IBM WebSphere Business Integration Adapters

Adapter for MetaSolv Applications User Guide

Version 2.8.x

IBM WebSphere Business Integration Adapters

Adapter for MetaSolv Applications User Guide

Version 2.8.x

Note!
Before using this information and the product it supports, read the information in "Notices" on page 73.

25June2004

This edition of this document applies to IBM WebSphere InterChange Server, version 4.2.2, WebSphere Business Integration Adapters, version 2.2.0, and to all subsequent releases and modification until otherwise indicated in new editions.

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

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

Contents

About this document

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

This document describes the installation, configuration, business object development, and troubleshooting for the IBM WebSphere Business Integration Adapter for MetaSolv(R) Applications.

Audience

This document is for WebSphere consultants and customers who are implementing the connector as part of a WebSphere business-integration system. To use the information in this document, you should be knowledgeable in the following areas:

- Connector development
- Business object development
- MetaSolv application architecture

Related documents

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

You can install related documentation from the following sites:

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

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 may be available in Technical Support Technotes and Flashes issued after this document was published. These can be found on the WebSphere Business Integration Support Web site,

http://www.ibm.com/software/integration/websphere/support/

Select the component area of interest and browse the Technotes and Flashes sections.

Typographic conventions

This document uses the following conventions:

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

New in this release

Version 2.8.x

- A gateway server for PSR and LSR modules has been added
- The `mail.jar` and the `activation.jar` have been removed from the installation

Prior releases

Below are the features and changes in prior versions.

Version 2.7.x

The IBM WebSphere Business Integration Adapter for MetaSolv now supports business object processing and event publishing for the PSR (Product Service Requests), LSR (Local Service Request), and DLR (Design Layout Report) modules in MetaSolv version 5.0.3.

The MetaSolv Object Discovery Agent (ODA) has been added and the sample business object has been removed as the ODA creates business objects based on a template file.

Beginning with the 2.7 version, the adapter for MetaSolv is no longer supported on Microsoft Windows NT.

Adapter installation information has been moved from this guide. See Chapter 2 “Installing the MetaSolv adapter and other files” on page 6 for the new location of that information.

Version 2.6.x

A new connector-specific configuration property, “ORB.<property name>” on page 13 has been added to be used by the connector to initialize the ORB.

A complete sample MetaSolv business object has been added to provide a model for developing custom business objects.

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

Version 2.5.x

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

Due to testing limitations, only the following APIs are being supported in the current release of MetaSolv:

- exportCustomerAccount
- exportCustomerAccounts
- importNewCustomerAccount

- importUpdatedCustomerAccount
- exportServiceLocations
- exportCustServiceLocations
- exportAllServiceLocations
- importNewServiceLocation
- import Updated Service Location

Version 2.4.x

The changes to version 2.4.x of the connector do not affect the content of this document.

Version 2.2.x

The IBM WebSphere Business Integration Adapter for MetaSolv includes the connector for MetaSolv. 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 MetaSolv
- A sample business object, included in the `\connectors\MetaSolv\samples` directory
- 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.

Version 2.1.x

The changes made for version 2.1.0 of the connector do not affect the content of this document.

Version 2.0.x

Version 2.0.0 of the connector includes the following new features and changes:

- The connector now supports MetaSolv 5.0 API.
- The following connector properties have been removed: OrbixClasses, ConnectorClasses, APIClasses, APIHost, APIServer, SQLNetConnectString.
- New connector properties have been added: MSLVPSRIORfile, JDBCDriverClass, DatabaseURL.
- Oracle thin driver is the default JDBC driver to connect to the IBM CrossWorlds database where event and archive tables reside.

Version 1.1.x

Version 1.1.0 of the connector includes the following new features and changes:

- The connector now supports event notification through the MetaSolv API for the PSR OrderEntry module.

Chapter 1. Overview of the MetaSolv connector

This chapter describes the connector component of the IBM WebSphere Business Integration Adapter for MetaSolv. The connector enables an integration broker to exchange business objects with MetaSolv 5.2.

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

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

This document contains information about the application-specific component and connector framework. It refers to both of these components as the connector.

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

This chapter covers the following topics:

Architecture

Figure 1 shows the connector components and their relationships within the WebSphere business integration system. The figure assumes that WebSphere InterChange Server is being used as the integration broker.

