IBM WebSphere Business Integration Adapters

Adapter for Siebel eBusiness Applications User Guide

Adapter Version 4.6.x

IBM WebSphere Business Integration Adapters

Adapter for Siebel eBusiness Applications User Guide

Adapter Version 4.6.x

Note!

Before using this information and the product it supports, read the information in “Notices” on page 123

30September2004

This edition of this document applies to WebSphere Business Integration Adapter for Siebel eBusiness Applications (5724-H43), Version 4.6.

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

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

Contents

About this document

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

What this document includes

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

What this document does not include

This document does not describe deployment metrics and capacity planning issues such as server load balancing, number of adapter processing threads, maximum and minimum throughputs, and tolerance thresholds.

Such issues are unique to every customer deployment and must be measured within or close to the exact environment where the adapter is to be deployed. You should contact your IBM services representative to discuss the configuration of your deployment site, and for details on planning and evaluating these kinds of metrics, given your specific configuration.

Audience

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

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

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

Related documents

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

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

You can install documentation from the following sites:

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

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

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

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

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

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

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

Note: Important information about this product <<or “the products documented in this guide” or whatever works for your doc>> 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, file name, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
italic, *`italic`*	Indicates a variable name or a cross-reference.
blue outline	A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference.
ProductDir	Represents the directory where the product is installed.

New in this release

New in release 4.6.x

Updated in September, 2004

This guide has been updated to include the following changes:

- The adapter supports Siebel version 7.7.
- Adapter support for Siebel version 7.7 is provided on two additional platforms: Solaris 9.0 and Windows 2003.
- The adapter is modified so that it can process business services where the request contains only simple attributes and no SiebelMessage container attribute.
- Support for result set retrieval, which is enabled only with specific versions of DB2® Information Integrator broker. (For information about these specific versions, refer to the DB2 Information Integrator product documentation.)

New in release 4.5.x

Updated in June, 2004. Beginning with version 4.5.x, the Adapter for Siebel eBusiness Applications is no longer supported on Solaris 7, so references to that platform have been removed from this guide.

New in release 4.4.x

February 2004

The adapter supports retrieval of multiple records using wrapper objects and the `RetrievebyContent` verb.

December 2003

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

- The adapter now supports custom-written business services.
- Support for the generation of WebSphere Business Integration business object definitions for custom business services is added to the Object Discovery Agent (ODA). The generation process is similar to that of Siebel business objects and components.
- Two new configuration properties, EventProcessingSupport and SiebelVersion, have been added.
  - EventProcessingSupport can be used to switch off subscription delivery if necessary and takes a value of true or false, with the default value being true.
  - SiebelVersion enables the adapter to run against a specified version of the Siebel application while preventing it from accessing the Schema Version Siebel business object and business component. Valid values are 6 or 7, and the default value is `NONE`. Use of the default value is recommended.
- The Siebel business components CW Events and CW Archive have been renamed IBM Events and IBM Archive. The adapter uses these components as it did in earlier versions. For backward compatibility, the adapter works with both the old and new component names. For example, the adapter will check first for IBM Events, and if found it will use it as the event store. If IBM Events is not

found, the adapter will check for CW Events. If it does not find either, and the adapter is configured for subscription delivery, it will return an error and terminate.

Note: Adapter installation information has been removed from this guide. See Chapter 2 for the new location of that information.

New in release 4.3.x

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

- Solaris 7,8
- AIX® 5.x
- HP-UX 11i

New in release 4.2.x

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

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

New in release 4.1.x

The connector delivered with IBM WebSphere Business Integration Adapter for Siebel eBusiness Applications has been internationalized. For more information, see "Processing locale-dependent data" on page 7 and Appendix A, "Standard configuration properties for connectors," on page 83

New in release 4.0.x

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

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

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

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

Chapter 1. Overview

This chapter provides an overview of adapter terminology and the WebSphere Business Integration Adapter for Siebel eBusiness Applications. It is important that you understand the topics in this chapter before you attempt to install, configure, and use the adapter.

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

Terminology

The following terms are used in this document.

adapter
The component in the WebSphere business integration system that provides components to support communication between an integration broker and either an application or a technology. An adapter always includes a connector, message files, and configuration tools. It can also include an Object Discovery Agent (ODA) or a data handler.

adapter framework
The software that IBM provides to configure and run an adapter. The runtime components of the adapter framework include the Java™ runtime environment, the connector framework, and the Object Discovery Agent (ODA) runtime. This connector framework includes the connector libraries (C++ and Java) needed to develop new connectors. The ODA runtime includes the library in the Object Development Kit (ODK) needed to develop new ODAs. The configuration components include the following tools:

- Business Object Designer,
- Connector Configurator,
- Log Viewer,
- System Manager,
- Adapter Monitor,
- Test Connector
- and, optionally, any Object Discovery Agents (ODAs) associated with an adapter.

Adapter Development Kit (ADK)
: A development kit that provides some samples for adapter development, including sample connectors and Object Discovery Agents (ODAs).

connector
: The component of an adapter that uses business objects to send information about an event to an integration broker (event notification) or receive information about a request from the integration broker (request processing). A connector consists of the connector framework and the connector’s application-specific component.

connector framework
: The component of a connector that manages interactions between a connector’s application-specific component and the integration broker. This component provides all required management services and retrieves the meta-data that the connector requires from the repository. The connector framework, whose code is common to all connectors, is written in Java and includes a C++ extension to support application-specific components written in C++.

connector controller
: The subcomponent of the connector framework that interacts with collaborations. A connector controller runs within InterChange Server and initiates mapping between application-specific and generic business objects, and manages collaboration subscriptions to business object definitions.

integration broker
: The component in the WebSphere business integration system that integrates data among heterogeneous applications. An integration broker typically provides a variety of services that include: the ability to route data, a repository of rules that govern the integration process, connectivity to a variety of applications, and administrative capabilities that facilitate integration. Examples of integration brokers: the WebSphere Business Integration Message Broker; WebSphere Business InterChange Server.

WebSphere business integration system
: An enterprise solution that moves information among diverse sources to perform business exchanges, and that processes and routes information among disparate applications in the enterprise environment. The business integration system consists of an integration broker and one or more adapters.

Siebel application architecture

The Siebel application architecture contains three layers, as follows:

- User interface objects layer--This layer contains the visual elements that the user interacts with.
- Business objects layer--This layer contains both business components and business objects. A business component is a fundamental business entity, consisting of multiple fields that represent it. A business object is a collection of related business components. The Siebel connector communicates with this layer using the Siebel Java Data Bean.
- Data objects layer--This layer contains the object definitions which provide logical representation of the underlying physical database. It is independent of the installed relational database management system, and it is not accessible by the Siebel Java Data Bean.

Connector architecture

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

The following diagram illustrates the Siebel connector architecture.

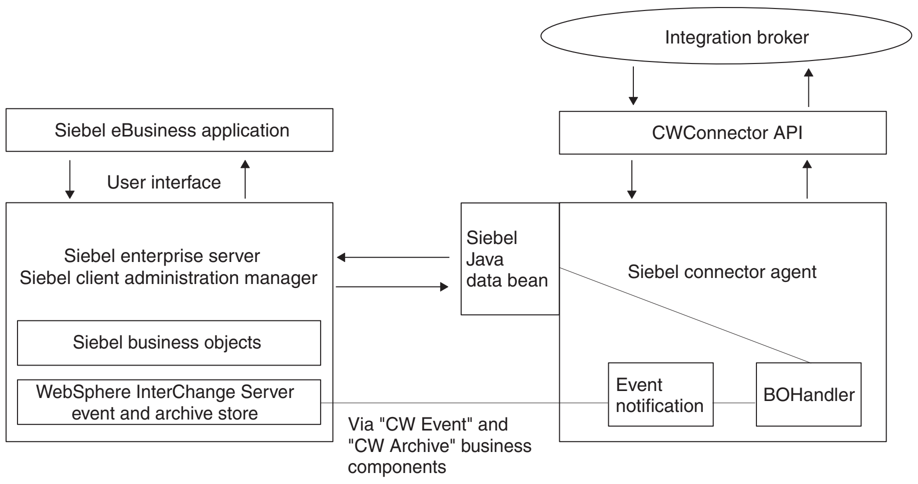

Integration broker
CWConnector API
Siebel eBusiness application
User interface
Siebel enterprise server
Siebel client administration manager
Siebel business objects
WebSphere InterChange Server event and archive store
Siebel Java data bean
Siebel connector agent
Event notification
BOHandler
Via "CW Event" and "CW Archive" business components

Figure 1. Siebel connector architecture

How the connector works

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

The connector and meta-data

The connector is meta-data-driven. Meta-data is application-specific data that is stored in business objects and that assists the connector in its interaction with the application. A meta-data-driven connector handles each business object that it supports based on meta-data encoded in the business object definition rather than on instructions hardcoded in the connector. A business object corresponds to a Siebel business component. For more information about business objects, see Chapter 6, “Using the adapter with Siebel business services,” on page 73

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

Business object processing

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

Processing integration broker requests

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

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

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

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

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

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

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

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

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

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

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

Processing application events

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

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

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

Table 1. Events business component structure

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

Table 1. Events business component structure (continued)

Fields	Description
Event Ts	Event creation timestamp

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

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

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

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

Event management

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

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

The setting of its ArchiveProcessed property determines whether the connector archives an event into the Archive business component after updating its status. For more information on the ArchiveProcessed property, see "ArchiveProcessed" on page 109.

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

Table 2. Archiving behavior

Archive processed setting	Event processing status	Connector behavior
`true` or no value	Successful	Event is deleted from the Events business component and archived with status of Success
	Unsuccessful	Archived with status of Error

Table 2. Archiving behavior (continued)

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

Smart filtering

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

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

Handling lost connections to the Siebel application

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

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

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

Processing locale-dependent data

The connector has been internationalized so that it can support double-byte character sets, and deliver message text in the specified language. When the connector transfers data from a location that uses one character code set to a location that uses a different code set, it performs character conversion to preserve the meaning of the data. The Java runtime environment within the Java Virtual Machine (JVM) represents data in the Unicode character code set. Unicode contains encodings for characters in most known character code sets (both single-byte and multibyte). Most components in the WebSphere business integration system are written in Java. Therefore, when data is transferred between most Server Access components, there is no need for character conversion. To log error and informational messages in the appropriate language and for the appropriate country or territory, configure the Locale standard configuration property for your environment. For more information on these properties, see Appendix A, "Standard configuration properties for connectors," on page 83

Common Event Infrastructure

This adapter is compatible with the IBMs Common Event Infrastructure, a standard for event management that permits interoperability with other IBM WebSphere event-producing applications. If Common Event Infrastructure support is enabled, events produced by the adapter can be received (or used) by another Common Event Infrastructure-compatible application.

For more information refer to the Common Event Infrastructure appendix in this guide.

Application Response Measurement

This adapter is compatible with the Application Response Measurement application programming interface (API), an API that allows applications to be managed for availability, service level agreements, and capacity planning. An ARM-instrumented application can participate in IBM Tivoli® Monitoring for Transaction Performance, allowing collection and review of data concerning transaction metrics.

For more information refer to the Application Response Measurement appendix in this guide.

Chapter 2. Installing the adapter

This chapter describes how to install the WebSphere® Business Integration Adapter for Siebel eBusiness Applications.

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

Adapter environment

Before you can install, configure, and run an adapter, you must understand its environmental requirements.

Broker compatibility

This adapter runs with the WebSphere Business Integration Adapter FrameworkV2.6 and requires one of the following:

- WebSphere InterChange ServerV4.2.2,V4.3
- WebSphere MQ IntegratorV2.1
- WebSphere MQ Integrator BrokerV2.1
- WebSphere Business Integration Message BrokerV5.0.1
- WebSphere Application Server EnterpriseV5.0.2, in conjunction with WebSphere Studio Application Developer Integration EditionV5.0.1
- WebSphere Business Integration Server FoundationV5.1.1
- DB2 Information IntegratorV8.2.3 - supported by WebSphere Business Integration Adapters for mySAP.com, PeopleSoft, and Siebel only.

Adapter platforms

In addition to a broker, this adapter requires one of the following operating systems:

- All operating system environments require the Java compiler (IBM JDK 1.4.2for Windows 2000) for compiling custom adapters
- **AIX:**
  AIX 5.1 with Maintenance Level 4
  AIX 5.2 with Maintenance Level 1. This adapter supports 32-bit JVM on a 64-bit platform.
- **Solaris:**
  Solaris 8 (2.8) with Solaris Patch Cluster dated Feb. 11, 2004 or later
  Solaris 9 (2.9) with Solaris Patch Cluster dated February 11, 2004 or later. This adapter supports 32-bit JVM on a 64-bit platform.
- **HP-UX:**
  HP-UX 11.i (11.11) with June 2003 GOLDBASE11i and June 2003 GOLDAPPS11i bundles

- **Windows:**
  Windows 2000 (Professional, Server, or Advanced Server) with Service Pack 4
  Windows XP with Service Pack 1A, for WebSphere Business Integration Adapter Framework (administrative tools only)
  Windows 2003 (Standard Edition or Enterprise Edition)

Adapter dependencies

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

- Install the Siebel 6.2.x, Siebel 7.0.x, Siebel 7.5.x, or Siebel 7.7x .jar files that will be used.
- Verify the existence of a user account in the application. This user account must be the same as the user specified in the Siebel scripts for event creation in Siebel Tools.
- Copy the Siebel `Connector.txt` file from the *%ProductDir%*`/connectors/messages/Siebel` directory to the *%ProductDir%*`/connectors/messages`directory

User setup

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

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

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

Siebel 6

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

Siebel 7.0x or 7.5x

- SiebelJI_Common.jar
- SiebelJI_enu.jar

Siebel 7.7

- Siebel.jar
- SiebelJI_enu.jar

Installing the adapter and related files

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

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

Verifying an installation

This section describes the file structures after the product has been installed on a UNIX or Windows system.

Verifying installed files on a UNIX system

To verify the installation on a UNIX system, compare the files in the directory where you installed the adapter to those listed in table 3. Table 3 describes the UNIX file structure used by the connector.

Table 3. Installed UNIX file structure for the connector

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

Verifying installed files on a Windows system

To verify the installation on a Windows system, compare the files in the directory where you installed the adapter to those listed in Table 4.Table 4 describes the Windows file structure used by the connector.

Table 4. Installed Windows file structure for the connector

Subdirectory of %*ProductDir*%	**Description**
\connectors\Siebel	Contains the connector CWSiebel.jar and the start_Siebel.bat files.
\connectors\Siebel\dependencies	Contains the patch files for event management in the Siebel eBusiness applications. This folder should also contain the Siebel .jar files.
\connectors\messages	Contains the relevant message file, SiebelConnector.txt
\connectors\Siebel\Samples\Repository	Contains the following BO samples: Siebel_BCAccount Siebel_BCQuote Siebel_BCContact Siebel_BCInternalProduct Siebel_BCAsset
\repository\Siebel\	Contains the CN_Siebel.xsd file.
\lib	Contains the WBIA. jar file.
\bin	Contains the CWConnEnv.bat file.

