationship between parent and child business objects
- Format of the application-specific text
- Database representation of a business object

Therefore, when you create or modify a business object for Clarify CRM, your modifications must conform to the rules the connector is designed to follow, or the connector will not be able to process new or modified business objects correctly.

The following sections provide information on implementing business objects for Clarify CRM.

Business object structure

WebSphere Business Integration Adapter business objects can be hierarchical: parent business objects can contain child business objects, which can in turn contain child business objects, and so on. A cardinality 1 container occurs when an attribute in a parent business object contains a single child object. A cardinality n container object occurs when an attribute in the parent business object contains an array of child business objects.

The connector supports both cardinality 1 and cardinality n relationships between business objects. In each type of cardinality, the relationship between the parent and child business objects is described by the application-specific text of the key attribute of the child object.

An illustration of this relationship for a cardinality 1 container is shown in Figure 4. An illustration of a cardinality n relationship for business objects for Clarify CRM would look the same as Figure 4 except that the container attribute would contain an array of child business objects.

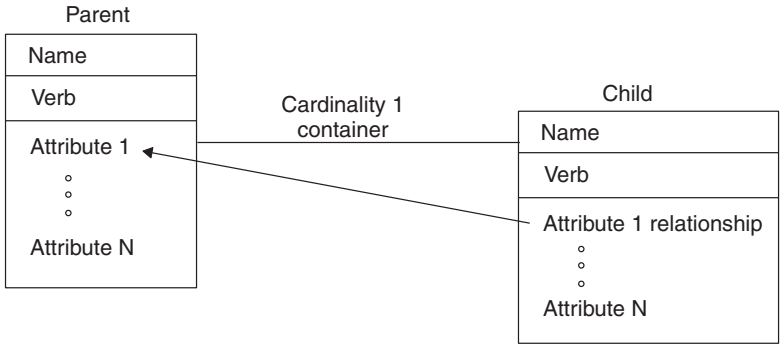

Figure 4. Business object for Clarify CRM cardinality 1 containment

Business objects and Clarify CRM database tables

In each individual business object, the first attribute is the object key. Each subsequent attribute is either a container attribute or a simple attribute that does not contain a child business object.

Note: The connector does not support specifying an attribute that represents a child business object or an array of child business objects as a key attribute.

An individual business object for Clarify CRM can represent one or more Clarify CRM database tables. For a business object that represents only one database table, the key attribute refers to the object ID of the Clarify CRM table or to another field in the Clarify CRM table that is used as the key. Each additional simple attribute within the business object represents a field in the table.

For an individual business object that represents multiple database tables, the business object key refers to the main table that the object represents. Each additional simple attribute within the business object represents a field in either the primary table or another table. The relationship between tables is contained in the attribute application-specific text in the business object definition.

Business object implementation of relationships between Clarify CRM tables

In Clarify CRM, relationships between database tables are contained in named relations. Named relations are defined by Clarify CRM and are implemented as columns in the database tables. Clarify CRM application-specific business objects use named relations to provide the connector with information about how the business object relationships correspond to Clarify CRM tables.

WebSphere Business Integration Adapter business objects for Clarify CRM can use named relations in two ways:

- An individual business object definition can include attributes that reference one or more related database tables using named relations specified in attribute application-specific text. Once an attribute has established a relationship between two tables, additional attributes can reference the new table without redefining the relationship. Figure 5 illustrates a business object with attributes that represent related tables and use named relations.
- A parent business object definition that represents one table can have one or more child business object definitions that represent other tables. The parent and each child business object can also reference other tables by means of named relations specified in attribute application-specific text.

When one or more relations are specified, the connector creates the relations between objects as part of Create or Update operations. For specific information on combining preprocessing and use of a relationship, see Chapter 4, "Troubleshooting," on page 49.

Application-specific text can use either named relations or inverse relations to join tables. An inverse relation is typically used in a child business object to point back to the parent business object.

Note: Each individual business object can contain only one join to a table. For example, if a top-level business object represents the `site` table, an error will occur if an attribute in the business object attempts links to the `site` table a second time using a named relation such as `child_site2site`. To prevent this, you can create a child business object that contains an attribute linking back to the parent business object. Or you can add a column to the `site` table, make the column a field rather than a relationship, and copy the attribute value into the field with a trigger.

Tips on designing business objects for Clarify CRM

When designing a business object definition that references multiple Clarify CRM tables, you should make a design decision about the structure and purpose of the business object. For example, if a business object definition for a table includes a few attributes for another table, defining a single business object for the two tables may be sufficient. However, if a business object definition includes many important attributes from another table, you may want to create a child business object definition for the second table rather than locating attributes for both tables in the same business object definition. In addition, if the business object already references other child business objects, creating another child business object to represent a new table makes the business object structure consistent.

If you are creating a business object that will be used primarily for Create operations, organizing attributes for different tables into child business objects may be a logical approach. However, if the business object will be used primarily for

Retrieve operations or event notification, it may be simpler to pass along information about multiple tables in one business object.

Figure 5 shows an example individual business object with attributes that reference a primary database table and attributes that reference additional tables by means of named relations.

For information on modifying a business object to add attributes that reference other tables, see "Specifying relations between objects" on page 33.

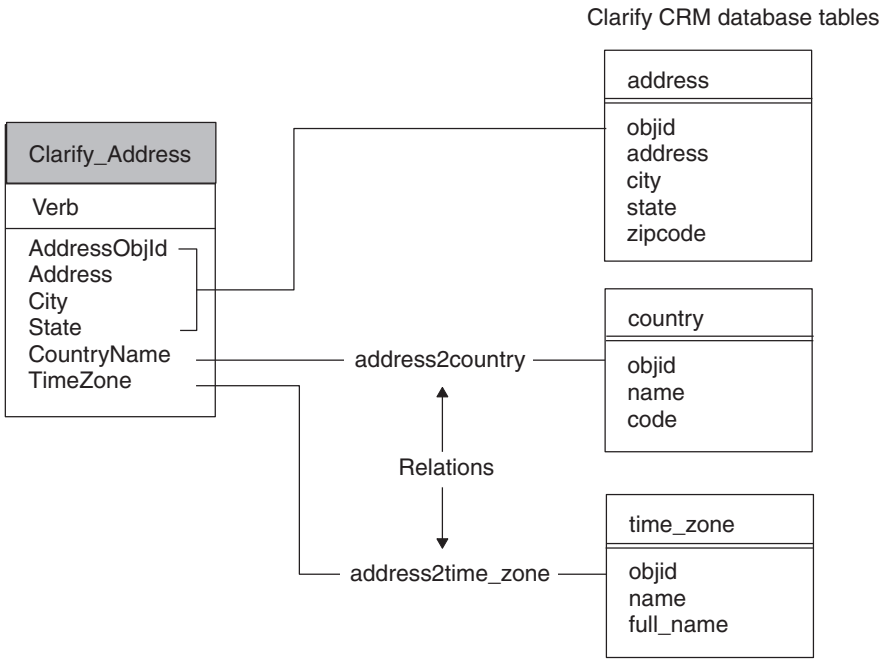

Figure 5. Business objects related to multiple Clarify CRM database tables

Business object attribute properties

Business object architecture defines various properties that apply to attributes. This section describes how the connector interprets several of these properties and describes how you should set them when modifying a business object.

Name property

Each business object attribute must have a unique name.

Type property

Each business object attribute must have a type, such as Integer, String or the type of a contained child business object.

Key property

At least one simple attribute of every business object for Clarify CRM must be specified as the key. To do so, set this property to `true`. The key attribute is always the first attribute. Therefore, when creating a business object for Clarify CRM, be sure to define the first attribute as the key so that Create operations will return a value for the new key and Retrieve operations will return the correct record from the table that the business object represents.

Note: The connector does not support specifying an attribute that represents a child business object or an array of child business objects as a key attribute.

Note that the connector ignores multiple keys and only uses the first key, except in Retrieve operations when the application-specific text in the business object definition specifies multiple key retrieve; for information on multiple key retrieve, see "Specifying multiple key attributes for retrieve operations" on page 31.

The key property is processed slightly differently in top-level business objects and child business objects.

Key property in top-level business objects

For Create operations, the connector processes key attributes as follows:

- If the key represents the value stored in the `objid` field in a Clarify CRM table, and the value for the key is null, Clarify CRM generates a value for the key, and the connector returns the value in the business object.

  If the key represents the `objid` field in a Clarify CRM table, but the value for the key is not null, the connector attempts to create a record with a new key value. Depending on the Clarify CRM schema and indexes, one of the following behaviors will occur:

  – A unique index violation will occur.

  – A record will be created successfully with a new `objid`. However, in this case you may have duplicate information in the database.

  On Create and Update operations, the Clarify CRM application is designed to generate its own `objid` values. For this reason, it is strongly recommended that you *not* design business objects to pass in values for `objid` fields unless you know how the Clarify CRM application will behave.
- On a Create operation, if the key does not represent the value in the `objid` field but represents another field in the table, such as the part number in the `part_num` table, and the value of the attribute is not null, the connector stores the attribute value in the database. Note that if the record already exists, one of the following behaviors will occur:

  – A unique index violation will occur.

  – A record will be created successfully. However, in this case you may have duplicate information in the database.
- On a Create operation, if the key is not the `objid` field, and the attribute application-specific text specifies `SchemeName=`*`Value`* for automatic generation of a value for the key, and the value of the key attribute is null, the connector generates the next sequential number in Clarify CRM and uses that value to create the record. The connector returns the generated value in the business object.

  For information on automatically generated values for attributes, see "Generating attribute values automatically" on page 43.

In all cases, on a successful completion of a Create operation, the connector will return a populated key in the business object, whether the value was the `objid` value, a value for a non-`objid` field, or an automatically generated value.