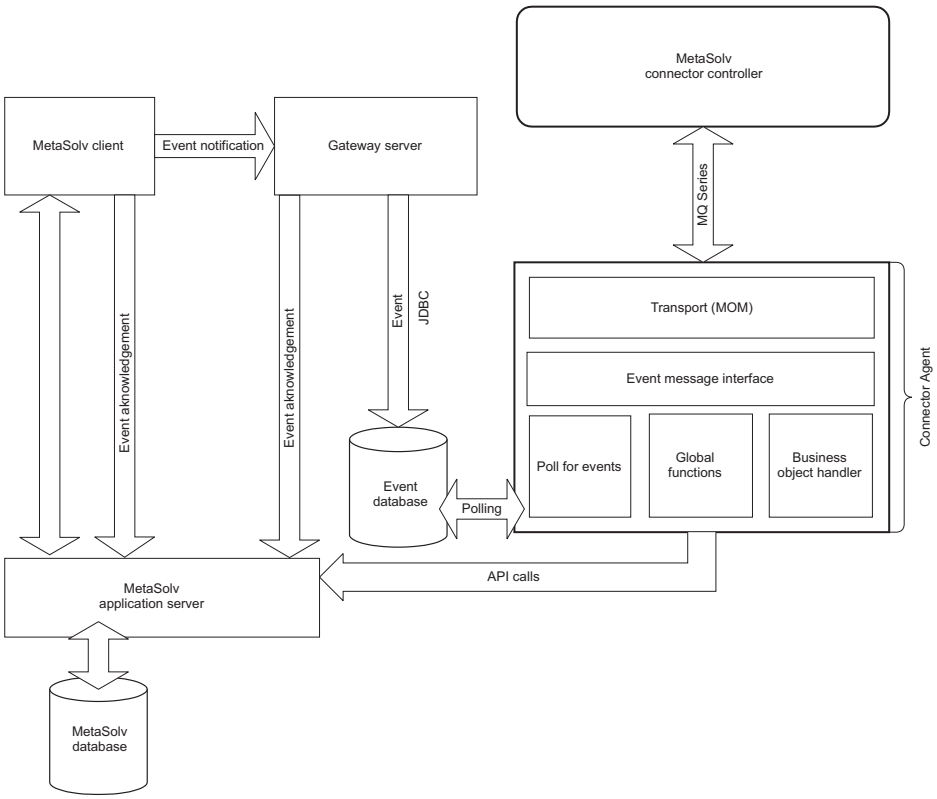

Figure 1. Architecture of the connector (with WebSphere InterChange Server as the integration broker)

Business object request processing

The connector processes business object requests that it receives from the integration broker. When the connector receives a request to perform an application operation, it uses the Business Object Handler to extract the data from the application-specific business object, and communicates with the MetaSolv application by using the API provided by MetaSolv. The connector uses adapter-delivered meta-objects, business object definitions, attribute descriptions, and the values in a business object instance to generate MetaSolv API calls.

Event notification

Event Notification in the connector is handled through the adapter-delivered Gateway Server. An event is triggered when a task is initiated in "Waiting" status. The MetaSolv Client sends the event to the IBM Gateway Server depending on the type of event. The Gateway Server reads the event details from the Event Notification object and places the event in the business-integration system's event table. The event tables can be created in any Oracle database.

Gateway server

Gateway Servers are responsible for extracting event information from the MetaSolv Client event notification and creating an event in the event table. There are four adapter-delivered Gateway Servers:

- `CwDLRGatewayServer` is the gateway server handles the gateway events for DLR.

- `CwLSRGatewayServer` is the gateway server that handles the gateway events for LSR.
- `CwPSRGatewayServer` handles the Gateway events related to Service Requests.
- `CwPSRBillingGatewayServer` handles the Customer related events for the billing system

These servers should run on the same machine as the MetaSolv Application Server is running.

An event is triggered when a MetaSolv task gets initiated. The MetaSolv Client then sends Event to the appropriate Gateway Server. When the event is successfully populated in the event table, a status of `completed` is returned to the MetaSolv Client via the MetaSolv Application Server.

If the event does not get populated, a status of `error` is be returned to the MetaSolv Client. If the error is related to the WebSphere business-integration system's database, a notification e-mail is sent. The e-mail address must be specified as a parameter in the `runCwPSRGatewayServer.bat`, `runCwPSRBillingGatewayServer.bat`, `runCwLSRGatewayServer.bat`, and `runCwDLRRGatewayServer.bat` files.