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

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

Event and archive tables

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

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

The following three outcomes are possible for each event processed by a connector:
- Event was processed successfully
- Event was not processed successfully
- Event was not subscribed to

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

Configuring event and archive processing

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

- The number of events for each polling interval
- Whether the connector archives unsubscribed and unprocessed events
- The unique ID of the connector, which is important when multiple connectors poll the same table

Creating the event and archive tables in Siebel, versions 7.5 and 7.7

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

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

1. Ensure that all current projects have been checked in, including:
   - Siebel Sales Enterprise project
   - Projects that include objects that you want to modify, such as the Account project

   Note: Ensure that the projects are locked on both the local and development servers.
2. Apply the six patch files in the following order to your local database:
   - `ibmtable.sif`
   - `ibmview.sif`
   - `ibmapplet.sif`
   - `ibmbo.sif`
   - `ibmbc.sif`
   - `ibmcreen.sif`

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

   Edit the first line of each file from:

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

   to:

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

   Replace all instances of the "`ENU`" language setting with "`JPN`." If you use the search and replace function of your text editor, make sure you use quotation marks around the language setting to make sure no similar words (for example, MENU) are replaced.
3. When you are prompted, lock the IBM Audit project on your local database.
4. Ensure that the following have been created:
   - Two new tables, `CX_IBM_ARCH_Q` and `CX_IBM_Event_Q`
   - One new business object, `IBM Events`
   - One new business object, `Schema version`
   - Two new business components, `IBM Archive` and `IBM Events`
   - One new view, `IBM Event List View`
   - Two new applets, `IBM Archive List Applet` and `IBM Event List Applet`
   - One new screen `IBM Events` and one new screen view, `IBM Event List` view
5. Create a page tab as follows:
   a. Access the Application > Siebel Sales Enterprise > Page tab.

   b. Right-click and select New Record from the menu.
   c. Enter `IBM Events` as the screen name and `IBM Events` as the text name.
   d. For the sequence, enter a number greater than the rest of the sequence numbers. This selection determines where the tab is displayed in the application.
   e. Leave the inactive field unchecked.
   f. Go to the Page tab locale and create a new record for `IBM Events`. Add `ENU` for the Language Code and `IBMEvents` for text, if it does not exists.
6. Create a screen menu item as follows:
   a. Access the Application > Siebel Sales Enterprise > Screen Menu Item.
   b. Right-click and select New Record.
   c. Enter `IBM Events` as the screen and `IBM Events` as the text name.
   d. For the sequence, enter a number greater than the rest of the sequence numbers. This selection determines where the tab is displayed in the screen pull-down menu.
   e. Leave the inactive field unchecked.
   f. Go to the screen menu item locale and create a new record for `IBMEvents`. Add `ENU` for language and `IBMEvents` for text, if it does not exist.
7. Add or modify the Siebel VB scripts for the business components that correspond to the business objects used at your site. The Siebel VB scripts trigger event notification for business objects.
   - If you want to sort events by priority, edit the priority values in the business objects VB scripts before compiling them.
   - If you are installing multiple connectors, set and activate the Connector Id in the VB scripts.
8. Apply the physical schema for the new tables to your local database. You can do this by querying for the two new tables, `CX_IBM_ARCH_Q` and `CX_IBM_EVENT_Q`, and selecting the current query to create a physical schema. Make sure that you leave the table space and index space blank.
9. Activate the new schema using the activate button.
10. Compile the updated and locked projects on your local database to create a new Siebel repository (`.srf`) file.
11. Open Siebel Sales Enterprise on your local database. You must have administrative privileges to perform the following:
   a. .Create a new view called `IBM Event List View`. Tip: Copy the view name from tools and paste it into the View Name field.
   b. .Create a new responsibility called `IBM Responsibility` for `IBM Event List View`.
   c. .Add the employees or teams who are responsible for reviewing events to the newly created `IBM Responsibility`.
   d. .Create the `CWCONN` user and add it to `IBM Responsibility` and `Administrative Responsibility`.
12. Test the application in your local environment. Ensure that you have visibility to `IBM Event List View` and that an event is generated in the view after you create a supported object. For example, create a new account in Siebel and check that a new account event appears in the `IBM Event List View`.
13. Check in the following updated and locked projects to your development server.
   - IBM Audit

- Siebel Sales Enterprise
- The project for the business objects that you want to use

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

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

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

To set Siebel JAVABean:

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

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

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

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

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

1. Ensure that all current projects have been checked in.
2. On your local database, check out and lock the following files:
   - New Table Project
   - Siebel Sales Enterprise project
   - Projects that include objects that you want to modify, such as the Account project
   - Dock project

   Note: Ensure that the projects are locked on both the local and development servers.
3. Apply the seven patch files in the following order to your local database:
   - `cwtable.sif`
   - `cwview.sif`
   - `cwapplet.sif`
   - `cwbo.sif`
   - `cwbc.sif`
   - `cwdo.sif`
   - `cwscreen.sif`
   - `schemabo.sif`

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

Edit the first line of each file from:

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

to:

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

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

4. When you are prompted, lock the CW Audit project on your local database.
5. Ensure that the following have been created:
   - Two new tables, `CX_CW_Archive_Q` and `CX_CW_Event_Q`
   - One new business object, `Events`
   - One new business object, `schema version`
   - Two new business components, `Archive` and `Events`
   - One new view, `Event List View`
   - Two new applets, `Archive List Applet` and `Event List Applet`
   - One new screen `Events` and one new screen view, `Event List` view
   - Two new dock objects, `CX_CWArchive` and `CX_CWEvent`
6. Create a page tab as follows:
   a. Access the Application > Siebel Sales Enterprise > Page tab.
   b. Right-click and select New Record from the menu.
   c. Enter `CW Events` as the screen name and `IBM Events` as the text name.
   d. For the sequence, enter a number greater than the rest of the sequence numbers. This selection determines where the tab is displayed in the application.
   e. Leave the inactive field unchecked.
   - If you are using Siebel 6, proceed to Step 7.
   - If you are using Siebel 7, go to the Page tab locale and create a new record for `CW Events`. Add `ENU` for the Language Code and `IBMEvents` for text, if it does not exists.
7. Create a screen menu item as follows:
   a. Access the Application > Siebel Sales Enterprise > Screen Menu Item.
   b. Right-click and select New Record.
   c. Enter `Events` as the screen and `IBM Events` as the text name.
   d. For the sequence, enter a number greater than the rest of the sequence numbers. This selection determines where the tab is displayed in the screen pull-down menu.
   e. Leave the inactive field unchecked.
   - If you are using Siebel 6, proceed to Step 8.
   - If you are using Siebel 7, go to the screen menu item locale and create a new record for `CWEvents`. Add `ENU` for language and `IBMEvents` for text, if it does not exist.
8. Add or modify the Siebel VB scripts for the business components that correspond to the business objects used at your site. The Siebel VB scripts trigger event notification for business objects.

- If you want to sort events by priority, edit the priority values in the business objects VB scripts before compiling them.
- If you are installing is multiple connectors, set and activate the Connector Id in the VB scripts.

  Siebel 6

  If you want to use the `Additional Object Key` field, you must set it in the VB script.

9. Apply the physical schema for the new tables to your local database. You can do this by querying for the two new tables, `CX_CW_ARCHIVE_Q` and `CX_CW_EVENT_Q`, and selecting the current query to create a physical schema. Make sure that you leave the table space and index space blank.
10. Activate the new schema using the activate button.
11. Compile the updated and locked projects on your local database to create a new Siebel repository (`.srf`) file.
12. Open Siebel Sales Enterprise on your local database. You must have administrative privileges to perform the following:
    a. .Create a new view called `Event List View`. Tip: Copy the view name from tools and paste it into the View Name field.
    b. .Create a new responsibility called `CW Responsibility` for `Event List View`.
    c. .Add the employees or teams who are responsible for reviewing events to the newly created `CW Responsibility`.
    d. .Create the `CWCONN` user and add it to `CW Responsibility` and `Administrative Responsibility`.
13. Test the application in your local environment. Ensure that you have visibility to `Event List View` and that an event is generated in the view after you create a supported object. For example, create a new account in Siebel and check that a new account event appears in the `Event List View`.
14. Check in the following updated and locked projects to your development server.
    - New Table
    - CW Audit
    - Dock
    - Siebel Sales Enterprise
    - The project for the business objects that you want to use

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

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

To set Siebel JAVABean:

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

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

Chapter 3. Configuring the connector

This chapter describes how to install and configure the adapter using Connector Configurator.

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)

If your adapter supports DB2 Information Integrator, use the WMQI options and the DB2 II standard properties (see the Notes column in the Standard Properties appendix.)

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 20).

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 21 to set up a new one.

Running connectors on UNIX

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.

Some properties in the Connector Configurator use directory paths, which default to the Windows convention for directory paths. If you use the configuration file in a UNIX environment, revise the directory paths to match the UNIX convention for these paths. Select the target operating system in the toolbar drop-list so that the correct operating system rules are used for extended validation.

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 Business Integration Adapters>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 26.)

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 21.
- 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
  Property Subtype
  Updated Method
  Description
- **Flags**
  Standard flags
- **Custom Flag**
  Flag

The **Property Subtype** can be selected when **Property Type** is a String. It is an optional value which provides syntax checking when you save the configuration file. The default is a blank space, and means that the property has not been subtyped.

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. Right-click on the square to the left of the Value column heading.
2. From the pop-up menu, select **Add** to display the Property Value dialog box. Depending on the property type, the dialog box allows you to enter either a value, or both a value and a range.
3. Enter 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.

Setting pathnames

Some general rules for setting pathnames are:

- The maximum length of a filename in Windows and UNIX is 255 characters.
- In Windows, the absolute pathname must follow the format [Drive:][Directory]\filename: for example, C:\WebSphereAdapters\bin\Data\Std\StdConnProps.xml
  In UNIX the first character should be /.

- Queue names may not have leading or embedded spaces.

Creating a new configuration file

When you create a new configuration file, you must name it and select an integration broker.

You also select an operating system for extended validation on the file. The toolbar has a droplist called **Target System** that allows you to select the target operating system for extended validation of the properties. The available options are: Windows, UNIX, Other (if not Windows or UNIX), and None-no extended validation (switches off extended validation). The default on startup is Windows.

To start Connector Configurator:

- In the System Manager window, select **Connector Configurator** from the **Tools** menu. Connector Configurator opens.
- In stand-alone mode, launch Connector Configurator.

To set the operating system for extended validation of the configuration file:

- Pull down the **Target System:** droplist on the menu bar.
- Select the operating system you are running on.

Then 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.
- Complete 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. Set the operating system for extended validation of the configuration file using the **Target System:** droplist on the menu bar (see "Creating a new configuration file" above).
2. Click **File>New>Connector Configuration**.
3. 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**.

4. 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 complete the definition now, or you can save the file and complete the fields later.
5. 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.
6. 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 connector properties associated with the selected broker. The table shows **Property name**, **Value**, **Type**, **Subtype** (if the Type is a string), **Description**, and **Update Method**.
3. You can save the file now or complete the remaining configuration fields, as described in "Specifying supported business object definitions" on page 29..
4. 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 you created the configuration file, you used the **Target System** droplist that allows you to select the target operating system for extended validation of the properties.

   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.

   If you have elected to use the extended validation feature by selecting a value of Windows, UNIX or Other from the **Target System** droplist, the system will validate the property subtype s well as the type, and it displays a warning message if the validation fails.

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)
- Security

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 the Standard Properties appendix. 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.

   Note: If the property has a Type of String, it may have a subtype value in the Subtype column. This subtype is used for extended validation of the property.
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.

To get more information on a particular standard property, left-click the entry in the Description column for that property in the Standard Properties tabbed sheet. If you have Extended Help installed, an arrow button will appear on the right. When you click on the button, a Help window will open and display details of the standard property.

Note: If the hot button does not appear, no Extended Help was found for that property.

If installed, the Extended Help files are located in <*ProductDir*>\bin\Data\Std\Help\<*RegionalSetting*>\.

Setting connector-specific configuration properties

For connector-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.

   Note: If the property has a Type of String, you can select a subtype from the Subtype droplist. This subtype is used for extended validation of the property.
3. To encrypt a property, select the **Encrypt** box.

4. To get more information on a particular property, left-click the entry in the Description column for that property. If you have Extended Help installed, a hot button will appear. When you click on the hot button, a Help window will open and display details of the standard property.

   Note: If the hot button does not appear, no Extended Help was found for that property.
5. Choose to save or discard changes, as described for "Setting standard connector properties" on page 28.

If the Extended Help files are installed and the AdapterHelpName property is blank, Connector Configurator will point to the adapter-specific Extended Help files located in <*ProductDir*>\bin\Data\App\Help\<*RegionalSetting*>\. Otherwise, Connector Configurator will point to the adapter-specific Extended Help files located in <*ProductDir*>\bin\Data\App\Help\<*AdapterHelpName*>\<*RegionalSetting*>\. See the AdapterHelpName property described in the Standard Properties appendix.

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 Properties appendix.

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 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)

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 Binding**

  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** tab enables you to configure messaging properties. 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.

Validating messaging queues

Before you can validate a messaging queue, you must:

- Make sure that WebSphere MQ Series is installed.
- Create a messaging queue with channel and port on the host machine.
- Set up a connection to the host machine.

To validate the queue, use the Validate button to the right of the Messaging Type and Host Name fields on the Messaging tab.

Security (ICS)

You can use the **Security** tab in Connector Configurator to set various privacy levels for a message. You can only use this feature when the DeliveryTransport property is set to JMS.

By default, Privacy is turned off. Check the **Privacy** box to enable it.

The **Keystore Target System Absolute Pathname** is:

- For Windows:
  `<ProductDir>\connectors\security\<connectorname>.jks`
- For UNIX:
  `opt/IBM/WebSphereAdapters/connectors/security/<connectorname>.jks`

This path and file should be on the system where you plan to start the connector, that is, the target system.

You can use the Browse button at the right only if the target system is the one currently running. It is greyed out unless **Privacy** is enabled and the **Target System** in the menu bar is set to Windows.

The **Message Privacy Level** may be set as follows for the three messages categories (All Messages, All Administrative Messages, and All Business Object Messages):

- "" is the default; used when no privacy levels for a message category have been set.
- none
  Not the same as the default: use this to deliberately set a privacy level of none for a message category.
- integrity
- privacy
- integrity_plus_privacy

The **Key Maintenance** feature lets you generate, import and export public keys for the server and adapter.

- When you select **Generate Keys**, the Generate Keys dialog box appears with the defaults for the keytool that will generate the keys.
- The keystore value defaults to the value you entered in **Keystore Target System Absolute Pathname** on the Security tab.
- When you select OK, the entries are validated, the key certificate is generated and the output is sent to the Connector Configurator log window.

Before you can import a certificate into the adapter keystore, you must export it from the server keystore. When you select **Export Adapter Public Key**, the Export Adapter Public Key dialog box appears.

- The export certificate defaults to the same value as the keystore, except that the file extension is <filename>.cer.

When you select **Import Server Public Key**, the Import Server Public Key dialog box appears.

- The import certificate defaults to <*ProductDir*>\bin\ics.cer (if the file exists on the system).
- The import Certificate Association should be the server name. If a server is registered, you can select it from the droplist.

The **Adapter Access Control** feature is enabled only when the value of DeliveryTransport is IDL. By default, the adapter logs in with the guest identity. If the **Use guest identity** box is not checked, the **Adapter Identity** and **Adapter Password** fields are enabled.

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 in the properties window 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>
```

Chapter 4. Understanding business objects

This chapter describes how the connector processes business objects. To properly create or modify business objects for Siebel, you must understand the object relationships within the Siebel architecture.

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

Business object structure and relationships

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

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

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

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

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

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

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

Specifying key attributes

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

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

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

Attribute properties

The following tables describe simple attributes and child object attributes.

Table 5. Simple attribute

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

Table 6. Child object attributes

Name	Name of the child object
`Type`	Business object type for the child.
`ContainedObjectVersion`	The child business object version

Table 6. Child object attributes (continued)

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

Business object application-specific information

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

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

Business object application-specific information

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

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

Example of multiple unrelated business components:

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

[BusinessObjectDefinition]
Name = Siebel_BCProductDefect
Version = 1.0.0
AppSpecificInfo = CN=ProductDefect

[BusinessObjectDefinition]
```

```
Name = Siebel_BOInternalProduct
Version = 1.0.0
AppSpecificInfo = ON=InternalProduct

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

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

Attribute application-specific information format

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

Application-specific information for simple attributes

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

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

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

Field names for a simple attributes

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

```
FN=fieldname
```

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

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

Foreign key relationship using a pickList

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The AppSpecificInfo for restricting PickList fields follows this syntax:

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

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

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

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

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

Name = Account
```

```
Type = String
AppSpecificInfo = FN=AccountId

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

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

Note the following business processing tips for PickList attributes:

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

Application-specific information for container attributes

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

Parameter	Description
`LFN = ...;`	Multi Value Field Name related to the Siebel business component.
`MVL = ...;`	When `MVL` is set to `Active`, it specifies a one-to-many relationship. Setting MVL to Inactive indicates that there is an inactive multi-value link relationship between the parent and child objects, which means that the parent object does not have a multi-value field.
`PL = ...`	When `PL` is set to `True`, it indicates that there is a many-to-one relationship.
`Assoc = ...`	When `Assoc` is set to `True`, it indicates that a relationship is many-to-many through an intersection table. When the relationship is an association between the two business components, an active multi-value link may or may not exist. Based on whether or not the multi-value link exists, the application specific information can contain: `MVL=Active;Assoc=true;` If the application specific information contains only `Assoc=true`, the default is that there is an active multi-value link. If you explicitly specify, the application-specific information will contain: `MVL=Active;Assoc=true`. If a multi value link does not exist, the application specific information will contain: `MVL=Inactive;Assoc=true`.

Parameter	Description
From = ...; To = ...	These are preprocessing instructions to the connector to set the To attribute to the value of the From attribute. The From attribute must be populated, while the True attribute is set only if it is null. The objects containing the attributes must have a one-to-one relationship. This is used in a Retrieve operation and to specify which child record needs to be fetched.
SF	Simple link which provides the master detail view of the business component structure under a business object. `SF` represents the source field.
DF	Simple link which provides the master detail view of the business component structure under a business object. `DF` represents the destination field (foreign key).

Field names for container attributes

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

```
LFN=multiValueFieldName
```

For example, if the Siebel business component Account has a mulit-value field Street Address in the Siebel application, the corresponding WebSphere business integration system business object `Siebel_BCAccount` has a container attribute for the child business object `Siebel_BCBusinessAddress`. The application-specific information for this container attribute specifies Street Address as the Multi-Value Field that contains the link to the Siebel Business Address business component.

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

Relationships between parent and child business objects

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

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

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

```
[Example of Container Attribute]
Name = PrimaryAddress
Type = Siebel_BCBusinessAddress
ContainedObjectVersion = 1.0.0
Relationship = Containment
```

```
Cardinality = n
IsKey = false
IsForeignKey = false
AppSpecificInfo = MVL=Active;LFN=Street Address
```

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

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

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

Name = ...

Name = ...

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

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

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

Name = ...

Name = ...

Name = Siebel_BCOrderEntry
Type = Siebel_BCOrderEntry
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = N
MaxLength = 0
```

```
IsKey = true
IsFireignKey = false
IsRequired - false
AppspecificInfo = SF=QuoteId;DF=QuoteId
```

Assigning the value of an attribute to another attribute

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

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

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

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

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

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

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

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

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

Specifying pickList relationships

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

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

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

Name = ...

Name = ...

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

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

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

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

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

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

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

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

Verb application specific information format

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

```
[Verb] Name = Retrieve AppSpecificInfo = max=5
```

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

Multiple record retrieval

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

The adapter for Siebel supports wrapper business objects. See“Business object structure and relationships” on page 37. The adapter can retirieve multiple records during request processing using the `RetrievebyContent` verb and a wrapper business object. For a wrapper business object with a multiple cardinality container, all matching records are returned. For a single cradinality child, only one record is processed and MULTIPLE_HITS is returned.