For a Retrieve operation, if the attribute is a key, its value is used to retrieve the other values for the business object, but the key value itself is not retrieved from the database. If the verb is Retrieve and the key value is `CxMissingId`, the connector will return a failure because it cannot retrieve the object.

Key property in child business objects

If the verb is Create or Update and the value of the key is unknown for a child business object, use the string `CxMissingId` for the key attribute value. When the key value of a child business object is set to `CxMissingId`, the connector will perform a Create for a child object without first checking whether the record already exists.

However, because the query parameters are not constrained when there is more than one child object, there is a possibility of orphan data in the Clarify CRM database. To eliminate this possibility, cross reference child keys.

Foreign key property

Relationships between tables in the Clarify CRM application are defined by named relations rather than by foreign keys. As an example, information on a business organization is stored in Clarify CRM in the `bus_org` table. This table includes relations to the `site` and `address` tables, which store information on the organization's site and primary address. The relations are implemented as columns in the `bus_org` table.

WebSphere Business Integration Adapter business objects for Clarify CRM use Clarify CRM named relations to represent relationships between tables. Each individual business object can reference multiple tables by including attributes whose application-specific text specify the relations. When several attributes reference the same foreign table, the relation is specified in the first attribute that references the table.

Because relationships are contained in named relations, the Foreign Key property is not used in business objects for Clarify CRM to explicitly specify that an attribute represents a foreign key. Instead, this property is used to define how the connector processes attributes with relations.

The rules for connector processing of Foreign Key attributes for Create and Update operations are shown in Table 6. Note that the Foreign Key property is only checked for attributes that have named relations in their application-specific text.

Table 6. Connector processing of foreign key property

Foreign key property setting	Attribute value	Description of connector processing
`true`	Not null	If the attribute application-specific text includes a named relation, the connector checks for the existence of the record in the application database using the value of the Foreign Key attribute. If the record exists, the connector creates the relation between the entities. If the record does not exist, the connector returns `BON_FAIL`.
`true or false`	Null	The connector evaluates whether the foreign key attribute is required (the Required property is set to true). If the attribute is required, the connector returns `BON_FAIL`. If the attribute is not required, the connector skips the attribute.
`false`	Not null	If the attribute application-specific text includes a named relation, the connector checks for the existence of the record in the application database using the value of the Foreign Key attribute. If the record exists, the connector creates the relation between the entities. If the record does not exist, the connector creates the record and creates the relation.

As shown in Table 6 on page 26, the Foreign Key property is overloaded in business objects for Clarify CRM to specify what kind of foreign key lookup the connector will perform. If Foreign Key is set to `false`, the connector performs a dynamic lookup of the related record, creating the relation if the record exists and creating both the record and relation if the record does not exist. If Foreign Key is set to `true`, the connector simply creates the relation if the record exists and fails the operation if it does not exist.

As an example, suppose that the AddressState_prov attribute in the Clarify_SiteAddress business object specifies a state in the Clarify `state_prov` table. If the `state_prov` table already contains all state names, you can set Foreign Key to `true` so that the connector will find an existing state name or fail. This limits the spellings of state names to those in the table. If the spelling of state names is not important, you can set Foreign Key to `false` so that the connector will insert any spelling of state names.

In summary, when modifying business objects for Clarify CRM, use the Foreign Key property to specify what kind of lookup the connector should perform.

Required property

The Required property specifies whether an attribute must contain a value. If a particular attribute in a business object that you are modifying must contain a value, set the Required property to true.

If the verb is Create and Required is true, the connector will cause the Create operation to fail if the business object does not have a valid value or a default value for a Required attribute.

The connector does not check the Required property for verbs other than Create.

Max length property

The connector does not currently use the Max Length property. It is good practice, however, to set the value of this property to the number of bytes the attribute can contain. At a minimum, set the value of Max Length to 8 so that the attribute value can be `CxIgnore`.

UseDefaults property

This property specifies a default value for the attribute that will be used if there is no runtime value for the attribute in the business object. If the following criteria are met, the UseDefaults configuration property is enabled:

- UseDefaults = true
- Business object verb = Create
- Business object attribute = Required
- Default value is populated for the attribute that is marked Required

If the UseDefaults property is enabled the following will occur:

- If the business object verb is Create and the value of an attribute is null, the connector will populate the field with the value in the attribute Default Value property.
- If the attribute is Required and there is no default value in the business object definition and the value of the attribute is null, the connector will fail.

Attribute and database types

The connector supports the data types shown in Table 7. When you add a business object attribute, set the attribute data type according to the data type of the field in the Clarify CRM table.

Table 7. Business object attribute and database types

Clarify CRM data types	WebSphere Business Integration Adapter attribute data types
`int, tinyint, smallint`	`integer`
`real`	`float`
`varchar, text`	`string`
`decimal`	`string`
`datetime`	`string`

Important notes on attributes that reference Clarify fields of type text

The connector supports Clarify CRM text fields of types `varchar` and `text`. WebSphere Business Integration Adapter business objects for Clarify CRM can include attributes with a data type of `string` that reference fields in Clarify CRM with the data type of `text`.

Note, however, that an attribute that references a field of type `text` cannot have a relation in its business object application-specific text. To work around this, you can add a preceding attribute that represents the object ID of the table, put the relation between the tables in the application-specific text of that attribute, and then reference the table in the application-specific text of the attribute representing the `text` field.

For example, suppose that you have a business object named Clarify_Case that represents the Clarify CRM `case` table. You want to add a relation to the `notes_log` table so that you can include an attribute for the text of a note related to a case.

To do this, you add an attribute in the Clarify_Case business object for the object ID of the `notes_log` table, and, in the application-specific text for this attribute, include the relation between the tables. Then add an attribute for the text of the note. In this example, the attribute you want to reference is `notes_log:description`, which is a field of type `text`. Figure 6 illustrates these attributes in the example Clarify_Case business object.

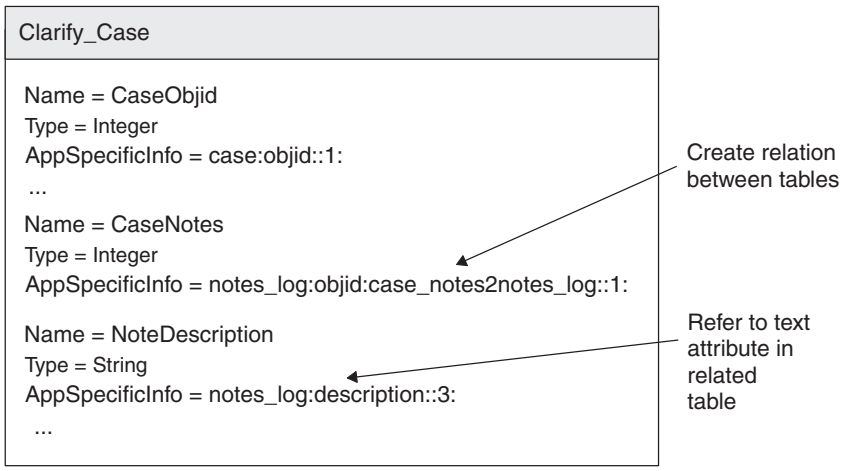

Figure 6. Referencing attributes of type text

If the connector attempts to select or update a text field in Clarify CRM, and the business object attribute with the `text` data type does not refer to the attribute representing the table object ID, the following errors may result:

- Execution of the SQL command failed due to unexpected problems.
- `TEXT` and `IMAGE` data types may not be used in the `WHERE` or `HAVING` clause, except with the `LIKE` predicate and the `IS NULL` predicate.

Note that although `ObjId` is necessary for `text` fields, it is not necessary for `varchar` fields.

Business object application-specific text

Application-specific text in business object definitions provides the connector with application-dependent instructions on how to process business objects. This meta-data is used in conjunction with a business object's attribute properties and structure. If you extend or modify Clarify CRM application-specific business objects, you must make sure that the application-specific information in the business object definition matches the syntax that the connector expects.

This section provides information on the object, attribute, and verb application-specific text format for business objects for Clarify CRM. Table 8 provides an overview of the functionality available in business object application-specific text for Clarify CRM.

Table 8. Overview of application-specific text in business objects for Clarify CRM

Scope of application-specific text	Types of functionality
Entire business object	Define the scope of transactions in the connector Specify that the connector perform retrieves using multiple keys
Simple attributes	[Required] Specify table name, field name, and data type for an attribute Specify one or more relations between attributes in the primary table and one or more foreign tables Specify how retrieve queries are constructed Specify automatic generation of attribute values Specify preprocessing for attributes Specify that Retrieve operations use non-key values
Container attributes	Specify whether existing relations are kept for child business objects on Update operations

Table 8. Overview of application-specific text in business objects for Clarify CRM (continued)

Scope of application-specific text	Types of functionality
Business object verb	Specify that the connector retrieve only the parent business object in a hierarchical object. Specify that the connector follow a retrieve by non-key values with a regular retrieve operation.

Using business object application-specific text

The business object for Clarify CRM includes application-specific text at the business object level that:

- Defines the scope of business object transactions
- Specifies multiple key retrieve operations

The sections that follow provide information on these types of application-specific text.

Defining the scope of business object transactions

When the connector is processing a request to change data in the Clarify CRM database, it can process business object transactions in two ways:

- It can make all changes at once and commit the entire transaction.
- For a hierarchical business object, it can commit changes to each child object as a set of incremental transactions.

By default, the connector commits an entire business object at once. You can use the object application-specific text to specify that the connector execute a commit after each business object.

Transactionality is set by specifying Transactionality=Partial or Transactionality=Full in the application-specific text. If Transactionality is set to Partial, the connector executes a commit after processing each business object. This means that if there is a parent object and several contained child objects, the connector will execute a commit after updating the parent object, a commit after updating the first child object, and so on. If Transactionality is set to Full or not set, the connector will wrap the whole transaction in one commit.