If the MetaSolv Application Server is down, the status of the gateway events is `Waiting` until the MetaSolv Application Server is brought up.

The MetaSolv Client must be restarted Whenever the Gateway Servers are restarted. MetaSolv Client gets a handle to the Gateway Server when it first sends the Gateway event. When a Gateway Server is restarted, the old handle is no longer valid.

Note: Any change to the WebSphere business-integration system's event table definition affect the Gateway servers.

Processing application events

Event retrieval

The connector polls the event table at configurable intervals. The connector picks up all unprocessed events in the event table, processes them, and then archives them. The number of events a connector can process at any time is determined by the `PollQuantity` connector property. The connector builds a query to extract events from the event table.

Each time the connector is started, it updates the `InProgress` events to `ReadyForPoll` and sets their priority to 0. This ensures that the status is updated for `InProgress` events in case the connector is abnormally shut down.

Create notification

When the connector encounters a Create event in the event table, it instantiates an object of the type specified by the event, sets the doc number for the object, and retrieves it through the MetaSolv Application Server APIs. When it is retrieved, the object is sent to the integration broker.

Update notification

When the connector encounters an Update event in the event table, it instantiates an object of the specified type, sets the doc number for the object, and retrieves it by using the MetaSolv APIs. When it is retrieved, the object is sent to the integration broker.

Archiving

When an event is deleted from the event table it is placed in the archive table. The archive table provides a means of recovering events that are not successfully processed, and a history of events that can be used to audit event processing.

To enable archiving, the `ArchiveProcessed` must be set to `true`. If the `ArchiveProcessed` property is set to `false`:

- If the event is successfully processed, it is deleted from the event table and without being archived.
- If the object in the event is not subscribed, the event remains in the event table with a status of `Unsubscribed`.
- If the object faces a problem while being processed it remains in the event table with event status set to `Ready for Poll`.

The `ArchiveProcessed` property defines whether unsubscribed or unprocessed events are archived. The archive table is specified through the configuration property `ArchiveTableName`.

Supported verbs and business processes

The connector supports `Create`, `Update` and `Retrieve` for Event Notification and Business Object Request processing.

The connector supports all PSR, LSR, and DLR business processes.

The connector also delivers a `CwPSRBilling` gateway server to fulfil the requirement of a billing business process from MetaSolv to an external billing system.

Chapter 2. Installing and configuring the adapter

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

Compatibility

The adapter framework that an adapter uses must be compatible with the version of the integration broker (or brokers) with which the adapter is communicating. Version 2.8 of the adapter for MetaSolv Applications is supported on the following versions of the adapter framework and with the following integration brokers:

Adapter framework: WebSphere Business Integration Adapter Framework versions 2.1, 2.2, 2.3.x, and 2.4.

Integration brokers:

- WebSphere InterChange Server, versions 4.2.x (if the environment uses only ISO Latin-1 data)
- WebSphere MQ Integrator, version 2.1.0
- WebSphere MQ Integrator Broker, version 2.1.0
- WebSphere Business Integration Message Broker, version 5.0
- WebSphere Application Server Enterprise, version 5.0.2, with WebSphere Studio Application Developer Integration Edition, version 5.0.1

See the Release Notes for any exceptions.

Note: For instructions on installing the integration broker and its prerequisites, see the following documentation. For WebSphere InterChange Server (ICS), see the System Installation Guide for UNIX or for Windows.

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

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

Prerequisites

MetaSolv application requirements

The adapter supports business object processing and event publishing for the PSR (Product Service Requests), LSR is (Local Service Request), and DLR (Design Layout Report) modules in MetaSolv version 5.2.x.

In this document it is assumed that MetaSolv has been properly installed and configured with its prerequisite software and that there is a functioning MetaSolv Application Server.

For more information about installing MetaSolv and its prerequisite software, consult the MetaSolv application documentation.

Adapter third-party dependencies

The adapter uses JBroker, the Java version of CORBA from SilverStream. The following is the key requirement for running the connector:

- Complete installation of the MetaSolv Application Server

Installing the MetaSolv adapter and other files

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

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

Third party jar files

The jar files listed in this section must be copied from MetaSolv installation directory to the *ProductDir*\connectors\MetaSolv\dependencies folder.

Note: The path following the file name is the location where the file is present in MetaSolv installation.