Key attribute for create and update verbs

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

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

Chapter 5. Creating business objects

Modifying business object samples

Business object samples are provided with the connector component of the adapter. All objects must have Siebel triggers on them for polling. In some cases, you may need to customize the object in Siebel Tools. This section describes sample objects and provides examples of how to customize them.

- “Siebel_BCAccount” on page 49
- “Siebel_BCQuote” on page 50
- “Siebel_BCContact” on page 50
- “Siebel_BCInternalProduct” on page 51
- “Siebel_BCAsset” on page 51

Siebel_BCAccount

1. Locate the following Siebel VB files:

   Siebel 6

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

   Siebel 7

   `Account.sbl or Account.js`

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

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

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

Siebel_BCQuote

1. Locate the following Siebel VB files:

   Siebel 6

   `Quote_Write.svb`, `Contact_PreDelete.svb`

   Siebel 7

   `Quote.sbl`, `Quote.js`

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

Siebel_BCContact

1. Locate the following Siebel VB files:

   Siebel 6

   `Contact_Write.svb`, `Contact_PreDelete.svb.`

   Siebel 7

   `Contact.sbl`, `Contact.js`

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

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

Siebel_BCInternalProduct

1. Locate the file `InternalProduct_Write.svb`. The Siebel VB files are located in the `Common/Siebel/Dependencies/Siebel_VB` directory.
2. Save the file in a temporary directory. For example, use Temp as the directory name.
3. Give the CWCONN account the System Administrator responsibility within Siebel.
4. In your Siebel Tools environment, check out and lock the Product project.
5. Add the Siebel VB script to the Internal Product business component.
   a. Right-click on the Internal Product business component, and select Edit Basic Scripts from the menu. This launches the script editor.
   b. Import the VB code by selecting Import from the File menu, choosing the Temp directory, and picking `InternalProduct_Write.svb` for import into the Bus Comp object and BusComp_WriteRecord procedure.
   c. Save changes to the object.
6. Change the business component properties as follows:
   a. In the tool bar, select View > Property Window.
   b. Go to Business Component/Internal Product.
   c. Change the following attributes to False:
      ```
      No Insert = False
      No Merge = False
      No Update = False
      ```

      Note: The purpose of changing the above properties is to allow Com Data Server Interface to create and update products inbound to Siebel.
7. You will not be able to test the object until you have saved all objects and compiled the result.

Siebel_BCAsset

1. Locate the Siebel VB file `Asset_Write.svb`. The Siebel VB files are located in the `Common/Siebel/Dependencies/Siebel_VB` directory.

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

      Note: WebSphere business integration components track the Products by their Product Id and not their Name.
7. You will not be able to test the object until you have saved all objects and compiled the result.

Overview of Siebel ODA

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

Installing and using Siebel ODA

This section discusses the following:
- "Installing SiebelODA" on page 52
- "Before using SiebelODA" on page 53
- "Launching the SiebelODA" on page 54
- "Running SiebelODA on multiple machines" on page 55
- "Working with error and trace message files" on page 55

Installing SiebelODA

To install SiebelODA, use the WebSphere Business Integration Adapter Installer. Follow the instructions in the *System Installation Guide for UNIX* or *for Windows*.

When the installation is complete, the following files are installed in the directory on your system where you have installed the product:

- `ODA\Siebel\SiebelODA.jar`
- `ODA\messages\SiebelODAAgent.xsd`
- `ODA\Siebel\start_SiebelODA.bat` (Windows only)
- `ODA/Siebel/start_SiebelODA.sh` (UNIX only)
- `bin\CWODAEnv.bat` (Windows only)
- `bin/CWODAEnv.sh` (UNIX only)

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

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

Before using SiebelODA

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

Siebel 7.7

```
SiebelJI_enu.jar
Siebel.jar
```

Siebel 7.x

```
SiebelJI_enu.jar
SiebelJI_Common.jar
```

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

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

Siebel 6.2.x

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

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

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

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

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

The following sections describe these steps in detail.

Understanding datatype mapping

The following table lists the Siebel application datatypes and their corresponding WBI business object definition datatypes. These are used by the WBI Adapter for Siebel ODA only; the application regards all attribute values as strings.

Table 7. Siebel application and WBI Adapter datatypes

Siebel datatype	WBI Adapter for Siebel datatype
`DTYPE_BOOL`	Boolean
`DTYPE_ID`	String
`DTYPE_PHONE`	String
`DTYPE_TEXT`	String
`DTYPE_NOTE`	String
`DTYPE_DATE`	Date
`DTYPE_TIME`	Date
`DTYPE_DATETIME`	Date
`DTYPE_UTCDATETIME`	Date
`DTYPE_INTEGER`	Integer
`DTYPE_NUMBER`	Float
`DTYPE_CURRENCY`	Double
The following datatype mappings are used when you create business object definitions using the Business Services option.	
Hierarchy	Container or String, depending on the populated content, if any, of the Integration Object field.
Integration object	Container or String, depending on the populated content, if any, of the Integration Object field.
Number	Integer
String	String
Date	Date

Launching the SiebelODA

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

UNIX

```
start_SiebelODA.sh
```

Windows

```
start_SiebelODA.bat
```

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

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

Running SiebelODA on multiple machines

You can run multiple instances of the ODA, either on the local host or a remote host in the network. Each instance runs on a unique port.

Working with error and trace message files

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

`AgentName``Agent.txt`

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

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

Important: Failing to correctly specify the message file's name when you configure the ODA causes it to run without messages. For more information on specifying the message file name, see Table 9 on page 58.

During the configuration process, you specify:

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

Table 8 describes these values.

Table 8. Tracing levels

Trace Level	Description
0	Logs all errors
1	Traces all entering and exiting messages for method
2	Traces the ODA's properties and their values
3	Traces the names of all business objects
4	Traces details of all spawned threads

Table 8. Tracing levels (continued)

5	• Indicates the ODA initialization values for all of its properties • Traces a detailed status of each thread that SiebelODA spawned • Traces the business object definition dump

For information on where you configure these values, see Table 9 on page 58.

Using SiebelODA in Business Object Designer

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

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

1. Open Business Object Designer.
2. From the File menu, select the New Using ODA... submenu.

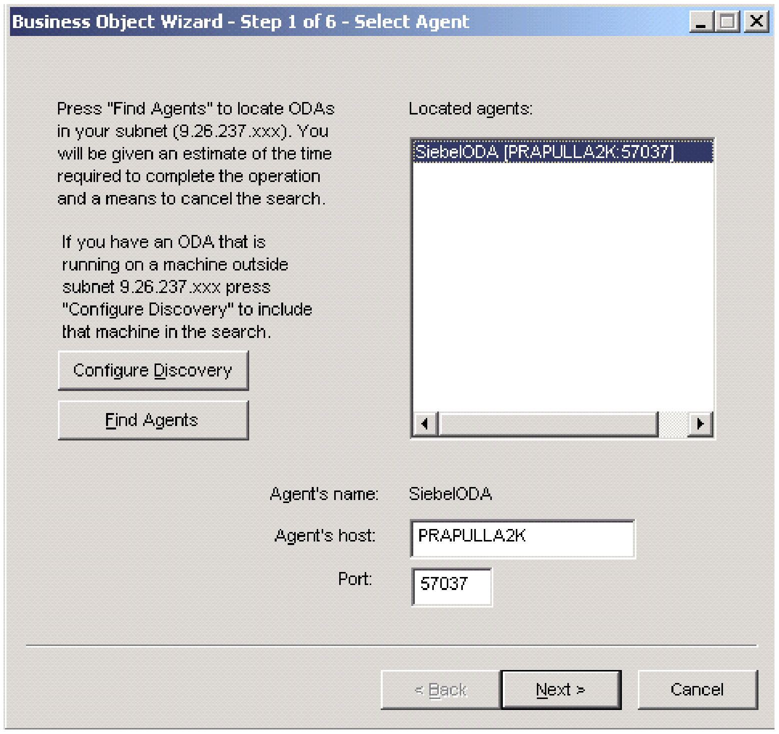
   Business Object Wizard displays the first window in the wizard, named Select Agent. Figure 2 on page 57 illustrates this window.

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

1. "Select the ODA"
2. "Specify configuration properties" on page 57
3. "Selecting the source" on page 59
4. "Confirm selection of objects" on page 63
5. "Generate the business object definition" on page 64 and, optionally, "Provide additional information" on page 65
6. "Save the business object definition" on page 68

Select the ODA

Figure 2 on page 57 illustrates the first dialog box in Business Object Wizard's six-step wizard. From this window, select the ODA to run.


Business Object Wizard - Step 1 of 6 - Select Agent

Press "Find Agents" to locate ODAs in your subnet (9.26.237.xxx). You will be given an estimate of the time required to complete the operation and a means to cancel the search.

If you have an ODA that is running on a machine outside subnet 9.26.237.xxx press "Configure Discovery" to include that machine in the search.

Located agents:
SiebelODA [PRAPULLA2K:57037]

Agent's name: SiebelODA
Agent's host: PRAPULLA2K
Port: 57037

Figure 2. Business Object Wizard, Select ODA screen

To select the ODA:

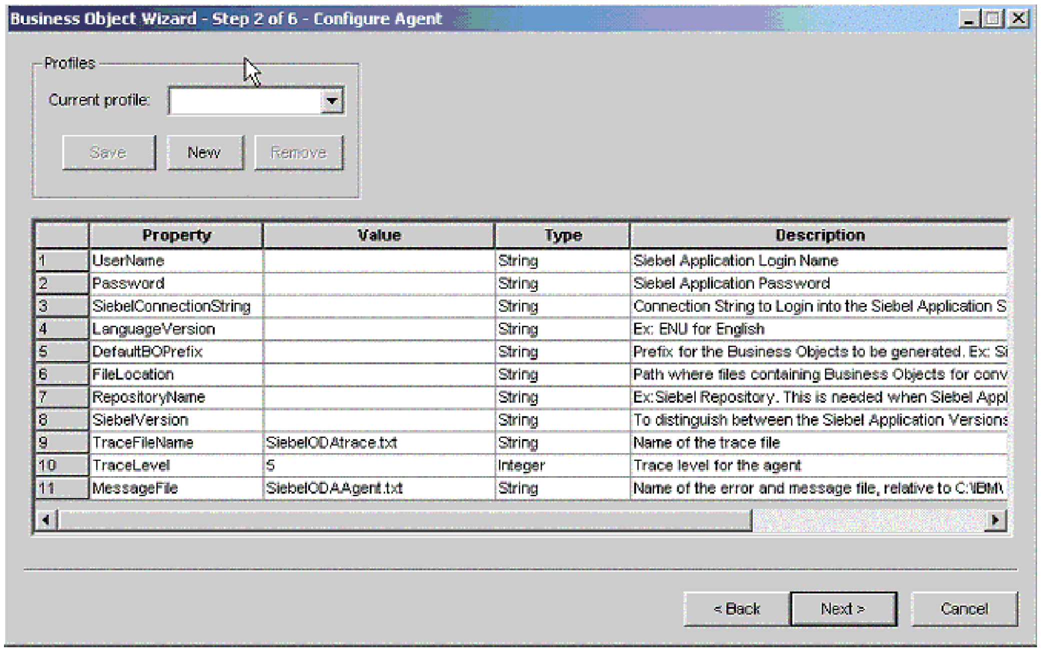
1. Click the Find Agents button to display all registered or currently running ODAs in the Located agents field. Alternatively, you can find the ODA using its host name and port number.

   Note: If Business Object Wizard does not locate your desired ODA, check the setup of the ODA.
2. Select the desired ODA from the displayed list.

   Business Object Wizard displays your selection in the Agent's name field.
3. Click Next.

Specify configuration properties

The first time Business Object Wizard communicates with SiebelODA, it prompts you to enter a set of ODA configuration properties as shown in Figure 4.



	Property	Value	Type	Description
1	UserName		String	Siebel Application Login Name
2	Password		String	Siebel Application Password
3	SiebelConnectionString		String	Connection String to Login into the Siebel Application S
4	LanguageVersion		String	Ex: ENU for English
5	DefaultBOPrefix		String	Prefix for the Business Objects to be generated. Ex: Si
6	FileLocation		String	Path where files containing Business Objects for conv
7	RepositoryName		String	Ex:Siebel Repository. This is needed when Siebel Appl
8	SiebelVersion		String	To distinguish between the Siebel Application Versions
9	TraceFileName	SiebelODAtrace.txt	String	Name of the trace file
10	TraceLevel	5	Integer	Trace level for the agent
11	MessageFile	SiebelODAAgent.txt	String	Name of the error and message file, relative to C:\IBM\

Figure 3. Business Object Wizard, Configure Agent screen

Configure the SiebelODA properties described in Table 9.

Table 9. SiebelODA configuration properties

Row number	Property name	Property type	Description
1	`UserName`	String	Siebel application login name
2	`Password`	String	Siebel application password

3	`SiebelConnection String`	String	Connect string to log into the Siebel application.
			
			Examples:
			
			For Siebel 7.x:
			`//machinename/enterprisename/objectmanager /servername`
			
			For Siebel 7.5:
			`//machinename/enterprisename/objectmanager /servername`
			
			For Siebel 7.7:
			`siebel://machinename:portno/enterprisename /objectmanager`

4	`Language version`	String	Language version. For example, use `ENU` for English.
5	`DefaultBOPrefix`	String	Prefix that the ODA applies to the name of each business object definition for the Siebel document. If you do not specify a business-object prefix, the ODA does *not* prepend any string to the name of the business object definition.
6	`FileLocation`	String	The absolute path containing the files with previous versions of business object definitions. For example, in UNIX, the path is `/home/SiebelBos`, and in Windows, the path is `C:\SiebelBos`.
7	`RepositoryName`	String	The name of the Siebel repository in the Siebel application.
8	`SiebelVersion`	String	Identifies the Siebel Application version. For Siebel version 6.x, this property must be set to 6.x. For Siebel version 7.x, do not set.

Table 9. SiebelODA configuration properties (continued)

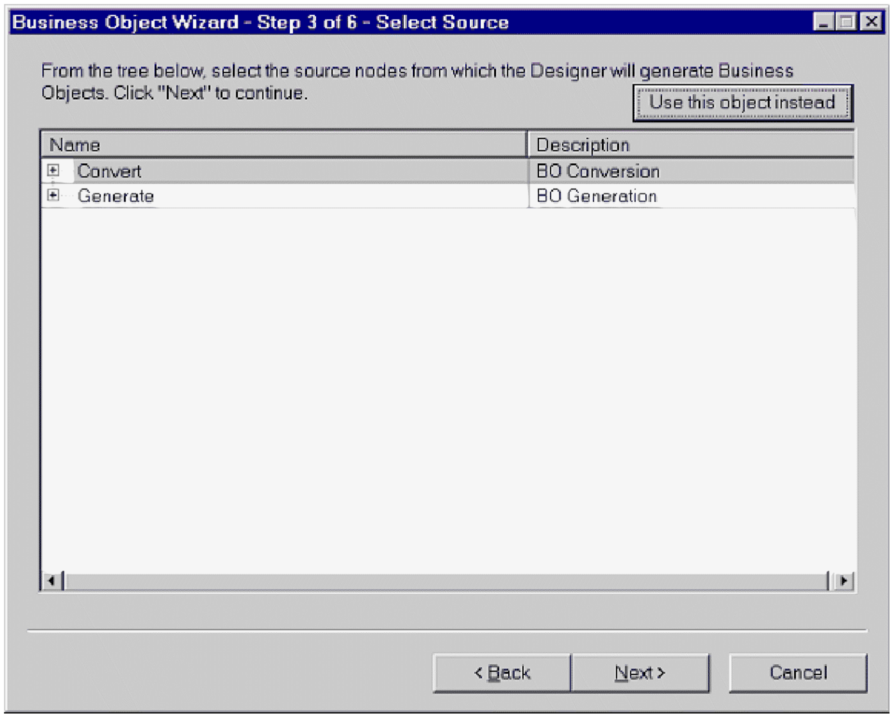
Row number	Property name	Property type	Description
9	`TraceFileName`	String	Full pathname of the file into which SiebelODA writes trace information. If the file does not exist, SiebelODA creates it in the specified directory. If the file already exists, SiebelODA appends to it. By default, SiebelODA creates a trace file named `SiebelODAtrace.txt` in the `ODA\Siebel` subdirectory of the product directory. Use this property to specify a different name for the trace file.
10	`TraceLevel`	Integer	Level of tracing enabled for SiebelODA. Valid values are zero through five (0-5). Property defaults to a value of 5 (full tracing enabled). For more information, see “Working with error and trace message files” on page 55.
11	`MessageFile`	String	Full pathname of the error and message file. By default, SiebelODA creates a message and error file named `SiebelODAAgent.txt`. **Important**: The error and message file *must* be located in the `ODA\messages` subdirectory of the product directory. Use this property to verify or specify an existing file.

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

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

Selecting the source

After you configure all initialization properties for SiebelODA, the Select Source screen appears (see Figure 4 on page 60).