For example, to specify that the connector execute a commit after each business object, use the following text:

```
AppSpecificInfo = Transactionality=Partial
```

For every business object, it is recommended that transactionality initially be set to Full. However, due to a known problem in the Clarify API, the connector cannot rely on the sequence of the SQL statements that the API generates. In certain cases, the API may perform an action in a different order than the connector requested.

To force the Clarify API to execute the statements in the order specified, set Transactionality to Partial. Note, however, that if transactional support is set to Partial, a hierarchical business object is not committed in one transaction, and a failure only partially rolls back transactions for a hierarchical business object.

To view the set of actions that the API has performed, examine the SQL log that the connector has generated. This file contains the actual SQL scripts executed by

the connector. The name of this log is set by the connector configuration property SQLDumpFileName; the default file is `C:\\temp\XrclarifySQL.log`.

Note: If the entire WebSphere Business Integration Adapter business object is committed in one transaction, the transaction log of the Clarify CRM database may need to be significantly increased. For example, if a Clarify_Contract business object includes 900 line items, the log must be large enough to handle this object in one transaction. Consult your database administrator for information.

Specifying multiple key attributes for retrieve operations

By default, when the connector performs a Retrieve operation, it uses the value of the first key attribute, which must be the first attribute, as the key to the record in Clarify CRM. You can also use the business object application-specific text to specify that the connector retrieve a business object using the value of several attributes designated as keys.

The use of more than one key attribute creates a compound key for the purpose of querying Clarify CRM. Because multikey retrieve is set at the parent business object level, this application-specific text enables multikey retrieve for all business objects of a particular type.

To allow multikey retrieve, edit the business object application-specific information to specify MultiKeyRetrieve=True. The format is:

```
AppSpecificInfo = MultiKeyRetrieve=True
```

The default is false; that is, only the first key attribute value is used. Note the following restrictions:

- All attributes used in the retrieve operation must have non-null values. Any attribute whose value is null is ignored.
- The first attribute in the business object determines the Clarify CRM table from which all other attributes are retrieved. Only key values that reference that table are used.

Attribute application-specific text

The application-specific text for attributes differs depending on whether the attribute is a simple attribute or a container attribute. For information on application-specific text for container attributes, see "Application-specific text for container attributes" on page 45.

Format of application-specific text for simple attributes

For simple attributes, application-specific text format consists of eight positional parameters separated by colon (:) delimiters. The positional parameters can be preceded by one or more name-value pairs. The name-value pairs are optional and can be specified in any order. Semi-colons are used as field delimiters between name-value pairs. If any name-value pairs are present, they are separated from the positional parameters by a vertical bar (|). Therefore, reserve the colon, semi-colon, and vertical bar characters as delimiters.

The format of attribute application-specific text is shown below. Note that only the table name, field name, and data type parameters are required; bracketed fields in the syntax example are optional.

```
[SchemeName = Value;PP=Value;RBC=Value| ]
tablename:fieldname:[relation[,...]]:datatype:[ParentIndex]:
[RelateToAttribute[,...]]:[AttrNameToLookFor:FieldNameToRetrieve]
```

Table 9 describes the meaning of each positional parameter. Table 10 briefly describes the name-value pairs. Examples of the use of these parameters are provided in the sections that follow.

Table 9. Positional parameters in attribute application-specific text

Field	Description
tablename	The name of the Clarify CRM database table for this attribute.
fieldname	The name of the column in the Clarify CRM database table.
relation[, ...]	A list of Clarify CRM relations used to relate this attribute to the attributes in the RelateToAttribute list described below.
datatype	An integer value indicating the type of data. Supported types are: 1 = integer data (int, tinyint), 2 = floating point data (real), 3 = string data (varchar, text, datetime), 4 = decimal data (corresponding to the Clarify CRM decimal type introduced in Clarify 5.0)
ParentIndex	The index of the parent object in a query.
RelateToAttribute[, ...]	A list of attributes to be related to this attribute using the relations specified in the relation list described above. A leading .. (two dots) points to an attribute in this object's parent object, and a leading ... (three dots) points to an attribute in this object's grandparent object.
AttrNameToLookFor	The name of an attribute in this business object whose table name should be used, along with the field name specified in Field Name To Retrieve parameter, to assign to the tablename and fieldname parameters for this attribute.
Field Name To Retrieve	Field name to use to assign to the tablename/fieldname parameters for this attribute. Used with the AttrNameToLookFor parameter.

Table 10. Name-value pairs in attribute application-specific text

Name-value pair	Description
SchemeName=*Value*	The constant SchemeName and its value as defined by the value in the name column of the Clarify CRM table_num_scheme table. This name-value pair is used to specify the automatic generation of one or more attribute values.
PP=*viewOrTable:column: condition:location*[...]	Specifies preprocessing of an attribute
RBC=*tablename*:*fieldname*	Specifies each attribute that is to be used in a RetrieveByContent operation.

Specifying the clarify table name and field name for an attribute

Attribute application-specific text must specify the tablename, fieldname, and datatype parameters for all attributes. These parameters enable the connector to create a new record in a table for a Create operation or locate a record in a table for a Retrieve operation or an Update operation. All other application-specific text fields are optional.

When you modify or add business object attributes, be sure to specify the name of the table, name of the field, and the data type for the attribute in the Clarify CRM database.

Example: Simple create operations: The following example shows a business object named Clarify_PartnerSite and its key attribute SiteObjid. In the

application-specific text for the SiteObjid attribute, the business object designer coded the location for the attribute value as the `objid` field in the Clarify CRM `site` table. The data type of the field is `integer`.

```
Name = SiteObjid
IsKey = true
AppSpecificInfo = site:objid::1:
```

The connector can use the application-specific text to create a new record in the `site` table. Figure 7 shows an example Clarify CRM `site` table and an example business object definition for a Clarify_PartnerSite business object.

Note: Do not name a Clarify CRM application-specific business object Clarify_Site. The business object Clarify_Site is obsolete, and the use of this name will trigger obsolete processing in the connector that may result in a user code exception.

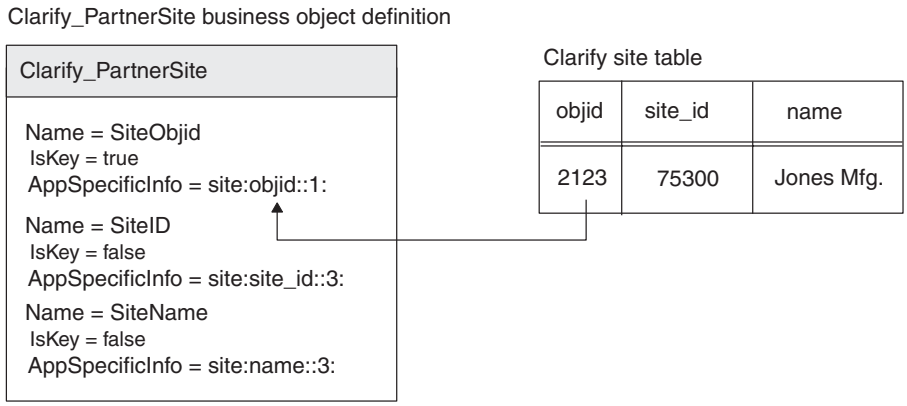

Figure 7. Attribute application-specific text specifying table name, field name, and data type

Specifying relations between objects

If an attribute references a Clarify CRM table other than the primary table, the attribute application-specific text must specify the `relation` and `RelateToAttribute` parameters. These parameters enable the connector to create a new record in the associated table for a Create operation or find a record in an associated table for a Retrieve operation.

The `relation` parameter must be included in the application-specific text of the first attribute that references the table. Subsequent attributes that reference the table cannot include a relation.

The `RelateToAttribute` parameter specifies the attribute that is to be related to this attribute by the relation. This parameter can point to an attribute in the current business object, the parent business object, or the grandparent business object using the following syntax:

- `RelateToAttribute` points to the current object
- `..RelateToAttribute` points to the parent object
- `...RelateToAttribute` points to the grandparent object

When you modify or add business object attributes that refer to associated tables, be sure to specify the name of the associated table, the name of the field, the data type for the attribute, the relation name, and the RelateToAttribute name.

The next sections provide examples of using several attribute application-specific text parameters.

Example: Creating an individual business object with relations to a single associated table: To define attributes that reference a table related to the primary table, define an attribute whose application-specific text contains the relation between the tables, and then include other attributes that reference the foreign table.

For example, a business object definition named Clarify_PartMaster, representing the Clarify `part_num` table, includes several attributes that reference the `mod_level` table. In the business object definition shown in Figure 8, the `part_num` table is specified as the primary table in the application-specific text of the business object's primary key, PartId. The relation between the `part_num` table and `mod_level` table is specified in the application-specific text of the ModLevelObjId attribute. The application-specific text of the following attribute, ModLevel, simply references the `mod_level` table.

The application-specific text in the ModLevelObjId attribute shows the use of the `tablename`, `fieldname`, `relation`, `datatype`, and `RelateToAttribute` parameters.