- appserver.jar - *%METASOLV%*\appserver\classes
- jbroker_stubs.jar - *%METASOLV%*\appserver\classes
- jbroker-rt.jar - *%METASOLV%*\appserver\jre\lib\ext
- jbroker-tools.jar - *%METASOLV%*\appserver\jre\lib\ext
- jbroker-ssl.jar - *%METASOLV%*\appserver\jre\lib\ext
- mail.jar - *%METASOLV%*\AppServer\lib
- activation.jar - *%METASOLV%*\AppServer\lib

Installing and configuring Gateway servers and databases

The CwPSRGatewayServer, CwDLRGatewayServer, CwLSRGatewayServer, and CwPSRBillingGatewayServer are the gateway event handling servers. They must run on the same machine where the MetaSolv Application Server is running.

To set up the Gateway Servers:

1. On the MetaSolv Application Server machine, create the following directories:
   - *ProductDir*
   - *ProductDir*\GWDependencies
   - *ProductDir*\GWServer
2. Copy the following files from *%ProductDir%*\Connectors\MetaSolv\dependencies to the GWServer subdirectory on the MetaSolv Application Server machine:
   - CwPSRGatewayServer.jar
   - RunCwPSRGatewayServer.bat
   - CwPSRBillingGatewayServer.jar
   - RunCwPSRBillingGatewayServer.bat
   - CwDLRGatewayServer.jar
   - RunCwDLRGatewayServer.bat
   - CwLSRGatewayServer.jar
   - RunCwLSRGatewayServer.bat
3. Start the MetaSolv Application Server.
4. Set up event and archive tables:
   a. Create an Oracle database instance for the event and archive tables. The size of the database depends on your event and archive volume. A minimum of 2MB is recommended.
   b. Locate the Event_Table.sql script files in the *%ProductDir%*\connectors\MetaSolv\dependencies directory.
   c. Run the script in your new Oracle database instance to create event and archive tables.

   Table 1 describes the event and archive table schema.

Table 1. Event and archive table schema

Name	Description	Type	Constraint
event_id	Internal identifier of the event	NUMBER	Primary key
connector_id	Unique ID of the connector for which the event is destined. This value is important when multiple connectors poll the same table	NUMBER	
object_key	Primary key of the business object. Multiple keys can be concatenated with a colon or other configurable delimiter, for example, 1000065:10056:23333	VARCHAR	Not null
object_name	Name of the Business Object	STRING	Not null
object_verb	verb associated with the event	STRING	Not null

Table 1. Event and archive table schema (continued)

Name	Description	Type	Constraint
event_priority	Event priority (0 is the highest, n is the lowest); which the connector uses to pull events on a priority basis. The connector does not use this value to lower or raise priorities.	NUMBER	Not null
event_time	Date and time when the event occurred	DATE	Default current date/time (for archive table, actual event time)
event_status	This attribute is used only in the event table. 0 - Ready for poll 1 - Sent to InterChange Server 2 - Unsubscribed event 3 - In progress -1 = Error processing event -2 = Error sending event to the integration broker	NUMBER	Not null
event_comment	Name of the tasks that have been completed	STRING	
archive_time	Date and time when the event was archived (applies only to the archive table)	DATE	Archive date/time

5. Locate `runcwPSRgatewayserver.bat`, `runcwDLRgatewayserver.bat`, `runcwLSRgatewayserver.bat`, and `runcwpsrbillinggatewayserver.bat` in the *%ProductDir%*`\connectors\MetaSolv\dependencies` directory. Edit the files to include the required path information and parameters as indicated in the file comments.

   The following example illustrates how to set the parameters in the `.bat` files. Remember to set the directories specific to your environment. Do not delete the comments provided in the `.bat` files. Table 2 on page 9 describes the parameters in the Gateway Server batch files.

```
setlocal
set PATH=%PATH%
set CROSSWORLDS=path to the CrossWorlds directory on the API Server
machine which was created earlier.

REM name of the directory which contains the appserver.jar,
jbroker.jar, jbroker_stubs.jar,
JBrokerLicenses.class, mail.jar and activation.jar
set DEPENDENCIES=xxxx

REM name of the directory which contains the jar of the gateway server
set SOURCE_DIR=xxxx

REM USER_NAME is the Event table user name
set USER_NAME=system

REM USER_NAME is the Event table password
set PASSWORD=manager

REM DATABASE_URL is the URL for the event table database
set DATABASE_URL=xxxx

REM EVENT_TABLE is the name of the Event table
set EVENT_TABLE=xworlds_events

REM GATEWAY_IOR is the name of the CrossWorlds gateway server IOR file
set GATEWAY_IOR=CwGatewayServer.ior

REM name of the MetaSolv server file
set MSVLV_IOR=PSRServer.ior
```

```
REM name of the EUB server file for MetaSolv
set MSLV_PSR_EVBIOR=PSREUBServer.IOR

REM EMail ID of the database administrator
set EMAIL_ID=xxxx

REM INI_FILE_NAME is the name of the INI file which has the events names
and the BO names
set INI_FILE_NAME=GatewayEvents.ini

REM MAIL_SERVER is the name of the Mail Server
set MAIL_SERVER=xxxx
```