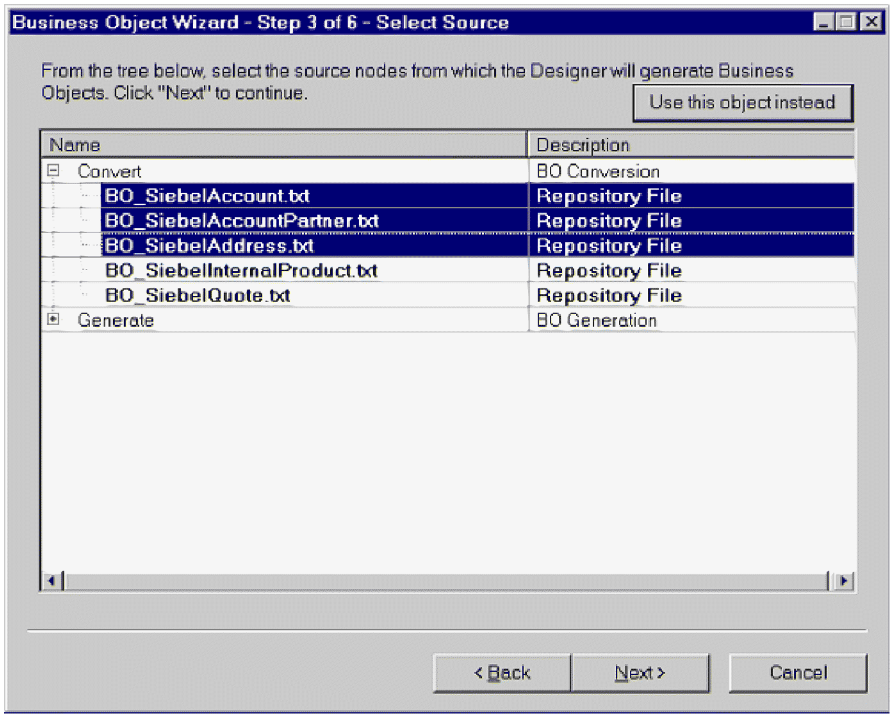

Business Object Wizard - Step 3 of 6 - Select Source

From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

Name	Description
Convert	BO Conversion
Generate	BO Generation

Figure 4. Business Object Wizard, Select Source screen

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


Business Object Wizard - Step 3 of 6 - Select Source

From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

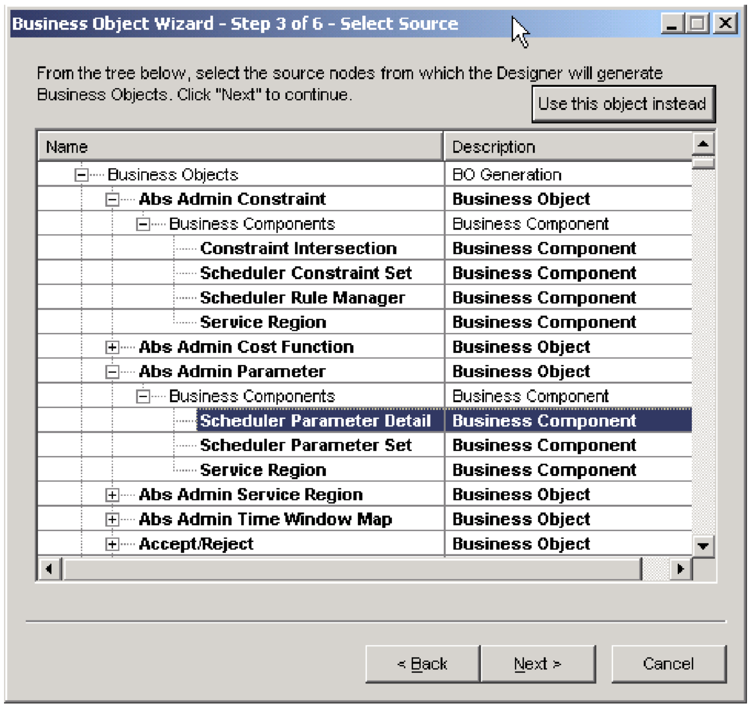
Name	Description
Convert	BO Conversion
BO_SiebelAccount.txt	Repository File
BO_SiebelAccountPartner.txt	Repository File
BO_SiebelAddress.txt	Repository File
BO_SiebelInternalProduct.txt	Repository File
BO_SiebelQuote.txt	Repository File
Generate	BO Generation

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

If you need to generate new business objects, expand Generate. From there, you have three expandable options: Business objects, Integration objects, and Application services. For examples of these options expanded, see Figure 6, Figure 7 on page 62, and Figure 8 on page 62. When you expand a business object, you can select a business component for that object. Similarly, when you expand an integration object, you can select an integration component for that object. When you expand an application service, however, a corresponding integration object is already selected.

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

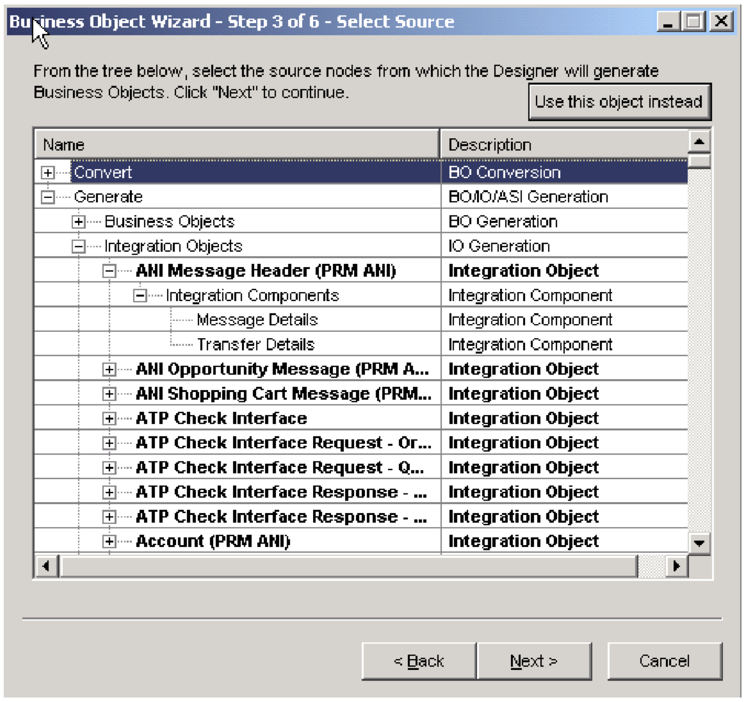
Note: When you generate an integration objects, all of the components listed for that object are generated.



From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

Name	Description
Business Objects	BO Generation
Abs Admin Constraint	Business Object
Business Components	Business Component
Constraint Intersection	Business Component
Scheduler Constraint Set	Business Component
Scheduler Rule Manager	Business Component
Service Region	Business Component
Abs Admin Cost Function	Business Object
Abs Admin Parameter	Business Object
Business Components	Business Component
Scheduler Parameter Detail	Business Component
Scheduler Parameter Set	Business Component
Service Region	Business Component
Abs Admin Service Region	Business Object
Abs Admin Time Window Map	Business Object
Accept/Reject	Business Object

Figure 6. Business Object Wizard, displaying Business Objects expanded

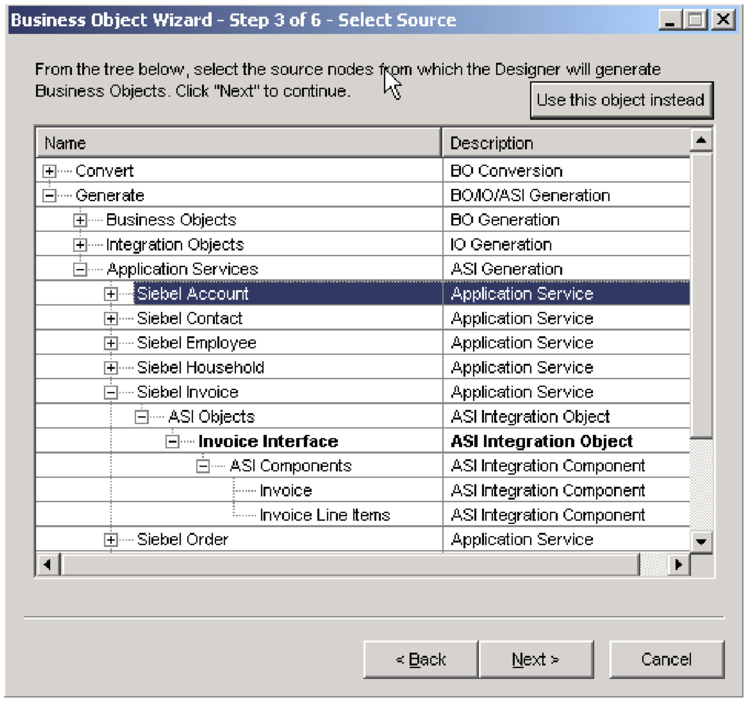


Business Object Wizard - Step 3 of 6 - Select Source

From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

Name	Description
Convert	BO Conversion
Generate	BO/IO/ASI Generation
Business Objects	BO Generation
Integration Objects	IO Generation
ANI Message Header (PRM ANI)	**Integration Object**
Integration Components	Integration Component
Message Details	Integration Component
Transfer Details	Integration Component
ANI Opportunity Message (PRM A...	**Integration Object**
ANI Shopping Cart Message (PRM...	**Integration Object**
ATP Check Interface	**Integration Object**
ATP Check Interface Request - Or...	**Integration Object**
ATP Check Interface Request - Q...	**Integration Object**
ATP Check Interface Response - ...	**Integration Object**
ATP Check Interface Response - ...	**Integration Object**
Account (PRM ANI)	**Integration Object**

Figure 7. Business Object Wizard, displaying Integration Objects expanded

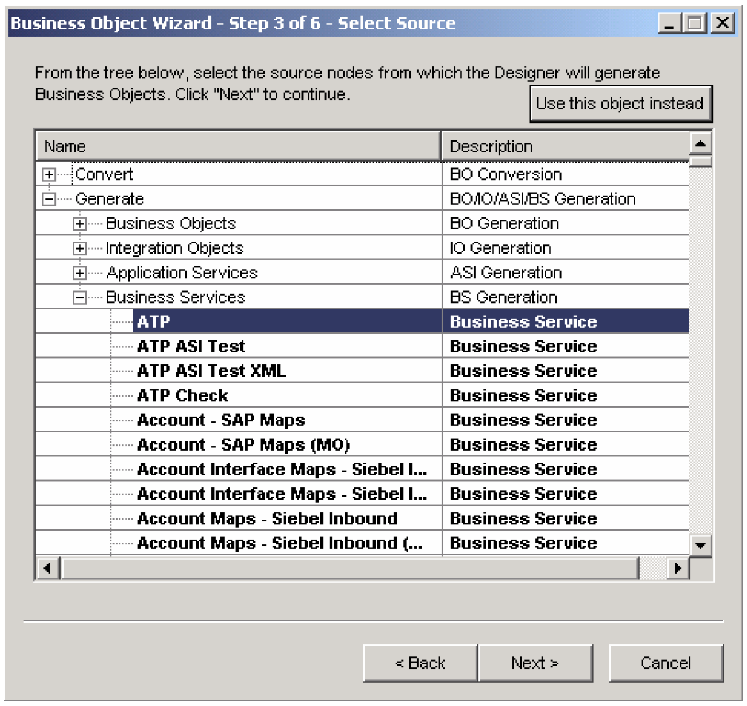


Business Object Wizard - Step 3 of 6 - Select Source

From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

Name	Description
Convert	BO Conversion
Generate	BO/IO/ASI Generation
Business Objects	BO Generation
Integration Objects	IO Generation
Application Services	ASI Generation
Siebel Account	Application Service
Siebel Contact	Application Service
Siebel Employee	Application Service
Siebel Household	Application Service
Siebel Invoice	Application Service
ASI Objects	ASI Integration Object
Invoice Interface	**ASI Integration Object**
ASI Components	ASI Integration Component
Invoice	ASI Integration Component
Invoice Line Items	ASI Integration Component
Siebel Order	Application Service

Figure 8. Business Object Wizard, displaying Application Services expanded


Business Object Wizard - Step 3 of 6 - Select Source

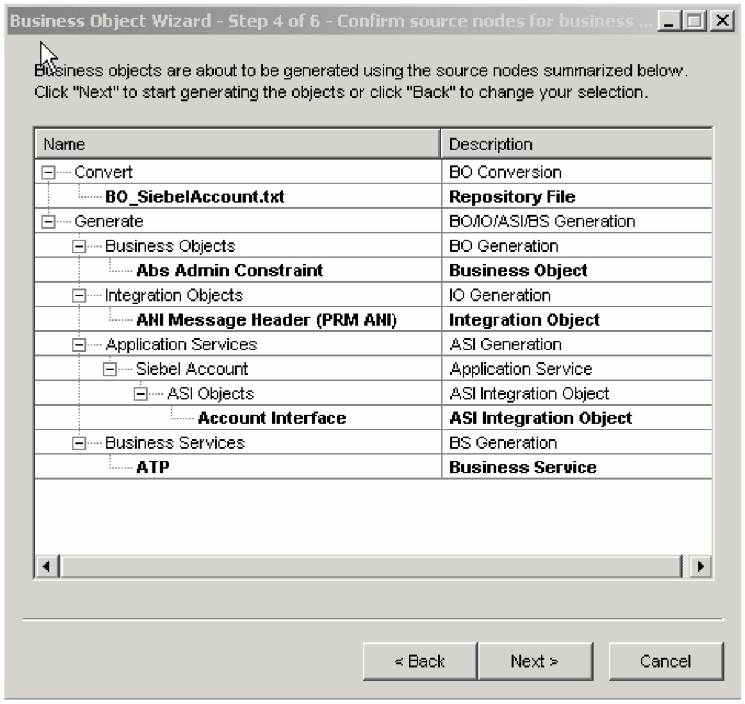
From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

Name	Description
Convert	BO Conversion
Generate	BO/IO/ASI/BS Generation
Business Objects	BO Generation
Integration Objects	IO Generation
Application Services	ASI Generation
Business Services	BS Generation
ATP	**Business Service**
ATP ASI Test	**Business Service**
ATP ASI Test XML	**Business Service**
ATP Check	**Business Service**
Account - SAP Maps	**Business Service**
Account - SAP Maps (MO)	**Business Service**
Account Interface Maps - Siebel I...	**Business Service**
Account Interface Maps - Siebel I...	**Business Service**
Account Maps - Siebel Inbound	**Business Service**
Account Maps - Siebel Inbound (...	**Business Service**

Figure 9. Business Object Wizard, displaying Business Services

Confirm selection of objects

After you identify all the Siebel elements to be associated with the generated business object definitions, Business Object Designer displays the dialog box with only the selected objects and components. Figure 10 on page 64 illustrates this dialog box.


Business Object Wizard - Step 4 of 6 - Confirm source nodes for business ...

Business objects are about to be generated using the source nodes summarized below.
Click "Next" to start generating the objects or click "Back" to change your selection.

Name	Description
Convert	BO Conversion
BO_SiebelAccount.txt	**Repository File**
Generate	BO/IO/ASI/BS Generation
Business Objects	BO Generation
Abs Admin Constraint	**Business Object**
Integration Objects	IO Generation
ANI Message Header (PRM ANI)	**Integration Object**
Application Services	ASI Generation
Siebel Account	Application Service
ASI Objects	ASI Integration Object
Account Interface	**ASI Integration Object**
Business Services	BS Generation
ATP	**Business Service**

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

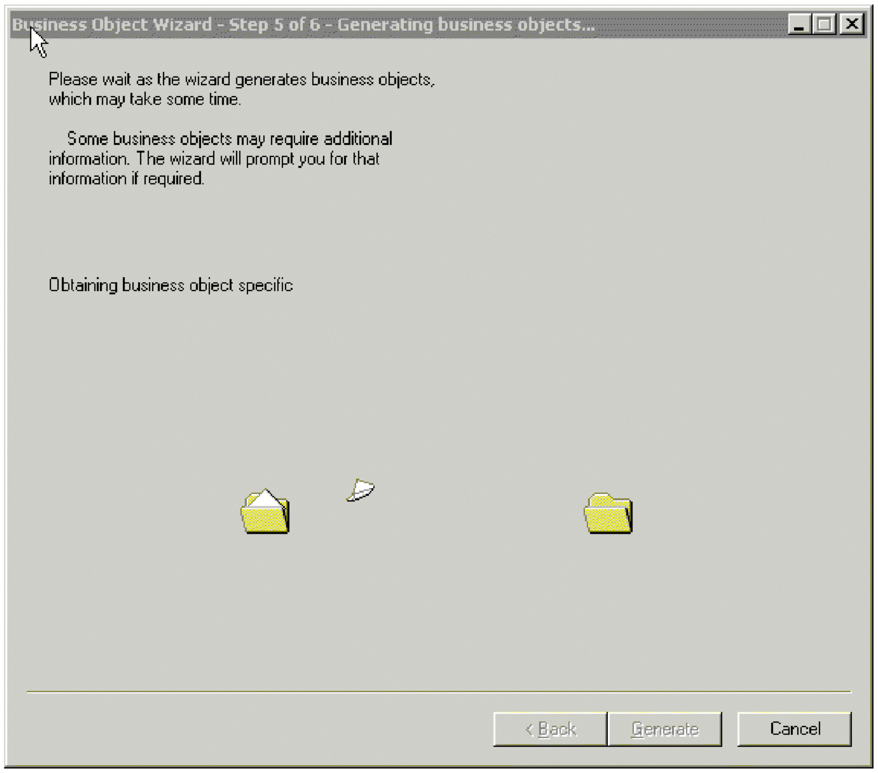
This window provides the following options:

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

Generate the business object definition

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

Figure 11 on page 65 illustrates this dialog box.


Business Object Wizard - Step 5 of 6 - Generating business objects...

Please wait as the wizard generates business objects, which may take some time.