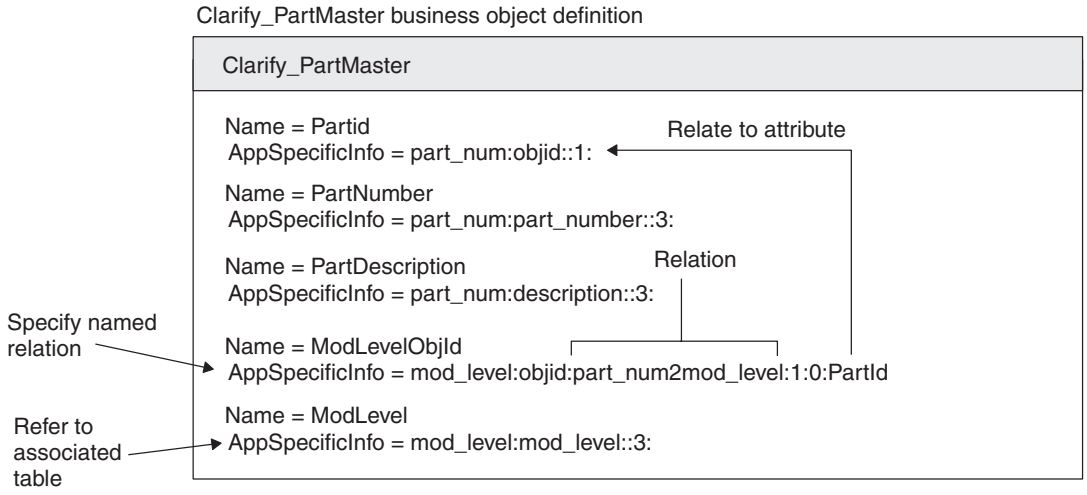

Figure 8. Business object with relation to a foreign table

Example: Creating a hierarchical business object with relations between parent and child business objects: In this example, assume that the business object Clarify_PartnerSite is a hierarchical business object containing a child business object named Clarify_SiteAddress. When the connector receives a Create request for Clarify_PartnerSite, it does the following:

- Creates a new site record
- Creates a new address record
- Relates the address record to the site record. The application-specific text in the child business object specifies the relation to create.

The Clarify_PartnerSite and Clarify_SiteAddress business object definitions are illustrated in Figure 9. The application-specific text in the primary key attribute of the child business object shows the use of the `tablename`, `fieldname`, `relation`, `datatype`, and `RelateToAttribute` parameters. The `RelateToAttribute` parameter specifies the SiteObjid attribute in the parent business object.

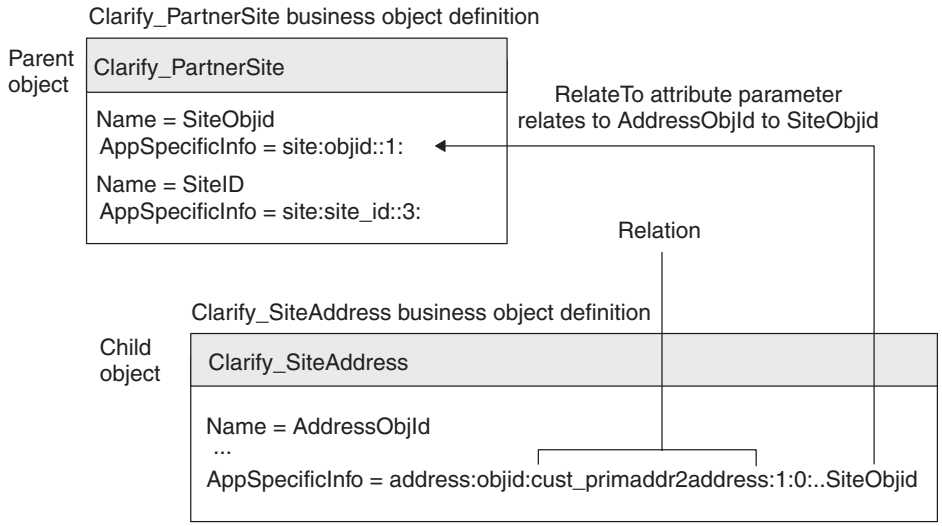

Figure 9. Creating relations within a hierarchical business object

The connector uses the application-specific text shown in the business objects above to process the business objects. The connector follows these general steps:

1. The connector creates a new site record in the site table. The value in the `objid` key becomes the value for the SiteObjid attribute in the parent Clarify_PartnerSite business object. The data type of the field is `string`.
2. The connector creates a new record in the `address` table. The value of the `objid` key becomes the value for the AddressObjId attribute in the child Clarify_SiteAddress business object. The data type of the field is `integer`.
3. From the child Clarify_SiteAddress object, the connector locates the `site` table using the ..SiteObjid parameter in the application-specific text. This parameter points to the SiteObjid attribute in the parent business object. The SiteObjid attribute in the parent business object provides the reference to the business object's primary table and primary key.
4. The connector relates the address record to the `site` table by populating the `cust_primaddr2address` relation field of the `site` table with the value of `address.objid` of the address record. The connector uses the inverse relation `cust_primaddr2address`.

Figure 10 shows the Clarify CRM site table and address table and the corresponding WebSphere Business Integration Adapter business object definitions.

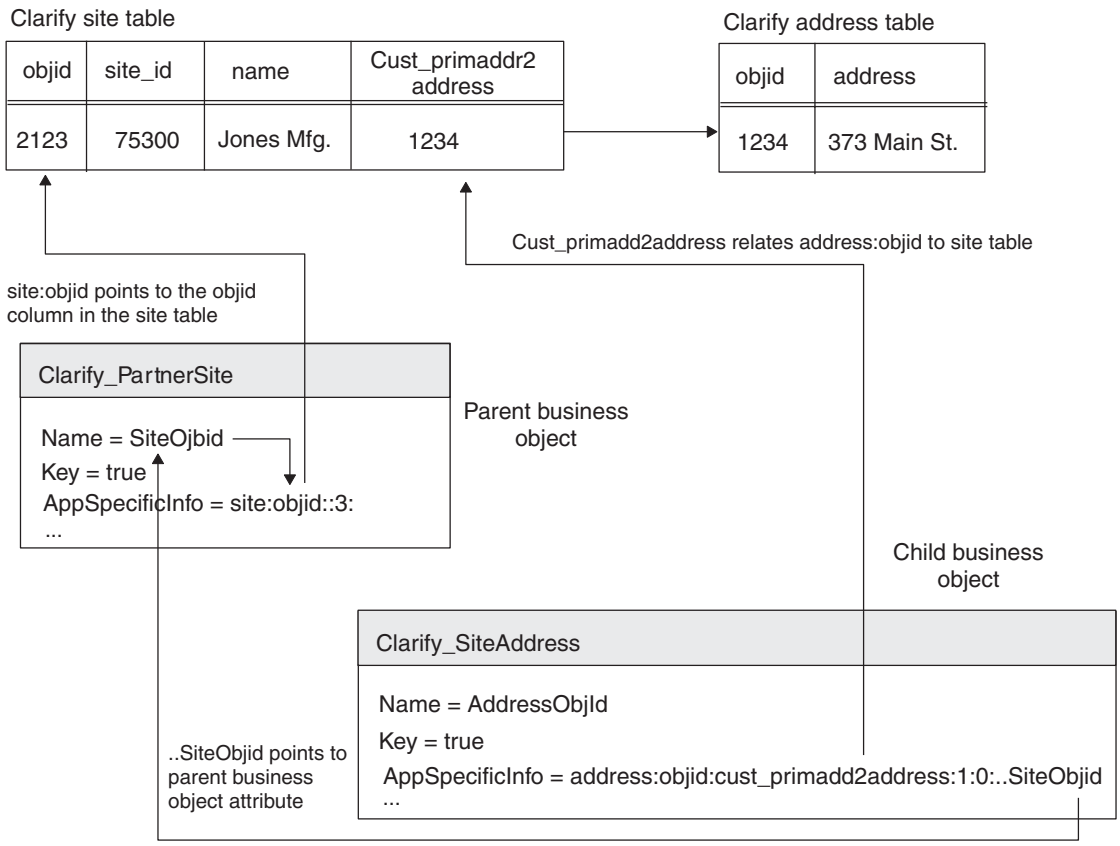

Figure 10. Creating a hierarchical business object

Example: Creating a hierarchical business object with multiple relations to the same field: In the next example, the child business object has multiple relations to the parent business object. The application-specific text instructs the connector to create a new site record, create a new address record, and link the address record to the site record as primary address, billing address, and shipping address. The three relations between the address table and the site table are created using each inverse relation in the list of relations: `cust_primaddr2address`, `cust_billadd2address`, and `cust_shipaddr2address`. The format is:

```
Name = AddressObjId
AppSpecificInfo = address:objid:cust_primaddr2address,
   cust_billaddr2address,cust_shipaddr2address:1:0:..SiteObjid,
   ..SiteObjid,..SiteObjid
```

The site table relation columns are populated with the address.objid of the related address record. Because all three customer types use the same address, the relation columns contain the same ID. Figure 11 shows these relations.