6. Save and close the files.
7. Run `runcwPSRgatewayserver.bat`, `runcwDLRgatewayserver.bat`, `runcwLSRgatewayserver.bat`, and `runcwpsrbillinggatewayserver.bat` by double-clicking on their file icons.

Table 2. Parameters for Gateway server batch files

Parameter	Required	Example	Description
-u	Yes	-usystem	Event table user name.
-p	Yes	-pmanager	Event table password.
-n	Yes	-nevt	SQL net connect string for event table database name.
-t	Yes	-txworlds_events	Event table name. The default value is `xworlds_events`.
-e	Yes	admin@CW.com	Database Administrator's e-mail ID.
-m	Yes	cage	Mail Server Name.
-i	Yes	GatewayEvents.ini	See "Setting up the .ini file" on page 10 for more information.

Setting up the Gateway servers in the MetaSolv application

Note: Before beginning, make sure the WM gateway Event Processing Preference is enabled.

To set up the Gateway Server:

1. In the Work Management screen, select the Gateways tab.
2. Click the New button on the Gateways tool bar.
3. In the Name field, enter the name of the new gateway.
4. In the Number of Retries field, enter the number of times the gateway event server should attempt to connect with the Gateway Server before setting the event status to error.
5. In the retry interval (secs) field, enter the number of seconds that you want the gateway event server to wait between retries.
6. In the Module field, select the name of the IDL module in which the WDI manager is defined.
7. In the Description field, enter a description of the gateway.
8. In the Username and Password fields, enter the values that the gateway needs to contact the Gateway Server. If no values are needed, leave these fields blank.
9. Save.
10. Right click on your new gateway event and choose a new binding.
11. Select the IOR radio button.

12. Add the location where the gateway event server looks for the IOR file generated by the external application.

Setting up the .ini file

The Gateway Event names are not hardcoded in the MetaSolv application, therefore CwGatewayServer uses an `.ini` file to determine which Business Object to retrieve based on the Gateway Event Name. You must create a text file with the `.ini` extension. This file must reflect the Gateway Event names in your MetaSolv application. The Gateway Event names must be linked to triggering tasks in the MetaSolv application. Changes to the names of Gateway Events in the MetaSolv Client also impact the Gateway Server.

Except the created events `SendBillOrd` and `SendBillCust`, any custom event names must use the following format in MetaSolv:

```
<MSLV Object Name>_<Verb Name>_ <Comment>
```

Where `MSLV_Object_Name` is the MetaSolv object name. The `Verb_Name` could be either Create of Update. Finally, `Comment` can be any string.

Note: The .ini file is not required for the PSR Billing Gateway.

The format of entries in the `.ini` file is *<Gateway_Event_Name><tab><Business_Object_Name>*. For example:

```
PSR_CREATE_DD     MetaSolv_ServiceRequest
```

Testing the Gateway server

Use the following procedure to test the Gateway Server:

1. Start the CwPSRGatewayServer, CwDLRGatewayServer, CwLSRGatewayServer, or the CwPSRBillingGateway server by running the `runcwPSRgatewayserver.bat`, `runcwDLRgatewayserver.bat`, `runcwLSRgatewayserver.bat`, and `runcwpsrbillinggatewayserver.bat` file on the MetaSolv Application Server machine.
2. From the MetaSolv Client, send a Gateway Event that is registered with the server. If the Gateway Server is able to extract the event information and insert the event into the event table, a status of complete is sent.
   - If the event is processed successfully, the server returns a status of "Completed."
   - If the event is errored, the server returns a status of "Errored."
   - If no status is returned, the event is waiting for a response from the server.