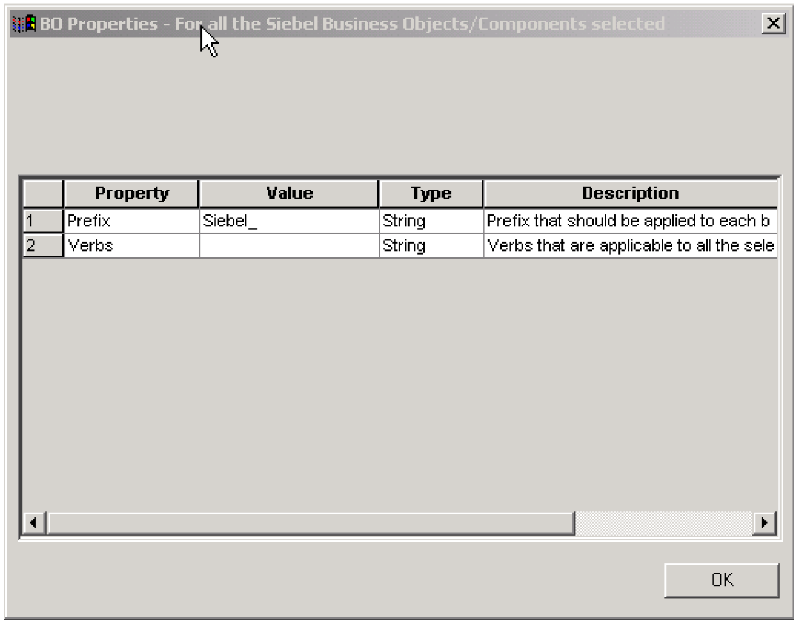
Some business objects may require additional information. The wizard will prompt you for that information if required.

Obtaining business object specific

Figure 11. Generating the business object definitions

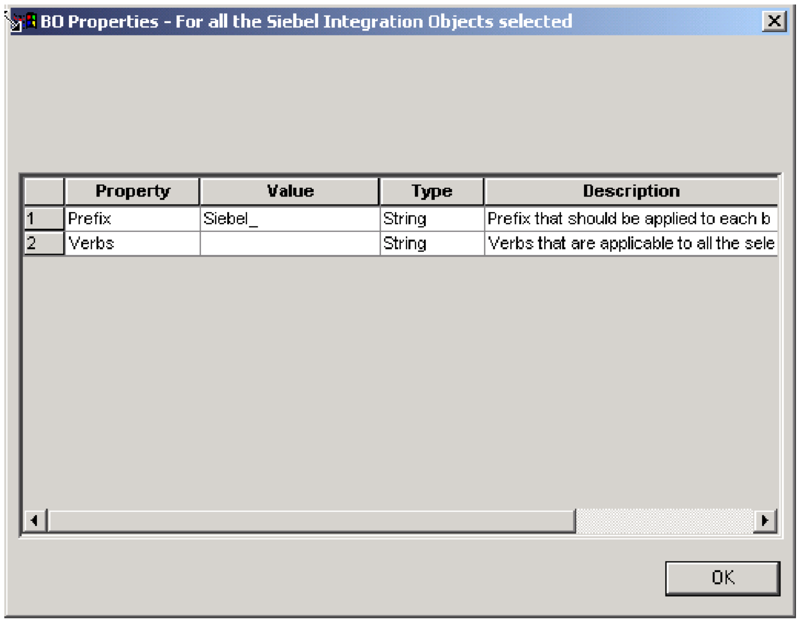
Provide additional information

Because SiebelODA needs additional information about the verbs, Business Object Designer displays the BO Properties window for each of the generation types you chose (business objects, integration objects, and application services), which prompts you for the information. Figure 12 on page 66 illustrates these screens.


BO Properties - For all the Siebel Business Objects/Components selected

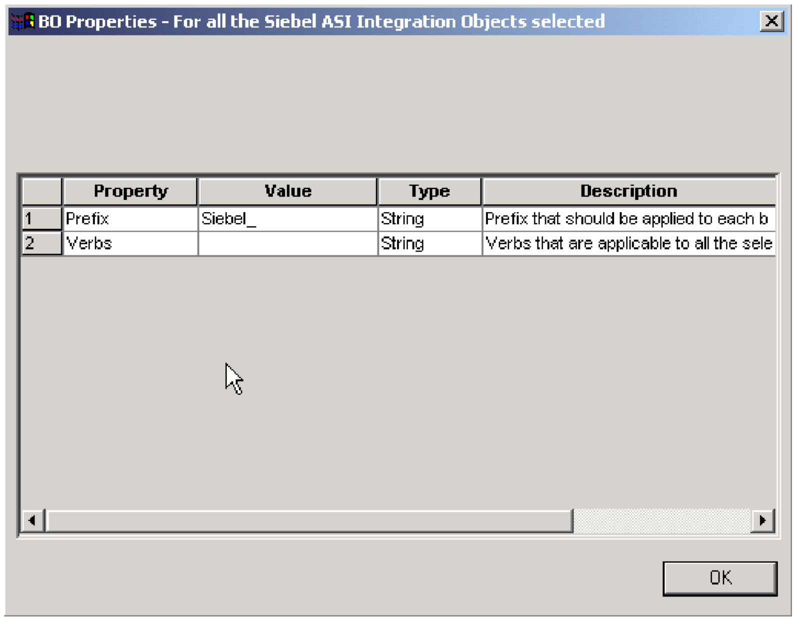
	Property	Value	Type	Description
1	Prefix	Siebel_	String	Prefix that should be applied to each b
2	Verbs		String	Verbs that are applicable to all the sele

Figure 12. Providing additional information for business object


BO Properties - For all the Siebel Integration Objects selected

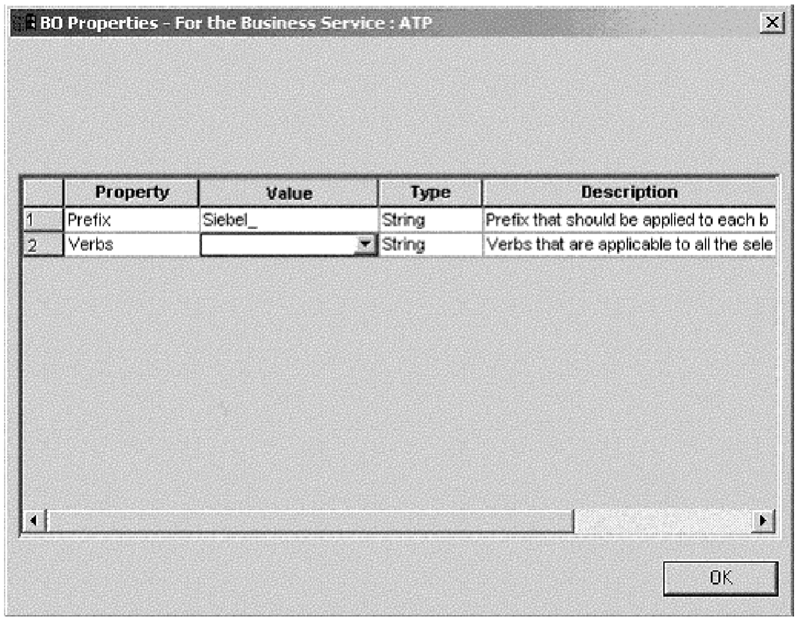
	Property	Value	Type	Description
1	Prefix	Siebel_	String	Prefix that should be applied to each b
2	Verbs		String	Verbs that are applicable to all the sele

Figure 13. Providing additional information for integration object


BO Properties - For all the Siebel ASI Integration Objects selected

	Property	Value	Type	Description
1	Prefix	Siebel_	String	Prefix that should be applied to each b
2	Verbs		String	Verbs that are applicable to all the sele

Figure 14. Providing additional information for application service


BO Properties - For the Business Service : ATP

	Property	Value	Type	Description
1	Prefix	Siebel_	String	Prefix that should be applied to each b
2	Verbs		String	Verbs that are applicable to all the sele

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

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

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

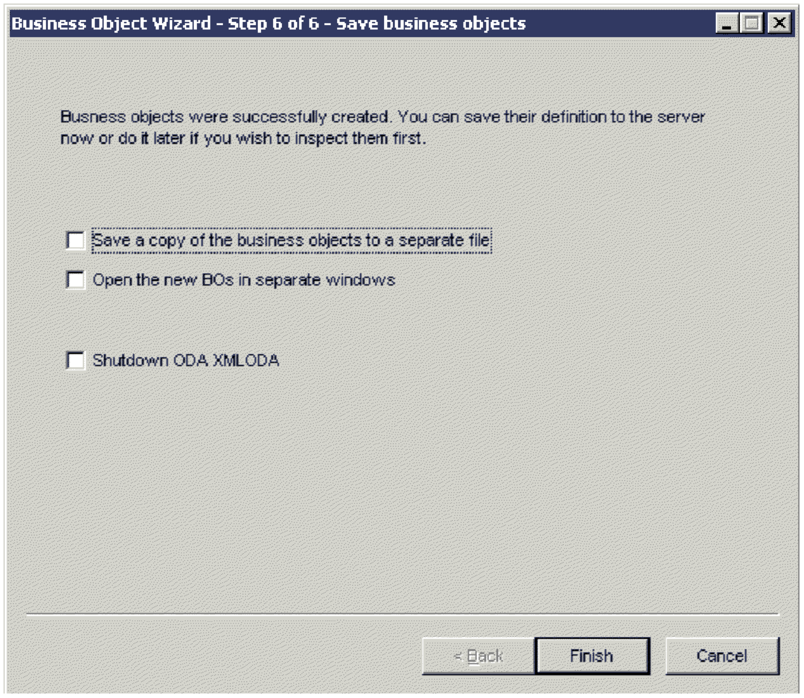
Save the business object definition

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

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

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

Figure 16 illustrates this dialog box.



Business Object Wizard - Step 6 of 6 - Save business objects

Busness objects were successfully created. You can save their definition to the server now or do it later if you wish to inspect them first.

☐ Save a copy of the business objects to a separate file

☐ Open the new BOs in separate windows

☐ Shutdown ODA XMLODA

Figure 16. Saving the business object definition

Reviewing the generated definition

The business object definition that SiebelODA generates contains:

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

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

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

This section describes:
- “Business-object-level properties” on page 69
- “Attribute properties” on page 69
- “Verbs” on page 70

Business-object-level properties

SiebelODA generates the following information at the business-object level:
- Name of the business object
- Version—defaults to `1.0.0`
- Application-specific information

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

Attribute properties

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

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

Name property

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

Data type property

When setting the type of an attribute, SiebelODA converts the data type of the attribute in the Siebel business component and converts it to the corresponding data type, as shown in Table 10. This is only in the case of business object generation, since business object conversion is for existing business objects.

Table 10. Correspondence of data types

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

Table 10. Correspondence of data types (continued)

DTYPE_CURRENCY	DOUBLE	

Note: If an attribute's data type is not one of those shown in Table 10, SiebelODA skips the column and displays a message stating that the column cannot be processed.

Cardinality property

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

MaxLength property

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

IsKey property

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

IsRequired property

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

AppSpecificInfo Property

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

PollQuantity

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

The default is 1.

Verbs

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

Adding information to the business object definition

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

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

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

Chapter 6. Using the adapter with Siebel business services

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

Understanding business services

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

"Description of business services"

"Processing business objects that support business services" on page 74

Description of business services

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

The adapter supports EAI Siebel Adapter, a generic business service provided by Siebel; Siebel-defined Application Service Interface (ASI); and custom-written business services.

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

Custom-written business objects are treated differently. Because they can implement any method, the IBM business object represents the service itself, not an integration object. For more information, see "Custom business service support" on page 76.

EAI Siebel Adapter and ASIs can be treated by the adapter as custom-written business services, and IBM business objects can be created to directly represent these services, although this is not recommended. See Chapter 5, "Creating business objects," on page 49.

Note: The adapter distinguishes between IBM objects representing business objects to Siebel, and IBM business objects representing Siebel integration objects by application-specific information ″BSN=″. See Table 11.

Processing business objects that support business services

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

Example:

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

Example: Siebel Integration object

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

Example: IBM business object representing Siebel integration object

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

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

For the Child Integration Component, it would be IO=<Name>;IC=<Name>

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

Table 11. Business object level application text

Parameter	Description
IO=	The name of the Siebel integration object corresponding to this business object.

Table 11. Business object level application text (continued)

Parameter	Description
`IC=`	The name of the Siebel integration component corresponding to this business object.
`BSN=`	The name of the business service used by this business object. When using application specific information, such as Siebel Account or Siebel Contact, the specific business service must be present. When using other integration objects, the Siebel Enterprise Applications Integration (EAI) must be present.
`SiebASI=`	(Deprecated) When a business object represents the ASI integration object, it contains `SiebASI=true`. The current version of the adapter and ODA instead use the `BSTYPE` application-specific information tag.
`BSTYPE=`	Determines the type of business service. • For EAI Siebel Adapter, where the IBM business object represents the Siebel integration object, the application-specific information should contain BSTYPE=GENERIC. • For Siebel ASIs, where the IBM business object represents the Siebel integration object, the application-specific information should contain BSTYPE=ASI. • For custom-written business services, where the IBM business object represents the service, the application-specific information should contain BSTYPE=CUSTOM.

Table 12. Simple attribute level application text

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

Verb processing with business services

The following verbs are supported by business services.

Note: The returned code for all the verbs in Table 13 is `VALCHANGE`.

Table 13. Verbs supported by business services

Verb	Description
`Delete`	Parent object keys are used to delete the Siebel object. The adapter verifies that all primary keys are present.
`Insert`	The complete incoming business object is used for the Insert verb.
`InsertOrUpdate (Upsert)`	If an object with the same keys as the input object exists, merge the specified input object with the existing object. Otherwise, create a new object in Siebel based on the input object. The adapter verifies the existence of all the primary keys before processing the object.

Table 13. Verbs supported by business services (continued)

Verb	Description
`QueryByExample` (or `Query` in the case of EAI Siebel Adapter)	Queries for objects based on the example object provided. This operation can be treated as a `Retrieve by content` operations.
`QueryById`	If the object with the keys exists, it is queried or retrieved. These operations can be treated as `Retrieve` operations.
`Update`	If an object with the same keys as the input object exists, merge the specified input object with the existing object. Otherwise, error out.
`Synchronize`	If an object with the same keys as the input object exists, make it look like the input object. Otherwise, create a new object in Siebel based on the input object.

The following example describes a process flow for using any of the verbs in Table 13 on page 75.

Verbs of the IBM business object represent the Methods of the business service.

Verb Processing:
- Get the business service name
- construct the property set based on the input
- invoke the verb on the specified business service passing in the input property set, then
- construct the business object from the output property set

Events detection with business services

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

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

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

Custom business service support

When processing IBM business objects that correspond to Siebel custom business services, the adapter checks for application specific information. If `BSTYPE=Custom` is found, it gets the Request business object under the top level business object. When the adapter builds the property set that corresponds to the incoming business service, the simple attributes of the Request business object provide the properties. If present, the SiebelMessage container attribute provides the integration object. The adapter can process the business object with or without the existence of the SiebelMessage container attribute. The adapter does the following:

1. Instantiates a new property set
2. Sets the simple attribute vales as the properties of the new property set
3. Takes the Siebel Message object, if present, as the child property set

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

IBM business object corresponding to a business service:

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

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

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

IBM business object corresponding to an integration object:

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

Chapter 7. Running the connector

Starting the connector

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

```
ProductDir\connectors\connName
```

where *connName* identifies the connector.

On UNIX systems the startup script should reside in the *ProductDir*/bin directory.

The name of the startup script depends on the operating-system platform, as Table 14 shows.

Table 14. Startup scripts for a connector

Operating system	**Startup script**
UNIX-based systems	`connector_manager`
Windows	`start_connName.bat`

When the startup script runs, it expects by default to find the configuration file in the *Productdir* (see the commands below). This is where you place your configuration file.

Note: You need a local configuration file if the adapter is using JMS transport.

You can invoke the connector startup script in any of the following ways:
- On Windows systems, from the **Start** menu

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

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

    ```
    connector_manager -start connName brokerName [-cconfigFile ]
    ```

  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 (available only when the broker is WebSphere Application Server or InterChange Server), which is launched when you start System Manager

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.
- From System Manager (available for all brokers)

  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 8. Troubleshooting

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

MAX_LOG_FILE_SIZE error on UNIX

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

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

This error message may result from the following conditions:

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

Decreasing the size of the Siebel log file

Seibel JAVABean allows you to change the logging timeout value.

To reset the logging timeout value in Siebel JAVABean:

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

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

Memory limitations with result set support

When result set support is being used on DB2, The adapter has a JVM memory restriction of 2GB for J2SE JRE 1.4.1, version SR2, that is provided with the adapter. To enable the result set process to utilize the 2GB memory, your environment must use hardware that facilitates efficient memory utilization without excessive paging.

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 with 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 (and shown as WMQI in the Connector Configurator).
- Information Integrator (II)
- WebSphere Application Server (WAS)

If your adapter supports DB2 Information Integrator, use the WMQI options and the DB2 II standard properties (see the Notes column in Table 15 on page 85.)

The properties you set for the adapter depend on which integration broker you use. You choose the integration broker using Connector Configurator. After you choose the broker, Connector Configurator lists the standard properties you must configure for the adapter.

For information about properties specific to this connector, see the relevant section in this guide.