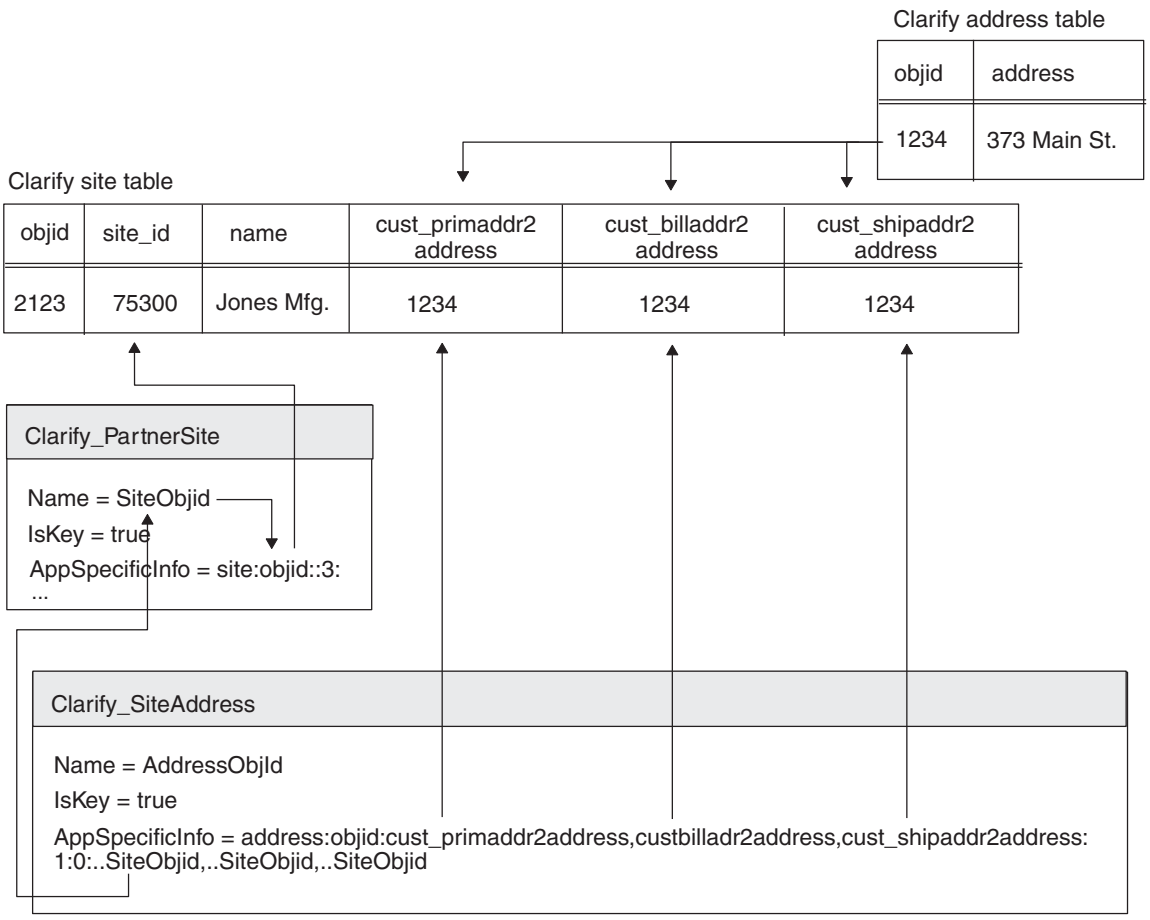

Figure 11. Multiple relations referring to the same field

Example: Using the relateToAttribute to point to parent and grandparent business objects: The business object Clarify_Contract_Schedule_Subline provides an example of application-specific text that points to both parent and grandparent business objects. The application-specific text is:

```
Name = ContractLineObjId
AppSpecificInfo =contr_itm:objid:schedule2contr_itm,parent2contr_itm:
1:0:...ContrSchedObjid,..ContractLineObjId
```

Using this text, the connector creates relations as follows:

- Using the `schedule2contr_itm` relation, the connector relates the current table, `contr_itm`, to the `contr_schedule` table, which is specified in the ContrSchedObjid key attribute of the grandparent business object Clarify_Contract_Schedule.
- Using the `parent2contr_itm` relation, the connector relates the current row in the `contr_itm` table to another row in the same table.

The second relation represents a line item hierarchy, where the current line item is a subordinate line item, and the parent line item is specified in the ContractLineOjbId key attribute of the parent business object. This line item hierarchy is represented by a WebSphere Business Integration Adapter business object hierarchy, where the parent line item is a Clarify_Contract_Schedule_Line business object, and the child line item is a Clarify_Contract_Schedule_Subline child business object.

Specifying the parentIndex parameter for retrieve operations

To retrieve data from the Clarify CRM database, the connector defines a query or a set of queries that selects the specified data. The connector determines the type of query based on whether the business object uses relations to join tables.

The connector builds queries as follows:

- For an individual business object whose attributes represent a single Clarify CRM table, the connector builds a single query to retrieve data from the specified fields.
- For an individual business object some of whose attributes use relations to join tables, the connector builds an array of queries.
- For a hierarchical business object some of whose attributes contain child business objects, the connector builds an array of queries.

To enable the connector to build an array of queries, the application-specific text must contain two index numbers: an index number in the ParentIndex parameter and the query index, which is generated by the connector. The connector uses these index numbers to define the query.

For an individual business object that uses relations to join tables, the connector generates the query index by indexing the first business object attribute as 0 and incrementing the index for each subsequent attribute that refers to a relation. For an attribute that refers to a relation, the application-specific information ParentIndex parameter specifies the query index number for the attribute that contains the other key for the join.

Figure 12 illustrates a sample business object and shows how the business object designer would index the attributes in the business object for a Retrieve operation. The illustration also shows the application-specific information for each attribute.

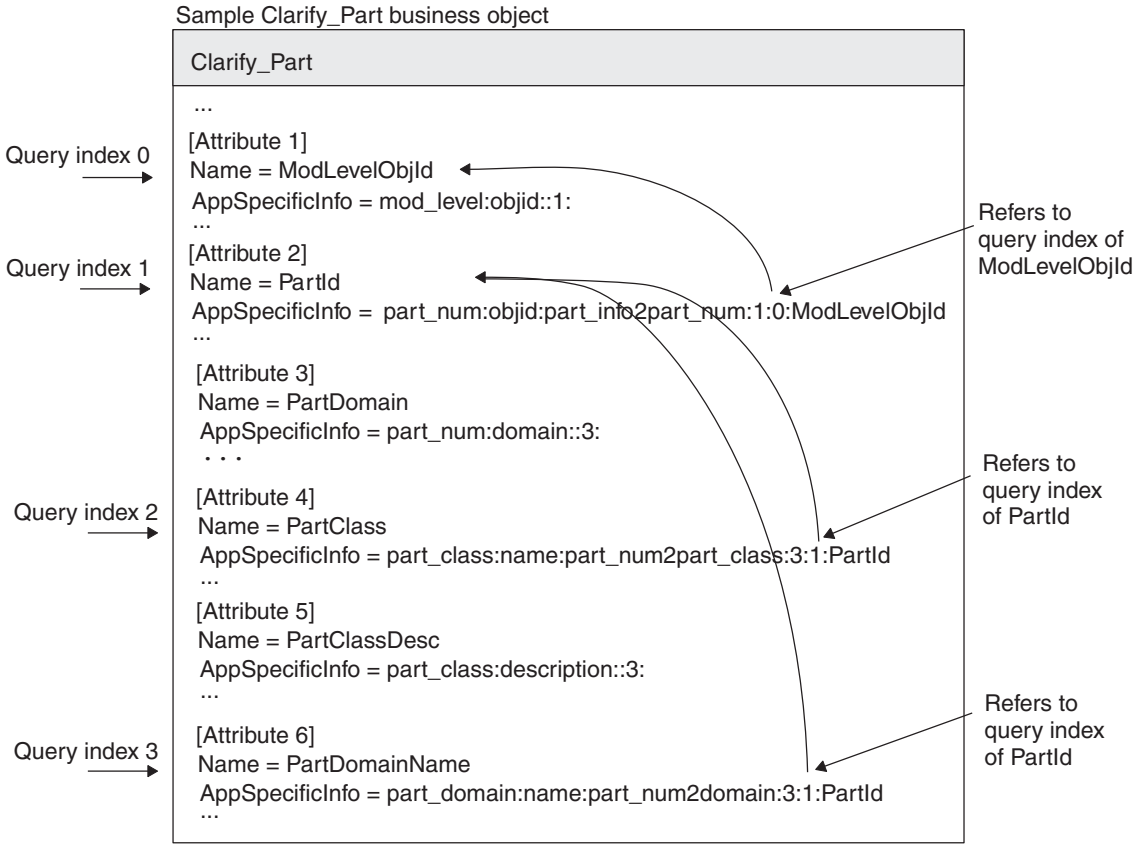

Figure 12. Query index and parentIndex parameters for a business object

When you create a business object, count the attributes that refer to new relations to determine what the query index will be. Then set the ParentIndex parameter to point to the correct query index.

Note: When modifying a business object, it is recommended that you add new attributes at the end of a business object. If you add an attribute in the middle of an existing business object or delete an attribute from a business object, and the business object will be used for Retrieve operations, you will have to re-index the Parent Indexes.

Retrieving hierarchical business objects

In a Retrieve operation on a hierarchical business object, the connector typically gets the key from the top-level business object and then recursively works through the entire business object retrieving all the child business objects. To do this, the connector builds a query for the parent object, and builds a new query for each child business object.

The query index for the parent business object is generated as described above. The query index for each child business object implicitly includes the parent key as index 0, and then increments the index for each new relation. Each child query index is generated in the same way as top-level objects except that it begins at 1.

Figure 13 shows how the connector indexes attributes for a parent business object and a child business object. The figure also shows the application-specific information.