Connector configuration

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

A connector obtains its configuration values at startup. During a run-time session, you may want to change the values of one or more connector properties. Changes to some connector configuration properties, such as AgentTraceLevel, take effect immediately. Changes to other connector properties require component restart or system restart after a change. To determine whether a property is dynamic (taking effect immediately) or static (requiring either connector component restart or system restart), refer to your integration broker's administration utility. For

instance, if you are using WebSphere InterChange Server, see the Update Method column in the Connector Properties window of the System Manager.

If you are using WebSphere MQ Integrator Broker as the integration broker, you configure connector properties from the Connector Configurator. If you are using WebSphere InterChange Server as the integration broker, you configure connector properties from Connector Designer, which you access from System Manager.

Standard connector properties

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

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

Note: Because this connector is single threaded, it cannot take advantage of the AgentConnections property.

Adapter-specific properties

Adapter-specific configuration properties provide information needed by the connector at run time. Adapter-specific properties also provide a way of changing static information or logic within the connector agent without having to re-code and rebuild the agent.

Table 3 on page 12 lists the adapter-specific configuration properties for the connector. See the sections that follow for explanations of the properties.

Table 3. Adapter-specific configuration properties

Name	Possible values	Default value	Required
ArchiveProcessed	*true/false*	`true`	
ArchiveTableName	*Name of archive table*	`xworlds_archive_events`	
DatabaseURL	`metasolv:weblogic:oci:metasolv`	No default value	Yes
EventDBBrand	*JDBC driver database type*	Oracle	
EventDBPassword	*Event database password*	asap	Yes
EventDBUsername	*Login to event database username*	asap	Yes
EventTableName	*Name of event table*	xworlds_events	
JDBCDriverClass	*Specifies the class name of the driver weblogic.jdbc.oci.Driver*	No default value	Yes
MaxEvents	*Number of events per poll*	25	
MSLVPassword	*MetaSolv server password*	asap	
MSLVUserName	*Username for MetaSolv server*	asap	No
ORB.<property name>	*The ORB vendor specific properties that would be used by the connector to initialize the ORB.*	com.inprise.vbroker.orb.ORB	Yes
Module-[PSR | LSR | DLR]-IOR	*Name of the MetaSolv server IOR file.*	None	Yes
Module-[PSR | LSR | DLR]	*PSR, LSR, DLR*	None	Yes
Module-[PSR | LSR | DLR]-Manager	*MetaSolv.CORBA.WDI[PSR | LSR | DLR].WDIManager*	None	Yes
Module-[PSR | LSR | DLR]-Session	*MetaSolv.CORBA.WDI[PSR | LSR | DLR].[PSR | LSR | DLR]Session*	None	Yes
Module-[PSR | LSR | DLR]-Root	*MetaSolv.CORBA.WDI[PSR | LSR | DLR].WDIRoot.*	None	Yes
Module- [PSR | LSR | DLR]-Impl	*com.crossworlds.connectors. MetaSolv.WDINotificationImpl*	None	Yes
Module- [PSR | LSR | DLR]-POA	*MetaSolv.CORBA. WDI[[PSR | LSR | DLR].WDINotification*	None	Yes
Module-[PSR | LSR | DLR]-StartMeth	*start[PSR | LSR | DLR]Session*	None	Yes
Module-[PSR | LSR | DLR]-StopMeth	*destroy[PSR | LSR | DLR]Session*	None	Yes

Note: For the `Module-` configuration properties, each set may be implemented with either `PSR`, `LSR`, or `DLR`.

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 table after they are deleted from the event table.

ArchiveTableName

Name of the archive table. The default value is `xworlds_archive_events`.

DatabaseURL

URL String for accessing the WebSphere business-integration system's database. There is no default value. URL should follow the Oracle thin driver's format as the default driver is Oracle thin driver.

EventDBBrand

Database type for the JDBC Driver. The default value is `Oracle`.

EventDBPassword
Password for the login into the event table database. The default value is `asap`.

EventDBUsername
User name for the login into the event table database. The default value is `asap`.

EventTableName
Name of event queue table. The default value is `xworlds_events`.

JDBCDriverClass
Class name of the JDBC driver. Use to connect to the database where event and archive tables reside.

MaxEvents
Maximum number of event processes during one poll. The default value is 25.

MSLVPassword
Password for the login into the MetaSolv server. The default value is `asap`.