New properties

These standard properties have been added in this release:

- AdapterHelpName
- BiDi.Application
- BiDi.Broker
- BiDi.Metadata
- BiDi.Transformation
- CommonEventInfrastructure
- CommonEventInfrastructureContextURL
- ControllerEventSequencing
- jms.ListenerConcurrency
- jms.TransportOptimized
- ResultsSetEnabled
- ResultsSetSize
- TivoliTransactionMonitorPerformance

Standard connector properties overview

Connectors have two types of configuration properties:

- Standard configuration properties, which are used by the framework
- Application, or connector-specific, configuration properties, which are used by the agent

These properties determine the adapter framework and the agent run-time behavior.

This section describes how to start Connector Configurator and describes characteristics common to all properties. For information on configuration properties specific to a connector, see its adapter user guide.

Starting 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.

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.

Configuration property values overview

The connector uses the following order to determine a property's value:

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

The default length of a property field is 255 characters. There is no limit on the length of a STRING property type. The length of an INTEGER type is determined by the server on which the adapter is running.

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.

The update characteristics of a property, that is, how and when a change to the connector properties takes effect, depend on the nature of the property.

There are four update methods for standard connector properties:

- **Dynamic**
  The new value takes effect immediately after the change is saved in System Manager. However, if the connector is in stand-alone mode (independently of System Manager), for example, if it is running with one of the WebSphere message brokers, you can change properties only through the configuration file. In this case, a dynamic update is not possible.
- **Agent restart (ICS only)**
  The new value takes effect only after you stop and restart the connector agent.
- **Component restart**
  The new value takes effect only after the connector is stopped and then restarted in System Manager. You do not need to stop and restart the agent or the server process.

- **System restart**
  The new value takes effect only after you stop and restart the connector agent and the server.

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 15 on page 85.

There are three locations in which a standard property can reside. Some properties can reside in more than one location.

- **ReposController**
  The property resides in the connector controller and is effective only there. If you change the value on the agent side, it does not affect the controller.
- **ReposAgent**
  The property resides in the agent and is effective only there. A local configuration can override this value, depending on the property.
- **LocalConfig**
  The property resides in the configuration file for the connector and can act only through the configuration file. The controller cannot change the value of the property, and is not aware of changes made to the configuration file unless the system is redeployed to update the controller explicitly.

Standard properties quick-reference

Table 15 provides a quick-reference to the standard connector configuration properties. Not all connectors require all of these properties, and property settings may differ from integration broker to integration broker.

See the section following the table for a description of each property.

Note: In the Notes column in Table 15, the phrase "RepositoryDirectory is set to <REMOTE>" indicates that the broker is InterChange Server. When the broker is WMQI or WAS, the repository directory is set to *<ProductDir>*\repository

Table 15. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdapterHelpName	One of the valid subdirectories in *<ProductDir>*\bin\Data \App\Help\ that contains a valid *<RegionalSetting>* directory	Template name, if valid, or blank field	Component restart	Supported regional settings. Include `chs_chn`, `cht_twn`, `deu_deu`, `esn_esp`, `fra_fra`, `ita_ita`, `jpn_jpn`, `kor_kor`, `ptb_bra`, and `enu_usa` (default).
AdminInQueue	Valid JMS queue name	*<CONNECTORNAME>* /ADMININQUEUE	Component restart	This property is valid only when the value of DeliveryTransport is JMS
AdminOutQueue	Valid JMS queue name	*<CONNECTORNAME>* /ADMINOUTQUEUE	Component restart	This property is valid only when the value of DeliveryTransport is JMS

Table 15. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
AgentConnections	1 through 4	1	Component restart	This property is valid only when the value of DeliveryTransport is MQ or IDL, the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
AgentTraceLevel	0 through 5	0	Dynamic if broker is ICS; otherwise Component restart	
ApplicationName	Application name	The value specified for the connector application name	Component restart	
BiDi.Application	Any valid combination of these bidirectional attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (five letters)	Component restart	This property is valid only if the value of BiDi.Transforma tion is true
BiDi.Broker	Any valid combination of these bidirectional attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (five letters)	Component restart	This property is valid only if the value of BiDi.Transformation is true. If the value of BrokerType is ICS, the property is read-only.
BiDi.Metadata	Any valid combination of these bidirectional attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (five letters)	Component restart	This property is valid only if the value of BiDi.Transformation is true.
BiDi.Transformation	true or false	false	Component restart	This property is valid only if the value of BrokerType is not WAS .
BrokerType	ICS, WMQI, WAS	ICS	Component restart	
CharacterEncoding	Any supported code. The list shows this subset: ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437 .	ascii7	Component restart	This property is valid only for C++ connectors.

Table 15. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
CommonEventInfrastruc ture	`true` or `false`	`false`	Component restart	
CommonEventInfrastruc tureURL	A URL string, for example, `corbaloc:iiop: host:2809`.	No default value.	Component restart	This property is valid only if the value of CommonEvent Infrastructure is `true`.
ConcurrentEventTrig geredFlows	`1` through `32,767`	`1`	Component restart	This property is valid only if the value of RepositoryDirectory is set to <REMOTE> and the value of BrokerType is ICS.
ContainerManagedEvents	Blank or `JMS`	Blank	Component restart	This property is valid only when the value of Delivery Transport is `JMS`.
ControllerEventSequenc ing	`true` or `false`	`true`	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ControllerStoreAndFor wardMode	`true` or `false`	`true`	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ControllerTraceLevel	`0` through `5`	`0`	Dynamic	This property is valid only if the value of RepositoryDirectory is set to <REMOTE> and the value of BrokerType is ICS.
DeliveryQueue	Any valid JMS queue name	`<CONNECTORNAME> /DELIVERYQUEUE`	Component restart	This property is valid only when the value of Delivery Transport is `JMS`.
DeliveryTransport	`MQ`, `IDL`, or `JMS`	`IDL` when the value of RepositoryDirectory is <REMOTE>, otherwise `JMS`	Component restart	If the value of RepositoryDirectory is not <REMOTE>, the only valid value for this property is `JMS`.
DuplicateEventElimina tion	`true` or `false`	`false`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS`.
EnableOidForFlowMoni toring	`true` or `false`	`false`	Component restart	This property is valid only if the value of BrokerType is ICS.
FaultQueue	Any valid queue name.	`<CONNECTORNAME> /FAULTQUEUE`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS`.
jms.FactoryClassName	`CxCommon.Messaging.jms .IBMMQSeriesFactory`, `CxCommon.Messaging .jms.SonicMQFactory`, or any Java class name	`CxCommon.Messaging. jms.IBMMQSeriesFactory`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS`.

Table 15. Summary of standard configuration properties (continued)

Property name	**Possible values**	**Default value**	**Update method**	**Notes**
jms.ListenerConcurrency	1 through 32767	1	Component restart	This property is valid only if the value of jms.TransportOptimized is true.
jms.MessageBrokerName	If the value of jms.FactoryClassName is IBM, use crossworlds.queue. manager.	crossworlds.queue. manager	Component restart	This property is valid only if the value of DeliveryTransport is JMS .
jms.NumConcurrent Requests	Positive integer	10	Component restart	This property is valid only if the value of DeliveryTransport is JMS .
jms.Password	Any valid password		Component restart	This property is valid only if the value of DeliveryTransport is JMS .
jms.TransportOptimized	true or false	false	Component restart	This property is valid only if the value of DeliveryTransport is JMS and the value of BrokerType is ICS.
jms.UserName	Any valid name		Component restart	This property is valid only if the value of Delivery Transport is JMS.
JvmMaxHeapSize	Heap size in megabytes	128m	Component restart	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
JvmMaxNativeStackSize	Size of stack in kilobytes	128k	Component restart	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
JvmMinHeapSize	Heap size in megabytes	1m	Component restart	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ListenerConcurrency	1 through 100	1	Component restart	This property is valid only if the value of DeliveryTransport is MQ.
Locale	This is a subset of the supported locales: en_US, ja_JP, ko_KR, zh_CN, zh_TW, fr_FR, de_DE, it_IT, es_ES, pt_BR	en_US	Component restart	

Table 15. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
LogAtInterchangeEnd	`true` or `false`	`false`	Component restart	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
MaxEventCapacity	1 through `2147483647`	`2147483647`	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
MessageFileName	Valid file name	`InterchangeSystem.txt`	Component restart	
MonitorQueue	Any valid queue name	`<CONNECTORNAME>` `/MONITORQUEUE`	Component restart	This property is valid only if the value of DuplicateEventElimination is `true` and ContainerManagedEvents has no value.
OADAutoRestartAgent	`true` or `false`	`false`	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
OADMaxNumRetry	A positive integer	`1000`	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
OADRetryTimeInterval	A positive integer in minutes	`10`	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.

PollEndTime	HH = 0 through 23
	MM = 0 through 59
		`HH:MM`	Component restart	

PollFrequency	A positive integer (in milliseconds)	`10000`	Dynamic if broker is ICS; otherwise Component restart	
PollQuantity	1 through `500`	`1`	Agent restart	This property is valid only if the value of ContainerManagedEvents is `JMS`.

PollStartTime	HH = 0 through 23
	MM = 0 through 59
		`HH:MM`	Component restart	

RepositoryDirectory	<REMOTE> if the broker is ICS; otherwise any valid local directory.	For ICS, the value is set to <REMOTE> For WMQI and WAS, the value is `<ProductDir` `\repository`	Agent restart	

Table 15. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
RequestQueue	Valid JMS queue name	*<CONNECTORNAME>* /REQUESTQUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS
ResponseQueue	Valid JMS queue name	*<CONNECTORNAME>* /RESPONSEQUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
RestartRetryCount	0 through 99	3	Dynamic if ICS; otherwise Component restart	
RestartRetryInterval	A value in minutes from 1 through 2147483647	1	Dynamic if ICS; otherwise Component restart	
ResultsSetEnabled	true or false	false	Component restart	Used only by connectors that support DB2II. This property is valid only if the value of DeliveryTransport is JMS, and the value of BrokerType is WMQI.
ResultsSetSize	Positive integer	0 (means the results set size is unlimited)	Component restart	Used only by connectors that support DB2II. This property is valid only if the value of ResultsSetEnabled is true.
RHF2MessageDomain	mrm or xml	mrm	Component restart	This property is valid only if the value of DeliveryTransport is JMS and the value of WireFormat is CwXML.
SourceQueue	Any valid WebSphere MQ queue name	*<CONNECTORNAME>* /SOURCEQUEUE	Agent restart	This property is valid only if the value of ContainerManagedEvents is JMS.
SynchronousRequest Queue	Any valid queue name.	*<CONNECTORNAME>* /SYNCHRONOUSREQUEST QUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
SynchronousRequest Timeout	0 to any number (milliseconds)	0	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
SynchronousResponse Queue	Any valid queue name	*<CONNECTORNAME>* /SYNCHRONOUSRESPONSE QUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
TivoliMonitorTransaction Performance	true or false	false	Component restart	

Table 15. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
WireFormat	`CwXML` or `CwBO`	`CwXML`	Agent restart	The value of this property must be `CwXML` if the value of RepositoryDirectory is not set to <REMOTE>. The value must be `CwBO` if the value of RepositoryDirectory is set to <REMOTE>.
WsifSynchronousRequest Timeout	`0` to any number (milliseconds)	`0`	Component restart	This property is valid only if the value of BrokerType is `WAS`.
XMLNameSpaceFormat	`short` or `long`	`short`	Agent restart	This property is valid only if the value of BrokerType is WMQI or WAS

Standard properties

This section describes the standard connector configuration properties.

AdapterHelpName

The AdapterHelpName property is the name of a directory in which connector-specific extended help files are located. The directory must be located in <*ProductDir*>\bin\Data\App\Help and must contain at least the language directory enu_usa. It may contain other directories according to locale.

The default value is the template name if it is valid, or it is blank.

AdminInQueue

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

The default value is `<`*`CONNECTORNAME`*`>/ADMININQUEUE`

AdminOutQueue

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

The default value is `<`*`CONNECTORNAME`*`>/ADMINOUTQUEUE`

AgentConnections

The AgentConnections property controls the number of ORB (Object Request Broker) connections opened when the ORB initializes.

It is valid only if the value of the RepositoryDirectory is set to <REMOTE> and the value of the DeliveryTransport property is MQ or IDL.

The default value of this property is 1.

AgentTraceLevel

The AgentTraceLevel property sets the level of trace messages for the application-specific component. The connector delivers all trace messages applicable at the tracing level set and lower.

The default value is `0`.

ApplicationName

The ApplicationName property uniquely identifies the name of the connector application. This name is used by the system administrator to monitor the integration environment. This property must have a value before you can run the connector.

The default is the name of the connector.

BiDi.Application

The BiDi.Application property specifies the bidirectional format for data coming from an external application into the adapter in the form of any business object supported by this adapter. The property defines the bidirectional attributes of the application data. These attributes are:

- Type of text: implicit or visual (`I` or `V`)
- Text direction: left-to-right or right-to-left (`L` or `R`)
- Symmetric swapping: on or off (`Y` or `N`)
- Shaping (Arabic): on or off (`S` or `N`)
- Numerical shaping (Arabic): Hindi, contextual, or nominal (`H`, `C`, or `N`)

This property is valid only if the BiDi.Transformation property value is set to true.

The default value is `ILYNN` (implicit, left-to-right, on, off, nominal).

BiDi.Broker

The BiDi.Broker property specifies the bidirectional format for data sent from the adapter to the integration broker in the form of any supported business object. It defines the bidirectional attributes of the data, which are as listed under `BiDi.Application` above.

This property is valid only if the BiDi.Transformation property value is set to true. If the BrokerType property is ICS, the property value is read-only.

The default value is `ILYNN` (implicit, left-to-right, on, off, nominal).

BiDi.Metadata

The BiDi.Metadata property defines the bidirectional format or attributes for the metadata, which is used by the connector to establish and maintain a link to the external application. The attribute settings are specific to each adapter using the bidirectional capabilities. If your adapter supports bidirectional processing, refer to section on adapter-specific properties for more information.

This property is valid only if the BiDi.Transformation property value is set to true.

The default value is `ILYNN` (implicit, left-to-right, on, off, nominal).

BiDi.Transformation

The BiDi.Transformation property defines whether the system performs a bidirectional transformation at run time.

If the property value is set to `true`, the BiDi.Application, BiDi.Broker, and BiDi.Metadata properties are available. If the property value is set to `false`, they are hidden.

The default value is `false`.

BrokerType

The BrokerType property identifies the integration broker type that you are using. The possible values are `ICS`, `WMQI (for WMQI, WMQIB or WBIMB)`, or `WAS`.

CharacterEncoding

The CharacterEncoding property 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. C++ connectors use the value `ascii7` for this property.

By default, only a subset of supported character encodings is displayed. To add other supported values to the list, you must manually modify the \Data\Std\stdConnProps.xml file in the product directory (<*ProductDir*>). For more information, see the Connector Configurator appendix in this guide.

CommonEventInfrastructure

The Common Event Infrastructure (CEI) is a simple event management function handling generated events. The CommonEventInfrastructure property specifies whether the CEI should be invoked at run time.

The default value is `false`.

CommonEventInfrastructureContextURL

The CommonEventInfrastructureContextURL is used to gain access to the WAS server that executes the Common Event Infrastructure (CEI) server application. This property specifies the URL to be used.

This property is valid only if the value of CommonEventInfrastructure is set to `true.`

The default value is a blank field.

ConcurrentEventTriggeredFlows

The ConcurrentEventTriggeredFlows property determines how many business objects can be concurrently processed by the connector for event delivery. You set the value of this attribute to the number of business objects that are mapped and delivered concurrently. For example, if you set the value of this property to `5`, five business objects are processed concurrently.

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), the following properties must configured:

- The collaboration must be configured to use multiple threads by setting its `Maximum number of concurrent events` property high enough to use multiple threads.
- The destination application's application-specific component must be configured to process requests concurrently. That is, it must be multithreaded, or it must be able to use connector agent parallelism and be configured for multiple processes. The Parallel Process Degree configuration property must be set to a value larger than 1.

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

This property is valid only if the value of the RepositoryDirectory property is set to <REMOTE>.

The default value is 1.

ContainerManagedEvents

The ContainerManagedEvents 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 one JMS transaction.

When this property is set to `JMS`, the following properties must also be set to enable guaranteed event delivery:

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

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

Although these properties are adapter-specific, here are some example values:

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

The fields for these values in the **Data Handler** tab are displayed only if you have set the ContainerManagedEvents property to the value `JMS`.

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

The ContainerManagedEvents property is valid only if the value of the DeliveryTransport property is set to `JMS`.

There is no default value.

ControllerEventSequencing

The ControllerEventSequencing property enables event sequencing in the connector controller.

This property is valid only if the value of the RepositoryDirectory property is set to set to <REMOTE> (`BrokerType` is `ICS`).

The default value is `true`.

ControllerStoreAndForwardMode

The ControllerStoreAndForwardMode property 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.

This property is valid only if the value of the RepositoryDirectory property is set to <REMOTE> (the value of the BrokerType property is ICS).

The default value is `true`.

ControllerTraceLevel

The ControllerTraceLevel property sets the level of trace messages for the connector controller.

This property is valid only if the value of the RepositoryDirectory property is set to set to <REMOTE>.

The default value is `0`.

DeliveryQueue

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

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

The default value is `<`*`CONNECTORNAME`*`>/DELIVERYQUEUE`.

DeliveryTransport

The DeliveryTransport property 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 value of the RepositoryDirectory property is set to <REMOTE>, the value of the DeliveryTransport property can be `MQ`, `IDL`, or `JMS`, and the default is `IDL`.
- If the value of the RepositoryDirectory property is a local directory, the value can be only `JMS`.