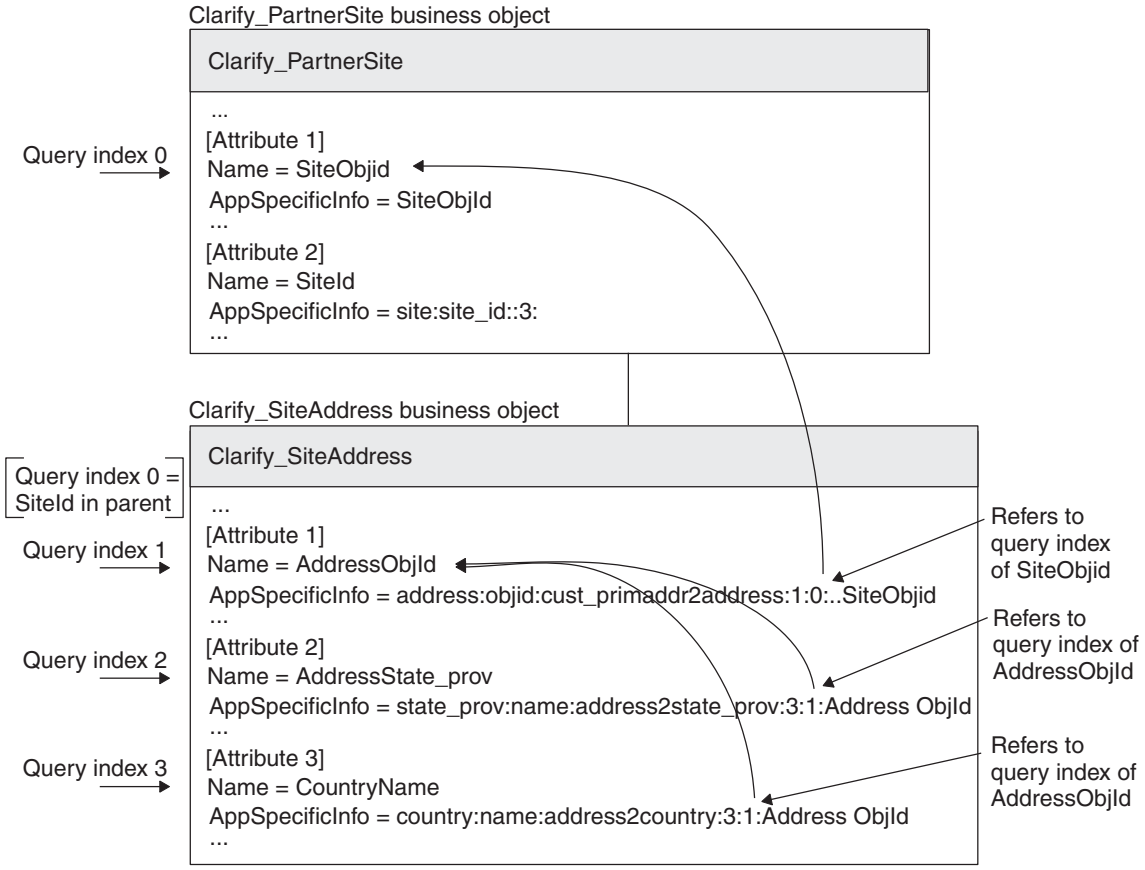

Figure 13. Query index and parent index for a hierarchical business object

A business object can have any number of attributes that refer to relations. Figure 14 shows another example of query indexes and parent indexes in hierarchical business objects.

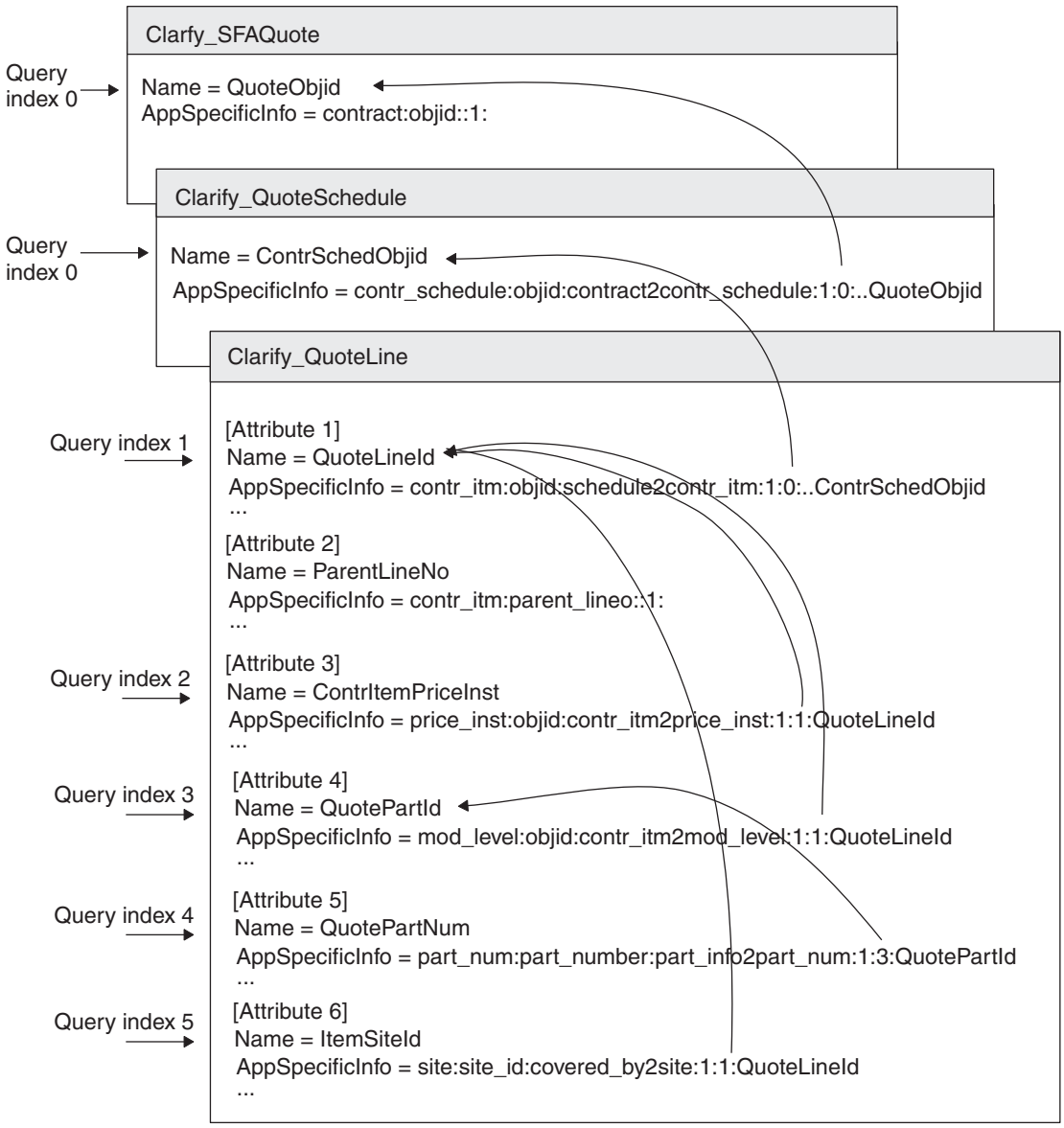

Figure 14. Another example of query indexes and parent indexes

Summary of rules for using the parentIndex parameter: If you are modifying a business object for Clarify CRM and want to use the business object for Retrieve operations, you will need to set up the ParentIndex parameter in the business object application-specific text. Follow these rules:

- The query index of the first attribute of a parent object is 0. To use that attribute as a key for a subsequent attribute in the same business object, set the ParentIndex to 0.
- The query index of the first attribute of a child object is 1. To access the parent object key in the child/parent relation, set the ParentIndex to 0 to point to the parent object.
- Grandchild objects work like child objects. The first attribute query index is set to 1. To access the grandparent object key from the grandchild object, set the ParentIndex to 0, just as if there were no object between the parent and the grandchild.

- The query index for all objects increments by 1 for every attribute that refers to a relation. To use an attribute as a key for a different attribute, set the ParentIndex to the query index for that attribute.

Assigning an attribute a value from another table

You can use the `AttrNameToLookFor` and `FieldNameToRetrieve` parameters to retrieve data from a field in another table and assign the value to the current attribute. To do this, you specify a table name using the `AttrNameToLookFor` parameter and a field name using the `FieldNameToRetrieve` parameter. This assigns the value in that field to the table and field for the current attribute. An example of application-specific text for this is:

```
AppSpecificInfo = site_part:site_objid::1:::CustomerId:objid
```

In this example, the connector retrieves the value of `objid` from the table pointed to by attribute name CustomerId and assigns it to the `site_objid` column of the `site_part` table. Figure 15 illustrates this application-specific text.

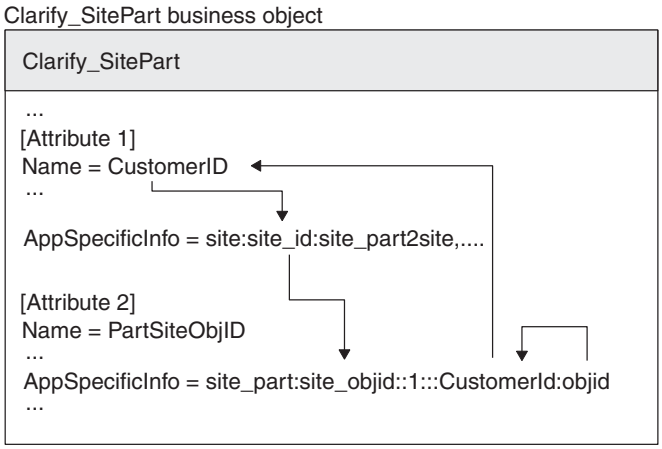

Figure 15. Assigning an attribute a value from an attribute in another table

Note that these parameters can only be used to retrieve values for attributes within a single business object; they cannot be used to retrieve values across business objects.

Preprocessing attributes

The application-specific information for an attribute can be coded so that the connector obtains the value for the attribute before executing a business object request, such as an Update or Retrieve.

To use preprocessing, you must edit the business object definition. In the application-specific information for the attribute you want to preprocess, use the following format:

```
PP= viewOrTable:column:condition:location[:condition2:location2]
```

where:

viewOrTable	is the name of the database view or table used for the preprocessing.
column	is the name of the column where the connector will search for the value of this attribute
condition	is the name of an attribute whose value is used to retrieve this attribute from location. The value cannot be a Clarify CRM relation

location	is the name of the column where the connector searches for the condition value
condition2	is the name of the optional second attribute whose value is used to retrieve this attribute. The value cannot be a Clarify CRM relation.
location2	is the name of the column where the connector searches for the condition2 value
condition3	is the name of the optional third attribute whose value is used to retrieve this attribute. The value cannot be a Clarify CRM relation.
location3	is the name of the column where the connector searches for the condition3 value

Add the information to the application-specific information as follows:

- If there is no vertical bar (|) separating items in the application-specific information, add this information to the left of the existing positional parameters, and add a vertical bar after it.
- If there is already a name-value pair and a vertical bar in the application-specific information, add this parameter to the list of name-value pairs. The order of name-value pairs is not significant. Separate sets of name-value pairs with semi-colons.

Example: Preprocessing attributes: View x_mp has three fields: modlevel_objid, modlevel_name, and part_id. The connector can retrieve the value of the ModLevelObjId attribute using the values of the attributes ModLevel and ItemId.

The ModLevelObjId attribute's application-specific text is:

```
PP=x_mp:modlevel_objid:ModLevel:modlevel_name:ItemId:part_id|...
```

The connector obtains the values of the conditions and uses them to obtain the value of the attribute whose preprocessing this represents. The connector could use the following SQL statement when preprocessing the ModLevelObjid attribute, if the value found in the modlevel_name column is x, and the value found in the ItemID column is y.

```
Select modlevel_objid from table_x_mp where modlevel_name = x and part_id = y
```

Generating attribute values automatically

You can use application-specific text for an attribute to automatically generate values for attributes such as IDs. This application-specific text uses the Clarify CRM table table_num_scheme, which contains all the ID fields for Clarify CRM tables. If you want to add IDs to this table, see the Clarify CRM documentation for information.

You can automatically generate the value of one or more attributes in a single business object. To automatically generate values for an attribute, you must edit its application-specific information. In the application-specific information, use the following format:

```
AppSpecificInfo = SchemeName=Value
```

where Value specifies the name of the scheme from the name column of the table_num_scheme table.

Add the name-value pair to the application-specific information as follows:

- If there is no vertical bar (|) separating items in the application-specific information, add this information to the left of the existing positional parameters, and add a vertical bar after it.

- If there is already a name-value pair and a vertical bar in the application-specific information, add this parameter to the list of name-value pairs. The order of name-value pairs is not significant. Separate name-value pairs with a semi-colon.

Note the following rules:

- The attribute must not contain a value. If the business object contains a value for the attribute, the value is used even if the application-specific text is set to SchemeName=Value.
- The connector-specific configuration property UseClarifyID must be set to True.

Example: Generating ID values for an attribute: For example, to generate sequential ID values for the SiteID attribute in the example Clarify_PartnerSite business object, use the following text:

```
AppSpecificInfo = SchemeName=Site ID|site:site_id::3:
```

Retrieving by non-key values

A business process might need to retrieve a business object for which it has a set of attribute values without having the one key attribute that uniquely identifies an application record. Such a retrieve is called ″retrieve by non-key values″ or ″retrieve by content.″

The connector allows retrieval by non-key values if the following conditions are met:

- The business object sent as a request must have the verb RetrieveByContent.
- Each attribute that can contain a value used in the Retrieve operation or receive a value as a result of the operation is designated in the application-specific text in the business object definition.

When the connector receives a request for a RetrieveByContent operation, it queries Clarify CRM for the appropriate record by using all the non-null values for attributes designated as RetrieveByContent (RBC). When the connector gets a response, it fills in values for the designated attributes that were previously null.

The connector can return the business object with all designated attributes filled in or follow-up with a subsequent Retrieve operation to get the entire business object. However, the connector does a subsequent Retrieve only if the application-specific business object definition specifies that it should.

Setting up a business object for retrieving by non-key values: To define a business object for retrieving by non-key values, do the following:

- Specify the attributes to be used for the retrieve.
- Specify whether to follow up the non-key retrieve with a subsequent retrieve.

Specifying the attributes: Each attribute that will supply a value to the Retrieve operation or receive a value from the retrieve must be defined by adding the following parameter to the attribute's application-specific information:

```
RBC=tableName:fieldName
```

The `tableName` argument is the name of a view or table in the database. This value must be the same for all attributes used in a non-key retrieve. The use of a view lets you include values from different tables.

The first attribute of the business object must always be coded for non-key retrieve in order for any attribute to be used. This ensures that the key value will be filled in as a result of the operation.

Add the information to the application-specific information as follows:

- If there is no vertical bar (|) separating items in the application-specific information, add this information to the left of the existing positional parameters, and add a vertical bar after it.
- If there is already a name-value pair and a vertical bar in the application-specific information, add this parameter to the list of name-value pairs. The order of name-value pairs is not significant. Separate name-value pairs with a semi-colon.

Specifying subsequent retrieve: To define a business object to follow a Retrieve by non-key values operation with a subsequent regular Retrieve operation, place the following property in the application-specific information for the RetrieveByContent verb:

```
SubsequentRetrieve=true
```

The default value for this property is false. The connector performs a deep or shallow retrieve based on the value of the RequestRetrieve configuration parameter.

Application-specific text for container attributes

For Update operations, you can edit the application-specific text for attributes that contain child business objects to specify that the connector keep any existing relations between tables and add new relations required by new child business objects. The format for this application-specific text is:

```
AppSpecificInfo = keeprelations
```

Case is ignored in checking for this application-specific text.

If this parameter is not specified, the connector deletes existing relations between the parent business object and child business object tables and then adds any new relations required by new child business objects.

Note: If the parent business object is non-existent, the update fails.

As an example, if an existing contract is associated with an existing site, such as New York, and the connector receives an Update Contract business object with a child business object representing a new site, such as San Francisco, and the container attribute application-specific text has the value `keeprelations`, the contract will be associated with both New York and San Francisco. If `keeprelations` is not specified, the contract will be associated only with San Francisco.

Figure 16 shows how the keeprelations value is set in the System Manager for the Contract business object to associate a new ContractLine with an existing Contract.

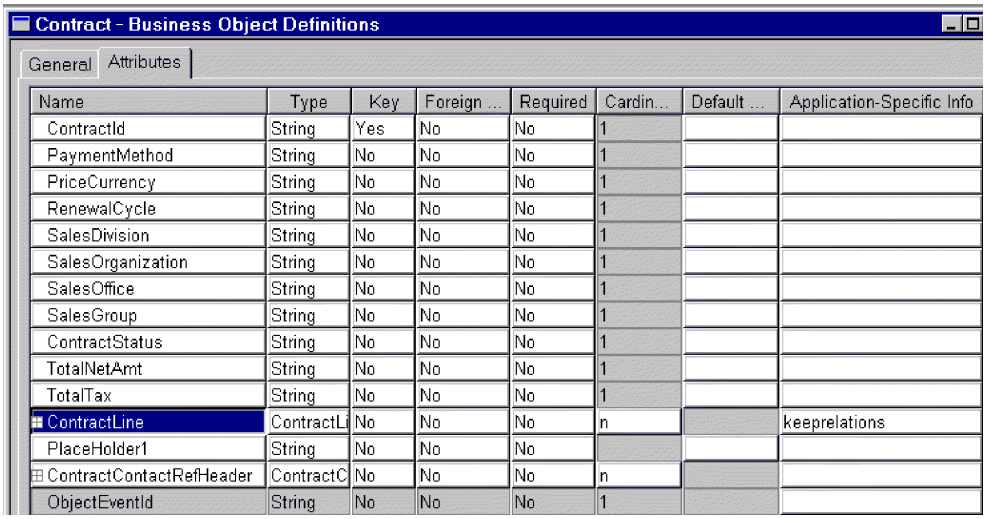

Figure 16. Example of the keeprelations value in the contract business object

Note: If updating data in a child object when there are multiple child objects, you must provide the child object ID (not `CxIgnore`) in order to preserve relations to the other children **even when keeprelations is used**. This is different from using `CxMissingID`, which would add a new child object (and keep relations to the existing children) instead of updating the data for an existing child. The child ID is required since the connector needs to know which child to update and will delete relations to all children except the first if it does not know the child ID to be updated.

The connector uses verb application-specific text to specify the following types of Retrieve operations:

- Retrieve only the parent business object in a hierarchical business object
- Perform a retrieve operation after a retrieve by non-key values

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

Retrieve verb

The default action for a Retrieve verb is to retrieve the parent business object and all related child objects. You can edit the application-specific text for the Retrieve verb to specify that the connector retrieve only the parent business object in a hierarchical object. This feature is used for event processing only.

The application-specific text to retrieve only the parent object is `RetrieveAll=False`. This text is associated with the Retrieve verb, but it applies to all verbs. An example of a Retrieve verb with this text is:

```
[Verb]
Name = Retrieve
AppSpecificInfo = RetrieveAll=False
```

Specifying a retrieve following a retrieve by non-key values

The RetrieveByContent verb specifies a set of attribute values other than the key attribute for the connector to use when building a Retrieve query. In a RetrieveByContent operation, the connector returns values for the business object key and for other designated attributes. You can edit the application-specific text

for the RetrieveByContent verb to specify that the connector follow a RetrieveByContent operation with a subsequent Retrieve operation to retrieve the entire business object.

To specify that the connector follow a RetrieveByContent operation with a Retrieve operation, place the following property in the application-specific text for the RetrieveByContent verb:

```
AppSpecificInfo = SubsequentRetrieve=true
```

The default value for this property is false.

Chapter 4. Troubleshooting

The chapter describes problems that you may encounter when starting up or running the connector.

Start-up problems

If you encounter difficulties when trying to start the connector, check to make sure that the integration broker is up and running.

The second parameter after the `start_connector` command should contain the integration broker name.

If you get an error similar to `symbol not found: __1cDstdNbasic_ostream4Ccn0ALchar_traits4Cc___2l6Ml_r1_ (../connectors/Clarify/libClarify.so)`, on the Solaris platform, make sure that you have `libCstd.so.1` in `/usr/lib` and that it is also included in `LD_LIBRARY_PATH`.

Event processing

If there are events in the event table, and they are not being processed while the connector is running, check to make sure the relevant business process is running.

Verifying DB_NAME during startup

The Clarify connector verifies the column "DB_NAME" within the table adp_db_header during startup. This entry must match the value which is set for the property database_name in the ClarifyConnector properties or a failure occurs.

Mapping (ICS Integration Broker only)

If the business objects are not being mapped or mapping is not being invoked, check to make sure the maps have been installed in the correct directory.