The connector sends service-call requests and administrative messages over CORBA IIOP if the value of the RepositoryDirectory property is `MQ` or `IDL`.

The default value is JMS.

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 prevents writing 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 polling thread picks up an event, places it in the connector queue, then picks up the next event. This is faster than IDL, which requires the connector polling thread to pick up an event, go across the network into the server process, store the event persistently in the repository database, then pick up the next event.

JMS

The JMS transport mechanism 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` are listed in Connector Configurator. The properties `jms.MessageBrokerName` and `jms.FactoryClassName` are required for this transport.

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
- 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 768MB of process heap size, set the following variable and property:

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

This script is located in the \bin directory below the product directory (<*ProductDir*>). Using 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 larger than this limit, page swapping can occur, which can adversely affect the performance of your system.

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

DuplicateEventElimination

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

Note: When the value of this property is `true`, the MonitorQueue property must be enabled to provide guaranteed event delivery.

The default value is `false`.

EnableOidForFlowMonitoring

When the value of this property is `true`, the adapter runtime will mark the incoming ObjectEventID as a foreign key for flow monitoring.

This property is only valid if the BrokerType property is set to ICS.

The default value is `false`.

FaultQueue

If the connector experiences an error while processing a message, it moves the message (and a status indicator and description of the problem) to the queue specified in the FaultQueue property.

The default value is `<`*`CONNECTORNAME`*`>/FAULTQUEUE`.

jms.FactoryClassName

The jms.FactoryClassName property specifies the class name to instantiate for a JMS provider. This property must be set if the value of the DeliveryTransport property is `JMS`.

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

jms.ListenerConcurrency

The jms.ListenerConcurrency property specifies the number of concurrent listeners for the JMS controller. It specifies the number of threads that fetch and process messages concurrently within a controller.

This property is valid only if the value of the jms.OptimizedTransport property is `true`.

The default value is `1`.

jms.MessageBrokerName

The jms.MessageBrokerName specifies the broker name to use for the JMS provider. You must set this connector property if you specify `JMS` as the delivery transport mechanism (in the DeliveryTransport property).

When you connect to a remote message broker, this property requires the following values:
`QueueMgrName``:`*`Channel`*`:`*`HostName`*`:`*`PortNumber`*
where:
`QueueMgrName` is the name of the queue manager.
`Channel` is the channel used by the client.
`HostName` is the name of the machine where the queue manager is to reside.
*`PortNumber`*is the port number used by the queue manager for listening

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

The default value is `crossworlds.queue.manager`. Use the default when connecting to a local message broker.

jms.NumConcurrentRequests

The jms.NumConcurrentRequests property 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 are blocked and must wait for another request to complete before proceeding.

The default value is `10`.

jms.Password

The jms.Password property specifies the password for the JMS provider. A value for this property is optional.

There is no default value.

jms.TransportOptimized

The jms.TransportOptimized property determines if the WIP (work in progress) is optimized. You must have a WebSphere MQ provider to optimize the WIP. For optimized WIP to operate, the messaging provider must be able to:

1. Read a message without taking it off the queue
2. Delete a message with a specific ID without transferring the entire message to the receiver's memory space
3. Read a message by using a specific ID (needed for recovery purposes)
4. Track the point at which events that have not been read appear.

The JMS APIs cannot be used for optimized WIP because they do not meet conditions 2 and 4 above, but the MQ Java APIs meet all four conditions, and hence are required for optimized WIP.

This property is valid only if the value of DeliveryTransport is `JMS` and the value of BrokerType is `ICS`.

The default value is `false`.

jms.UserName

the jms.UserName property specifies the user name for the JMS provider. A value for this property is optional.

There is no default value.

JvmMaxHeapSize

The JvmMaxHeapSize property specifies the maximum heap size for the agent (in megabytes).

This property is valid only if the value for the RepositoryDirectory property is set to <REMOTE>.

The default value is 128m.

JvmMaxNativeStackSize

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

This property is valid only if the value for the RepositoryDirectory property is set to <REMOTE>.

The default value is 128k.

JvmMinHeapSize

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

This property is valid only if the value for the RepositoryDirectory property is set to <REMOTE>.

The default value is 1m.

ListenerConcurrency

The ListenerConcurrency property supports multithreading in WebSphere MQ Listener when ICS is the integration broker. It enables batch writing of multiple events to the database, thereby improving system performance.

This property valid only with connectors that use MQ transport. The value of the DeliveryTransport property must be `MQ`.

The default value is 1.

Locale

The Locale property specifies the language code, country or territory, and, optionally, the associated character code set. The value of this property determines cultural conventions such 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 is a two-character language code (in lowercase letters)
TT is a two-letter country or territory code (in uppercase letters)
codeset is the name of the associated character code set (may be optional).

By default, only a subset of supported locales are listed. To add other supported values to the list, you modify the \Data\Std\stdConnProps.xml file in the <*ProductDir*>\bin directory. For more information, refer to the Connector Configurator appendix in this guide.

If the connector has not been internationalized, the only valid value for this property is `en_US`. To determine whether a specific connector has been globalized, refer to the user guide for that adapter.

The default value is `en_US`.

LogAtInterchangeEnd

The LogAtInterchangeEnd property specifies whether to log errors to the log destination of the integration broker.

Logging to the log destination also turns on e-mail notification, which generates e-mail messages for the recipient specified as the value of MESSAGE_RECIPIENT in the InterchangeSystem.cfg file when errors or fatal errors occur. For example, when a connector loses its connection to the application, if the value of LogAtInterChangeEnd is `true`, an e-mail message is sent to the specified message recipient.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

The default value is `false`.

MaxEventCapacity

The MaxEventCapacity property specifies maximum number of events in the controller buffer. This property is used by the flow control feature.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

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

The default value is 2147483647.

MessageFileName

The MessageFileName property specifies the name of the connector message file. The standard location for the message file is \connectors\messages in the product directory. Specify the message file name 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 connector has its own message file, see the individual adapter user guide.

The default value is `InterchangeSystem.txt`.

MonitorQueue

The MonitorQueue property specifies the logical queue that the connector uses to monitor duplicate events.

It is valid only if the value of the DeliveryTransport property is `JMS` and the value of the DuplicateEventElimination is `true`.

The default value is `<`*`CONNECTORNAME`*`>/MONITORQUEUE`

OADAutoRestartAgent

the OADAutoRestartAgent property specifies whether the connector uses the automatic and remote restart feature. This feature uses the WebSphere 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 WebSphere MQ-triggered OAD feature. see the *Installation Guide for Windows* or *for UNIX*.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

The default value is `false`.

OADMaxNumRetry

The OADMaxNumRetry property specifies the maximum number of times that the WebSphere MQ-triggered Object Activation Daemon (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.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

The default value is `1000`.

OADRetryTimeInterval

The OADRetryTimeInterval property specifies the number of minutes in the retry-time interval for the WebSphere MQ-triggered Object Activation Daemon (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.

This property is valid only if the value of the RespositoryDirectory property is set to <REMOTE> (the value of BrokerType is ICS).

The default value is `10`.

PollEndTime

The PollEndTime property specifies the time to stop polling the event queue. The format is *HH:MM*, where *HH* is 0 through 23 hours, and *MM* represents 0 through 59 minutes.

You must provide a valid value for this property. The default value is HH:MM without a value, and it must be changed.

If the adapter runtime detects:
- PollStartTime set and PollEndTime not set, or
- PollEndTime set and PollStartTime not set

it will poll using the value configured for the PollFrequency property.

PollFrequency

The PollFrequency property 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 the PollQuantity property.
- Process these objects. For some connectors, this may be partly done on separate threads, which execute asynchronously to the next polling action.
- Delay for the interval specified by the PollFrequency property.
- Repeat the cycle.

The following values are valid for this property:
- The number of milliseconds between polling actions (a positive integer).
- The word no, which causes the connector not to poll. Enter the word in lowercase.
- The word key, which causes the connector to poll only when you type the letter p in the connector Command Prompt window. 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

The PollQuantity property designates the number of items from the application that the connector polls for. If the adapter has a connector-specific property for setting the poll quantity, the value set in the connector-specific property overrides the standard property value.

This property is valid only if the value of the DeliveryTransport property is JMS, and the ContainerManagedEvents property has a value.

An e-mail message is also considered an event. The connector actions are as follows when it is polled for e-mail.
- When it is polled once, the connector detects the body of the message, which it reads as an attachment. Since no data handler was specified for this mime type, it will then ignore the message.

- The connector processes the first BO attachment. The data handler is available for this MIME type, so it sends the business object to Visual Test Connector.
- When it is polled for the second time, the connector processes the second BO attachment. The data handler is available for this MIME type, so it sends the business object to Visual Test Connector.
- Once it is accepted, the third BO attachment should be transmitted.

PollStartTime

The PollStartTime property specifies the time to start polling the event queue. The format is `HH:MM`, where `HH` is 0 through 23 hours, and `MM` represents 0 through 59 minutes.

You must provide a valid value for this property. The default value is `HH:MM` without a value, and it must be changed.

If the adapter runtime detects:

- PollStartTime set and PollEndTime not set, or
- PollEndTime set and PollStartTime not set

it will poll using the value configured for the PollFrequency property.

RepositoryDirectory

The RepositoryDirectory property is the location of the repository from which the connector reads the XML schema documents that store the metadata for business object definitions.

If the integration broker is ICS, this value must be set to 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 is set to <*ProductDir*>\repository by default. However, it may be set to any valid directory name.

RequestQueue

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

This property is valid only if the value of the DeliveryTransport property is `JMS`.

The default value is `<CONNECTORNAME>/REQUESTQUEUE`.

ResponseQueue

The ResponseQueue property specifies 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.

This property is valid only if the value of the DeliveryTransport property is `JMS`.

The default value is `<CONNECTORNAME>/RESPONSEQUEUE`.

RestartRetryCount

The RestartRetryCount property specifies the number of times the connector attempts to restart itself. When this property is used for a connector that is connected in parallel, it specifies the number of times the master connector application-specific component attempts to restart the client connector application-specific component.

The default value is `3`.

RestartRetryInterval

The RestartRetryInterval property specifies the interval in minutes at which the connector attempts to restart itself. When this property is used for a connector that is linked in parallel, it specifies the interval at which the master connector application-specific component attempts to restart the client connector application-specific component.

Possible values for the property range from 1 through 2147483647.

The default value is `1`.

ResultsSetEnabled

The ResultsSetEnabled property enables or disables results set support when Information Integrator is active. This property can be used only if the adapter supports DB2 Information Integrator.

This property is valid only if the value of the DeliveryTransport property is `JMS`, and the value of BrokerType is `WMQI`.

The default value is `false`.

ResultsSetSize

The ResultsSetSize property defines the maximum number of business objects that can be returned to Information Integrator. This property can be used only if the adapter supports DB2 Information Integrator.

This property is valid only if the value of the ResultsSetEnabled property is `true`.

The default value is `0`. This means that the size of the results set is unlimited.

RHF2MessageDomain

The RHF2MessageDomain property allows you to configure the value of the field domain name in the JMS header. When data is sent to a WebSphere message broker over JMS transport, the adapter framework writes JMS header information, with a domain name and a fixed value of `mrm`. A configurable domain name lets you track how the WebSphere message broker processes the message data.

This is an example header:

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

This property is valid only if the value of BrokerType is WMQI or WAS. Also, it is valid only if the value of the DeliveryTransport property is `JMS`, and the value of the WireFormat property is `CwXML`.

Possible values are `mrm` and `xml`. The default value is `mrm`.

SourceQueue

The SourceQueue property 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 94.

This property is valid only if the value of DeliveryTransport is `JMS`, and a value for ContainerManagedEvents is specified.

The default value is `<`*`CONNECTORNAME`*`>/SOURCEQUEUE`.

SynchronousRequestQueue

The SynchronousRequestQueue property 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 synchronous request queue and waits for a response from the broker on the synchronous response queue. The response message sent to the connector has a correlation ID that matches the ID of the original message.

This property is valid only if the value of DeliveryTransport is `JMS`.

The default value is `<`*`CONNECTORNAME`*`>/SYNCHRONOUSREQUESTQUEUE`

SynchronousRequestTimeout

The SynchronousRequestTimeout property specifies the time in milliseconds that the connector waits for a response to a synchronous request. If the response is not received within the specified time, the connector moves the original synchronous request message (and error message) to the fault queue.

This property is valid only if the value of DeliveryTransport is `JMS`.

The default value is `0`.

SynchronousResponseQueue

The SynchronousResponseQueue property delivers response messages in reply to a synchronous request from the broker to the connector framework. This queue is necessary only if the connector uses synchronous execution.

This property is valid only if the value of DeliveryTransport is `JMS`.

The default is `<`*`CONNECTORNAME`*`>/SYNCHRONOUSRESPONSEQUEUE`

TivoliMonitorTransactionPerformance

The TivoliMonitorTransactionPerformance property specifies whether IBM Tivoli Monitoring for Transaction Performance (ITMTP) is invoked at run time.

The default value is `false`.

WireFormat

The WireFormat property specifies the message format on the transport:

- If the value of the RepositoryDirectory property is a local directory, the value is `CwXML`.
- If the value of the RepositoryDirectory property is a remote directory, the value is `CwBO`.

WsifSynchronousRequestTimeout

The WsifSynchronousRequestTimeout property specifies the time in milliseconds that the connector waits for a response to a synchronous request. If the response is not received within the specified time, the connector moves the original synchronous request message (and an error message) to the fault queue.

This property is valid only if the value of BrokerType is `WAS`.

The default value is 0.

XMLNameSpaceFormat

The XMLNameSpaceFormat property specifies short or long namespaces in the XML format of business object definitions.

This property is valid only if the value of BrokerType is set to WMQI or WAS.

The default value is `short`.

Appendix B. Connector specific properties

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

The following table lists the connector-specific configuration properties for the connector. See the sections that follow for explanations of the properties.

Table 16. Connector-specific configuration properties

Name	Meaning	Possible values	Default value
ApplicationPassword	Password for the Siebel user account		`CWCONN`
ApplicationUserName	User account for the Siebel application		`CWCONN`
ArchiveProcessed	Specifies whether the connector archives events for which there are no current subscriptions.	True or False	True
ConnectErrors	A set of errors returned from Siebel which are checked in the connector. These errors are considered to be fatal, and the connector is terminated when it encounters these errors.	Any network failure or connectivity failure messages. These messages are separated by the ';' delimiter.	
ConnectorID	Used in case the system has been configured to handle multiple connectors.	An integer value denoting the connector.	
ConnectString	A string used by the Siebel Java Data Bean to connect to the Siebel Object manager.	`protocol://machinename/enterprisename/objectmanager/servername`	None
ConnectString (for Siebel, version 7.5)	A string used by the Siebel Java Data Bean to connect to the Siebel Object manager.	`protocol://machinename/enterprisename/objectmanager/servername`	None
ConnectString (for Siebel, version 7.7)	A string used by the Siebel Java Data Bean to connect to the Siebel Object manager.	`protocol://machinename:portno/enterprisename/objectmanager`	None
DataBeanPoolSize	Indicates the maximum number of beans in the data bean pool.	An integer determining the bean pool size.	
DataBeanRefreshInterval	The value is used to refresh the Siebel data bean resources when the connector is running against Siebel 6.2.x The connector logs off after the requests processed are equal to this value and logs back in.	An integer value indicating the DataBeanRefreshInterval which corresponds to the number of requests to be processed by the connector before a refresh call.	
EventProcessingSupport	Indicates whether the adapter processes the event or not. Can be used to switch off subscription services if necessary.	Boolean	True

Table 16. Connector-specific configuration properties (continued)

Name	Meaning	Possible values	Default value
PollAttributeDelimiter	In case of multiple name-value pairs in the object key, this value determines the delimiter between the keys. If not set, the default is; (semi-colon).	Character	;
PollQuantity	Determines the number of events that gets processed with a `pollForEvents` call.	Integer representing the number of events that gets processed with a `pollForEvents` call.	1
ResonateSupport	Indicates if Resonate has been installed with the Siebel server. The connector bean pool uses Attach/Detach calls (Siebel7) only if Resonate is installed. If not, it logs off after processing a certain number of requests.	Boolean(Logoff from the bean is decided by the DataBeanRefreshInterval) setting.	false
SiebelLanguageCode	Three letter NLS character set code used by Siebel for the languages supported. Default is US English with ENU as the NLS representation.	With Siebel 7, the languages supported with their language codes is listed below: Italian (Std) -- ITA Japanese -- JPN Korean --KOR Norwegian -NOR (Bokmal) Polish -- POL Portuguese -- PTB (Brazil) Portuguese -- PTG (Portugal) Russian - RUS Spanish -- ESN (Modern Sort) Swedish -- SVE Turkish -- TUR English (US) -- ENU + all the other languages supported by NLS	
SiebelVersion	Allows the adapter to run against a specified version of Siebel without accessing the SchemeVersion Siebel business object to obtain the version. Use of the default value is recommended.	6, 7, or NONE	NONE
SupportNameValuePair	Used for determining the event object key format. If not set or if set to true, the object key value needs to be a name-value pair with an "=" between the name and the value. If set to false, only one rowId can be specified.	True or False	False
UseDefaults	For create operations, determines whether the connector checks for a valid value or a default value for each required business object attribute.	True or False	False

Table 16. Connector-specific configuration properties (continued)

Name	Meaning	Possible values	Default value
ViewMode	Retained for backward compatibility. An integer value that determines the permissions of the user. The value specified for this property is used unless a VM asi tag is specified at the business object level.	An integer value. Refer to VM asi for details.	

ApplicationPassword

Password for the application user account.

There is no default value.

ApplicationUserName

Name of the application user account.

There is no default value.

ArchiveProcessed

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

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

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

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

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

If this property has no value, the connector assumes the value to be true.

The default value is `true`.

ConnectErrors

Connectivity errors returned from Siebel. When the connector encounters these errors, it terminates.

ConnectorID

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

Default value is null.

ConnectString

A string used by the Siebel Java Data Bean to connect to the Siebel Object Manager.

The value that you set depends on the version of Siebel that you are using. There is no default value for this property.

DataBeanPoolSize

An integer that indicates the maximum number of beans in the data bean pool.

DataBeanRefreshInterval

An integer value that indicates the number of requests to be processed by the connector before a call to refresh the Siebel data bean resources. Used by the connector when it runs against Siebel 6.x.

EventProcessingSupport

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

The default value is `true`.

PollAttributeDelimiter

When multiple name-value pairs are used in the object key column of the event table, this character value determines the delimiter between the keys.

If it is not set, the default is **;**.

PollQuantity

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

The default is 1.

ResonateSupport

Used with Siebel 7.x. A Boolean value that indicates whether Resonate support is installed with the Siebel server. When the connector runs against Siebel 7.x and ResonateSupport is set to true, it uses this property in conjunction with the value set for the DataBeanRefreshInterval property to determine logoff from the data bean pool.

If ResonateSupport is set to `true,` the connector uses Attach and Detach calls to attach and detach from an existing session after each request has been processed. If ResonateSupport is set to `false`, the connector logs of after processing a certain number of requests.

The default setting is `false`.

SiebelLanguageCode

Three letter NLS character set code used by Siebel for the languages supported. Default is US English with enu as the NLS representation.

SiebelVersion

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

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

SupportNameValuePair

Used to determine the event object key format. If this is set to true, or if it is not set, the object key value must be a name-value pair with an `"="` between name and value.

If this is set to false, only one rowID can be specified in the object key. Multiple keys are not supported.

The default setting is true.

UseDefaults

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

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

The default value is `false`.

ViewMode

An integer that determines the access permissions of the user. If a ViewMode application specific information tag is not specified at the business object level, then the value specified in the connector properties will be used.

Appendix C. Common Event Infrastructure

WebSphere Business Integration Server Foundation includes the Common Event Infrastructure Server Application, which is required for Common Event Infrastructure to operate. The WebSphere Application Server Foundation can be installed on any system (it does not have to be the same machine on which the adapter is installed.)

The WebSphere Application Server Application Client includes the libraries required for interaction between the adapter and the Common Event Infrastructure Server Application. You must install WebSphere Application Server Application Client on the same system on which you install the adapter. The adapter connects to the WebSphere Application Server (within the WebSphere Business Integration Server Foundation) by means of a configurable URL.

Common Event Infrastructure support is available using any integration broker supported with this release.

Required software

In addition to the software prerequisites required for the adapter, you must have the following installed for Common Event Infrastructure to operate:

- WebSphere Business Integration Server Foundation 5.1.1
- WebSphere Application Server Application Client 5.0.2, 5.1, or 5.1.1.

  (WebSphere Application Server Application Client 5.1.1 is provided with WebSphere Business Integration Server Foundation 5.1.1.)

Note: Common Event Infrastructure is not supported on any HP-UX or Linux platform.

Enabling Common Event Infrastructure

Common Event Infrastructure functionality is enabled with the standard properties `CommonEventInfrastructure` and `CommonEventInfrastructureContextURL`, configured with Connector Configurator. By default, Common Event Infrastructure is not enabled. The `CommonEventInfrastructureContextURL` property enables you to configure the URL of the Common Event Infrastructure server.(Refer to the "Standard Properties" appendix of this document for more information.)

Obtaining Common Event Infrastructure adapter events

If Common Event Infrastructure is enabled, the adapter generates Common Event Infrastructure events that map to the following adapter events:

- Starting the adapter
- Stopping the adapter
- An application response to a timeout from the adapter agent
- Any `doVerbFor` call issued from the adapter agent
- A `gotApplEvent` call from the adapter agent

For another application (the "consumer application") to receive the Common Event Infrastructure events generated by the adapter, the application must use the

Common Event Infrastructure event catalog to determine the definitions of appropriate events and their properties. The events must be defined in the event catalog for the consumer application to be able to consume the sending application's events.

The "Common Event Infrastructure event catalog definitions" appendix of this document contains XML format metadata showing, for WebSphere Business Information adapters, the event descriptors and properties the consumer application should search for.

For more information

For more information about Common Event Infrastructure, refer to the Common Event Infrastructure information in the WebSphere Business Integration Server Foundation documentation, available at the following URL:

http://publib.boulder.ibm.com/infocenter/ws51help

For sample XML metadata showing the adapter-generated event descriptors and properties a consumer application should search for, refer to"Common Event Infrastructure event catalog definitions."

Common Event Infrastructure event catalog definitions

The Common Event Infrastructure event catalog contains event definitions that can be queried by other applications. The following are event definition samples, using XML metadata, for typical adapter events. If you are writing another application, your application can use event catalog interfaces to query against the event definition. For more information about event definitions and how to query them, refer to the Common Event Infrastructure documentation that is available from the online IBM WebSphere Server Foundation Information Center.

For WebSphere Business Integration adapters, the extended data elements that need to be defined in the event catalog are the keys of the business object. Each business object key requires an event definition. So for any given adapter, various events such as start adapter, stop adapter, timeout adapter, and any `doVerbFor` event (create, update, or delete, for example) must have a corresponding event definition in the event catalog.

The following sections contain examples of the XML metadata for start adapter, stop adapter, and event request or delivery.

XML format for "start adapter" metadata

```
<eventDefinition name="startADAPTER"
      parent="event">
    <property name ="creationTime" //Comment: example value would be
 "2004-05-13T17:00:16.319Z"
         required="true" />
    <property name="globalInstanceId" //Comment: Automatically generated
 by Common Event Infrastructure
         required="true"/>
    <property name="sequenceNumber"    //Comment: Source defined number
for messages to be sent/sorted logically
         required="false"/>
    <property name="version"    //Comment: Version of the event
         required="false"
         defaultValue="1.0.1"/>
    <property name="sourceComponentId"
         path="sourceComponentId"
         required="true"/>
    <property name="application"   //Comment: The name#version of the
source application generating the event. Example is "SampleConnector#3.0.0"
         path="sourceComponentId/application"         required="false"/>
    <property name="component"   //Comment: This will be the name#version
 of the source component.
         path="sourceComponentId/component"
         required="true"
         defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"    //Comment: specifies the format
and meaning of the component
          path="sourceComponentId/componentIdType"
          required="true"
          defaultValue="Application"/>
    <property name="executionEnvironment"
 //Comment: Identifies the environment the application is running
 in...example is "Windows 2000#5.0"
          path="sourceComponentId/executionEnvironment"
          required="false" />
     <property name="location"    //Comment: The value of this is the
 server name...example is "WQMI"
          path="sourceComponentId/location"
          required="true"/>
    <property name="locationType"   //Comment specifies the format and
     meaning of the location
          path="sourceComponentId/locationType"
          required="true"
          defaultValue="Hostname"/>
     <property name="subComponent"     //Comment:further distinction
of the logical component
          path="sourceComponentId/subComponent"
          required="true"
          defaultValue="AppSide_Connector.AgentBusinessObjectManager"/>
     <property name="componentType"      //Comment: well-defined name
used to characterize all instances of this component
          path="sourceComponentId/componentType"
          required="true"
          defaultValue="ADAPTER"/>
     <property name="situation"   //Comment: Defines the type of
 situation that caused the event to be reported
          path="situation"
          required="true"/>
     <property name="categoryName="    //Comment: Specifies the type
of situation for the event
         path="situation/categoryName"
         required="true"
         defaultValue="StartSituation"/>
    <property name="situationType"    //Comment: Specifies the type
of situation and disposition of the event
         path="situation/situationType"
         required="true"
    <property name="reasoningScope" //Comment: Specifies the scope
 of the impact of the event
         path="situation/situationType/reasoningScope"
         required="true"
         permittedValue="INTERNAL"
         permittedValue="EXTERNAL"/>
    <property name="successDisposition" //Comment: Specifies the
 success of event
         path="situation/situationType/successDisposition"
         required="true"
         permittedValue="SUCCESSFUL"
         permittedValue="UNSUCCESSFUL" />
    <property name="situationQualifier"  //Comment: Specifies the
 situation qualifiers for this event
```

```
         path="situation/situationType/situationQualifier"
         required="true"
         permittedValue="START_INITIATED"
         permittedValue="RESTART_INITIATED"
         permittedValue="START_COMPLETED" />
</eventDefinition>
```

XML format for ″stop adapter″ metadata

The metadata for "stop adapter" is the same as that for "start adapter" with the following exceptions:

- The default value for the `categoryName` property is `StopSituation`:

  ```
  <property name="categoryName="
   //Comment: Specifies the type
   of situation for the event
               path="situation/categoryName"
               required="true"
               defaultValue="StopSituation"/>
  ```

- The permitted values for the `situationQualifier` property differ and are as follows for "stop adapter":

  ```
  <property name="situationQualifier"
   //Comment: Specifies the situation qualifiers for this event
            path="situation/situationType/situationQualifier"
            required="true"
            permittedValue="STOP_INITIATED"
            permittedValue="ABORT_INITIATED"
            permittedValue="PAUSE_INITIATED"
            permittedValue="STOP_COMPLETED"
   />
  ```

XML format for "timeout adapter" metadata

The metadata for "timeout adapter" is the same as that for "start adapter" and "stop adapter" with the following exceptions:

- The default value for the `categoryName` property is `ConnectSituation`:

  ```
  <property name="categoryName="
   //Comment: Specifies the type
   of situation for the event
               path="situation/categoryName"
               required="true"
               defaultValue="ConnectSituation"/>
  ```

- The permitted values for the `situationQualifier` property differ and are as follows for "timeout adapter":

  ```
  <property name="situationQualifier"  //Comment: Specifies
   the situation qualifiers for this event
            path="situation/situationType/situationQualifier"
            required="true"
            permittedValue="IN_USE"
            permittedValue="FREED"
            permittedValue="CLOSED"
            permittedValue="AVAILABLE"
   />
  ```

XML format for ″request″ or ″delivery″ metadata

At the end of this XML format are the extended data elements. The extended data elements for adapter request and delivery events represent data from the business object being processed. This data includes the name of the business object, the key (foreign or local) for the business object, and business objects that are children of parent business objects. The children business objects are then broken down into the same data as the parent (name, key, and any children business objects). This data is represented in an extended data element of the event definition. This data will change depending on which business object, which keys, and which child business objects are being processed. The extended data in this event definition is just an example and represents a business object named `Employee` with a key `EmployeeId` and a child business object `EmployeeAddress` with a key `EmployeeId`. This pattern could continue for as much data as exists for the particular business object.

```
<eventDefinition name="createEmployee"     //Comment: This
 extension name is always the business object verb followed by the business
 object name
          parent="event">
    <property name ="creationTime"  //Comment: example value would be
"2004-05-13T17:00:16.319Z"
         required="true" />
    <property name="globalInstanceId" //Comment: Automatically generated
 by Common Event Infrastructure
         required="true"/>
    <property name="localInstanceId"    //Comment: Value is business
 object verb+business object name+#+app name+ business object identifier
         required="false"/>
    <property name="sequenceNumber"    //Comment: Source defined number
for messages to be sent/sorted logically
         required="false"/>
    <property name="version"  //Comment: Version of the event...value is
 set to 1.0.1
         required="false"
         defaultValue="1.0.1"/>
    <property name="sourceComponentId"
         path="sourceComponentId"
         required="true"/>
    <property name="application"    //Comment: The name#version of the
 source application generating the event...example is
 "SampleConnector#3.0.0"
         path="sourceComponentId/application"
         required="false"/>
    <property name="component"   //Comment: This will be the name#version
of the source component.
         path="sourceComponentId/component"
         required="true"
         defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"     //Comment: specifies the format
 and meaning of the component
         path="sourceComponentId/componentIdType"
         required="true"
         defaultValue="Application"/>
    <property name="executionEnvironment" //Comment: Identifies the
 environment#version the app is running in...example is "Windows 2000#5.0"
         path="sourceComponentId/executionEnvironment"
         required="false" />
    <property name="instanceId" //Comment: Value is business object
  verb+business object name+#+app name+ business object identifier
          path="sourceComponentId/instanceId"
          required="false"
    <property name="location"   //Comment: The value of this is the
server name...example is "WQMI"
          path="sourceComponentId/location"
          required="true"/>
     <property name="locationType" //Comment specifies the format and
 meaning of the location
          path="sourceComponentId/locationType"
          required="true"
          defaultValue="Hostname"/>
     <property name="subComponent"  //Comment:further distinction of the
 logical component-in this case the value is the name of the business
 object
          path="sourceComponentId/subComponent"
          required="true"/>
     <property name="componentType"      //Comment: well-defined name used
 to characterize all instances of this component
          path="sourceComponentId/componentType"
          required="true"
          defaultValue="ADAPTER"/>
     <property name="situation" //Comment: Defines the type of
situation that caused the event to be reported
          path="situation"
          required="true"/>
    <property name="categoryName"    //Comment: Specifies the type
 of situation for the event
         path="situation/categoryName"
         required="true"
         permittedValue="CreateSituation"
         permittedValue="DestroySituation"
         permittedValue="OtherSituation" />
    <property name="situationType"    //Comment: Specifies the type
of situation and disposition of the event
         path="situation/situationType"
         required="true"
    <property name="reasoningScope" //Comment: Specifies the scope
of the impact of the event
         path="situation/situationType/reasoningScope"
         required="true"
         permittedValue="INTERNAL"
         permittedValue="EXTERNAL"/>
    <property name="successDisposition" //Comment: Specifies the
 success of event
         path="situation/situationType/successDisposition"
         required="true"
         permittedValue="SUCCESSFUL"
         permittedValue="UNSUCCESSFUL" />
    <extendedDataElements name="Employee" //Comment: name of business
 object itself
             type="noValue"
             <children name="EmployeeId"
                  type="string"/>  //Comment: type is one of the
 permitted values within Common Event Infrastructure documentation
             <children name="EmployeeAddress"
                  type="noValue"/>
                     <children name="EmployeeId"
                         type="string"/>
                      -
                      -
                      -
    </extendedDataElements
</eventDefinition>
```

Appendix D. Application Response Management

This adapter is compatible with the Application Response Measurement application programming interface (API), an API that allows applications to be managed for availability, service level agreements, and capacity planning. An ARM-instrumented application can participate in IBM Tivoli Monitoring for Transaction Performance, allowing collection and review of data concerning transaction metrics.

Application Response Measurement instrumentation support

This adapter is compatible with the Application Response Measurement application programming interface (API), an API that allows applications to be managed for availability, service level agreements, and capacity planning. An ARM-instrumented application can participate in IBM Tivoli Monitoring for Transaction Performance, allowing collection and review of data concerning transaction metrics.

Required software

In addition to the software prerequisites required for the adapter, you must have the following installed for ARM to operate:

- WebSphere Application Server 5.0.1 (contains the IBM Tivoli Monitoring for Transaction Performance server). This does not have to be installed on the same system as the adapter.
- IBM Tivoli Monitoring for Transaction Performance v. 5.2 Fixpack 1. This must be installed on the same system on which the adapter is installed and configured to point to the system on which the IBM Tivoli Monitoring for Transaction Performance server resides.

Application Response Measurement support is available using any integration broker supported with this release.

Note: Application Response Measurement instrumentation is supported on all operating systems supported with this IBM WebSphere Business Integration Adapters release *except* HP-UX (any version) and Red Hat Linux 3.0.

Enabling Application Response Measurement

ARM instrumentation is enabled via by setting the standard property `TivoliMonitorTransactionPerformance` in Connector Configurator to “True.” By default ARM support is not enabled. (Refer to the ″Standard Properties″ appendix of this document for more information.)

Transaction monitoring

When ARM is enabled, the transactions that are monitored are service events and event deliveries. The transaction is measured from the start of a service request or event delivery to the end of the service request or event delivery. The name of the transaction displayed on the Tivoli Monitoring for Transaction Performance console will start with either `SERVICE REQUEST` or `EVENT DELIVERY`. The next part of the name will be the business object verb (such as `CREATE`, `RETRIEVE`, `UPDATE` or `DELETE`). The final part of the name will be the business object name such as “`EMPLOYEE`.”

For example, the name of a transaction for an event delivery for creation of an employee might be `EVENT DELIVERY CREATE EMPLOYEE`. Another might be `SERVICE REQUEST UPDATE ORDER`.

The following metrics are collected by default for each type of service request or event delivery:
- Minimum transaction time
- Maximum transaction time
- Average transaction time
- Total transaction runs

You (or the system administrator of the WebSphere Application Server) can select which of these metrics to display, for which adapter events, by configuring Discovery Policies and Listener Policies for particular transactions from within the Tivoli Monitoring for Transaction Performance console. (Refer to “For more information.”)

For more information

Refer to the IBM Tivoli Monitoring for Transaction Performance documentation for more information. In particular, refer to the *IBM Tivoli Monitoring for Transaction Performance User's Guide* for information about monitoring and managing the metrics generated by the adapter.

Index

Notices

Programming interface information

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

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

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

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

Trademarks and service marks

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
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
IMS
Informix
iSeries
Lotus
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS

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.

Linux is a trademark of Linus Torvalds 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 V2.6.0

Printed in USA