Problems with date in business objects (ICS Integration Broker only)

If the date is wrong in business objects, check the Windows short date style under Regional Settings, Date in the Control Panel. The date format must correspond to the date format used in maps for the business objects for Clarify CRM. Note that WebSphere Business Integration Adapter native maps assume that the Windows short date style is MM/dd/yyyy. If the short date style is MM/dd/yy or any other format, change the map rules to reflect the new date format and recompile the maps.

Problems with business objects named Clarify_Site

Using the name Clarify_Site for a Clarify CRM application-specific business object will cause errors. The business object Clarify_Site is obsolete, and the use of this name will trigger obsolete processing in the connector that may result in a user code exception. Therefore, do not name a Clarify CRM application-specific business object Clarify_Site.

Combining preprocessing and use of relationship

When the relationship points to a Clarify View, combining preprocessing and use of a relationship leads to wrong data. To resolve this issue, implement the following: 1) Use preprocessing to get the value from the view. 2) Use the relationship to a Clarify table and not to a view.

Loss of connection to the application

If the connector determines that the connection with the application has been lost, the connector terminates and returns an application response status to the integration broker.

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 sections on Connector Configurator in this guide.

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.
- **Agent restart (ICS only)**
  The change takes effect only after you stop and restart the application-specific component.
- **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.

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 Table 11 on page 53 below.

Summary of standard properties

Table 11 on page 53 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.

Note: In the "Notes"column in Table 11 on page 53, the phrase "Repository directory is REMOTE" indicates that the broker is the InterChange Server. When the broker is WMQI or WAS, the repository directory is set to LOCAL

Table 11. 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> (broker is ICS)
AgentTraceLevel	0-5	0	Dynamic	
ApplicationName	Application name	Value specified for the connector application name	Component restart	
BrokerType	ICS, WMQI, WAS		Component restart	
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> (broker is ICS)
ContainerManagedEvents	No value or JMS	No value	Component restart	Delivery Transport is JMS
ControllerStoreAndForwardMode	true or false	true	Dynamic	Repository directory is <REMOTE> (broker is ICS)
ControllerTraceLevel	0-5	0	Dynamic	Repository directory is <REMOTE> (broker is ICS)
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

Table 11. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
DuplicateEventElimination	`true or false`	`false`	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	Component restart	JMS transport only
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> (broker is ICS)
JvmMaxNativeStackSize	Size of stack in kilobytes	`128k`	Component restart	Repository directory is <REMOTE> (broker is ICS)
JvmMinHeapSize	Heap size in megabytes	`1m`	Component restart	Repository directory is <REMOTE> (broker is ICS)
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	

Table 11. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
LogAtInterchangeEnd	`true` or `false`	`false`	Component restart	Repository Directory must be <REMOTE> (broker is ICS)
MaxEventCapacity	`1-2147483647`	`2147483647`	Dynamic	Repository Directory must be <REMOTE> (broker is ICS)
MessageFileName	Path or filename	`CONNECTORNAMEConnector.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> (broker is ICS)
OADMaxNumRetry	A positive number	`1000`	Dynamic	Repository Directory must be <REMOTE> (broker is ICS)
OADRetryTimeInterval	A positive number in minutes	`10`	Dynamic	Repository Directory must be <REMOTE> (broker is ICS)
PollEndTime	`HH:MM`	`HH:MM`	Component restart	
PollFrequency	A positive integer in milliseconds `no` (to disable polling) `key` (to poll only when the letter `p` is entered in the connector's Command Prompt window)	`10000`	Dynamic	
PollQuantity	`1-500`	`1`	Agent restart	JMS transport only: Container Managed Events is specified
PollStartTime	`HH:MM`(HH is 0-23, MM is 0-59)	`HH:MM`	Component restart	

Table 11. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
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.
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>
WsifSynchronousRequestTimeout	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 (Object Request Broker) connections opened by `orb.init[]`.

The default value of this property is set to 1. You can change it as required.

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-down list. To add other supported values to the drop-down list, you must manually modify the `\Data\Std\stdConnProps.xml` file in the product directory. For more information, see the sections on Connector Configurator in this guide.

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.

There is no default value.

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

- PollQuantity = 1 to 500
- SourceQueue = /SOURCEQUEUE

You must also configure a data handler with the MimeType, DHClass (data handler class), and DataHandlerConfigMOName (the meta-object name, which is optional) properties. To set those values, use the **Data Handler** tab in Connector Configurator.

Thes properties are adapter-specific, but **example** values are:

- MimeType = text\xml
- DHClass = com.crossworlds.DataHandlers.text.xml
- DataHandlerConfigMOName = MO_DataHandler_Default

The fields for these values in 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 the RepositoryDirectory is remote, 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. Use the default when connecting to a local message broker.

When you connect to a remote message broker, this property takes the following (mandatory) values:
QueueMgrName:<Channel>:<HostName>:<PortNumber>,
where the variables are:
QueueMgrName: The name of the queue manager.
Channel: The channel used by the client.
HostName: The name of the machine where the queue manager is to reside.
PortNumber: The port number to be used by the queue manager for listening.

For example:
`jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456`

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-down list. To add other supported values to the drop-down list, you must manually modify the `\Data\Std\stdConnProps.xml` file in the product directory. For more information, refer to the sections on Connector Configurator in this guide.

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

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

LogAtInterchangeEnd

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 in the product directory. Specify the message filename in an absolute path if the message file is not located in the standard location.

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

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

MonitorQueue

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

The default value is CONNECTORNAME/MONITORQUEUE

OADAutoRestartAgent

Valid only when the RepositoryDirectory is <REMOTE>.

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

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

The default value is `false`.

OADMaxNumRetry

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

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

The default value is `1000`.

OADRetryTimeInterval

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

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

The default is `10`.

PollEndTime

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

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

PollFrequency

This is the interval between the end of the last poll and the start of the next poll. `PollFrequency` specifies the amount of time (in milliseconds) between the end of one polling action, and the start of the next polling action. This is not the interval between polling actions. Rather, the logic is as follows:

- Poll to obtain the number of objects specified by the value of `PollQuantity`.
- Process these objects. For some adapters, this may be partly done on separate threads, which execute asynchronously to the next polling action.
- Delay for the interval specified by `PollFrequency`.
- Repeat the cycle.

Set `PollFrequency` to one of the following values:

- The number of milliseconds between polling actions (an integer).
- 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. Where they exist, these restrictions are documented in the chapter on installing and configuring the adapter.

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.

FIX

An email message is also considerd an event. The connector behaves as follows when it is polled for email.

Polled once - connector goes to pick 1. the body of the message as it is also considered an attachment also. Since no DH was specified for this mime type, it it will ignore the body. 2. conector process first PO attachment. DH is avaiable for this mime type so it sends the business object to the Visual Test Connector. If the 3. accept in VTC again no BO should come thru Polled second time 1. conector process second PO attachment. DH is avaiable for this mime type so it sends teh BO to VTC2. accept in VTC again now the third PO attachment should come through. This is the correct behaviour.

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 JMSand 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 58.

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.

WsifSynchronousRequestTimeout

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 70).

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 71 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 Tools>Connector Configurator**.
- Select **File>New>Connector Configuration**.
- 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 75.)

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:

- 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.
- From Connector Configurator, select **File>Open**. Select the name of the connector configuration file from a project or from the directory in which it is stored.
- 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 connector definition as the template.

- To create a new template, see "Creating a new template" on page 71.
- To use an existing file, simply modify an existing template and save it under the new name. You can find existing templates in your `\WebSphereAdapters\bin\Data\App` directory.

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 in Connector Configurator:

1. Click **File>New>Connector-Specific Property Template**.
2. The **Connector-Specific Property Template** dialog box appears.
   - Enter a name for the new template in the **Name** field below **Input a New Template Name.** You will see this name again when you open the dialog box for creating a new configuration file from a template.
   - 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 appears in the **Template Preview** display.
3. You can use an existing template whose property definitions are similar to those required by your connector as a starting point for your template. If you do not see any template that displays the connector-specific properties used by your connector, you will need to create one.

- If you are planning to modify an existing template, select the name of the template from the list in the **Template Name** table below **Select the Existing Template to Modify: Find Template.**
- This table displays the names of all currently available templates. You can also search for a template.

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**, enter your values.

To create a new property value:

1. Select the property in the **Edit properties** list and right-click on it.
2. From the dialog box, select **Add**.
3. Enter the name of the new property value and click OK. The value appears in the **Value** panel on the right.

The **Value** panel displays a table with three columns:

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

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

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

After a value has been created and appears in the grid, it can be edited from within the table display.

To make a change in an existing value in the table, select an entire row by clicking on the row number. Then right-click in the **Value** field and click **Edit Value**.

Setting dependencies

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

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

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

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

   <=(less than or equal to)
4. In the **Conditional Value** field, enter the value that is required in order for the dependent property to be included in the template.
5. With the dependent property highlighted in the **Available Properties** display, click an arrow to move it to the **Dependent Property** display.
6. Click **Finish**. Connector Configurator 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, you must name it and select an integration broker.

- 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.
- In stand-alone mode: from Connector Configurator, select **File>New>Connector Configuration**. In the New Connector window, enter the name of the new connector.

You also need 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 **Integration Broker** field, select ICS, WebSphere Message Brokers or WAS connectivity.
- drop-downte the remaining fields in the **New Connector** window, as described later in this chapter.

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 name. You can fill in all the field values to drop-downte 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 78..
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 displayed for each property. It indicates whether a component or agent restart is necessary to activate changed values. You cannot configure this setting.

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 Connector-specific Properties 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 52.

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 an ICL (Integration Component Library) 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 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 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. By default, the file is saved to `\WebSphereAdapters\bin\Data\App`.
- You can also save it to a WebSphere Application Server project if you have set one up.

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

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

Changing a configuration file

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

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

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

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

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

Appendix C. 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.

COPYRIGHT LICENSE

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

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