MSLVUserName
Username for the login into the MetaSolv server. The default value is `asap`.

ORB.<property name>
The ORB vendor specific properties that would be used by the connector to initialize the ORB. E.g. the property org.omg.CORBA .ORBClass would be named as "ORB. org.omg.CORBA.ORBClass". The number and names of such properties would depend on the ORB specific vendor. All the properties that start with "ORB." will be treated by the connector as orb properties and would be used in ORB initialization.

Module-[PSR | LSR | DLR]-IOR
The name of the MetaSolv server IOR file. The API uses IOR files to route events to/from the MetaSolv application.

Module-[PSR | LSR | DLR]
The specific instance of the module. The module name has to be prefixed with `Module-`

Module-[PSR | LSR | DLR]-Manager
The name of the manager class for the module.

Module-[PSR | LSR | DLR]-Session
The name of the session class for the module.

Module-[PSR | LSR | DLR]-Root
The name of the root class for the module.

Module- [PSR | LSR | DLR]-Impl
The name of the WDINotificationImpl class for the module.

Module- [PSR | LSR | DLR]-POA
The name of the class which is implemented by WDINotificationImpl class.

Module-[PSR | LSR | DLR]-StartMeth
The name of the method that starts a session in a module.

Module-[PSR | LSR | DLR]-StopMeth

The name of the method that stops a session in a module.

Creating multiple connector instances

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

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

Create a new directory

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

```
ProductDir\connectors\connectorInstance
```

where `connectorInstance` uniquely identifies the connector instance.

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

```
ProductDir\repository\connectorInstance
```

Create business object definitions

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

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

   ```
   ProductDir\repository\initialConnectorInstance
   ```

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

Create a connector definition

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

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

Create a start-up script

To create a startup script:

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

   dirname

2. Put this startup script in the connector directory you created in "Create a new directory" on page 14.
3. Create a startup script shortcut (Windows only).
4. Copy the initial connector's shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

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

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

Starting the connector

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

ProductDir\connectors*connName*

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

Table 4. Startup scripts for a connector

Operating system	Startup script
UNIX-based systems	connector_manager_*connName*
Windows	start_*connName*.bat

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

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

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

    start_*connName connName brokerName* [-c*configFile*]
  - On UNIX-based systems:

    connector_manager_*connName* -start

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

  Note: For a WebSphere message broker or WebSphere Application Server on a Windows system, you *must* include the -c option followed by the name of the connector configuration file. For ICS, the -c is optional.

- From Adapter Monitor (WebSphere Business Integration Adapters product only), which is launched when you start System Manager

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

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

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

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

Stopping the connector

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

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

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

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

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

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

Chapter 3. Creating or modifying business objects

The connector is driven primarily by meta-data that is contained in WebSphere Business Integration Adapter business objects and meta-objects.

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

The connector makes assumptions about the structure of its business objects, including:

- The format of the application-specific text.
- The relationships between parent and child business objects.
- The database representation of the business object.

The connector follows rules based on those assumptions, and when you create or modify a business object for MetaSolv, the business object must conform to those rules for the connector to process the business object correctly.

This chapter describes the assumptions the connector makes and how the connector processes business objects. You can use this information as a guide to modifying existing business objects for MetaSolv or as suggestions for implementing new business objects. The following topics are covered:

Creating objects for the connector

To use the connector for PSR (Product Service Requests), LSR (Local Service Request), or DLR (Design Layout Report) modules, you need to create two types of objects:

- Meta-objects, which tell the connector how to send a business object into the MetaSolv API.
- The application-specific business objects for the business processes that you intend to use.

Creating meta-objects

The connector uses the Java.Reflection class to dynamically call MetaSolv APIs to exchange data with the application. To do this, the connector requires meta-objects. Meta-objects contain the information needed to build dynamic calls to the MetaSolv API. The meta-objects must be loaded into the repository, along with the application-specific business objects for MetaSolv.

You must create a meta-object for each top-level application-specific business object and verb that you intend to use. You do not need to create individual meta-objects for child business objects. The meta-object is a flat, non-hierarchical object.

For example, the PSRCustomerAccount business object requires the following meta-objects, one for each of the PSRCustomerAccount verbs:

- MO_MetaSolv_Customer_Create
- MO_MetaSolv_Customer_Update
- MO_MetaSolv_Customer_Retrieve

The meta-objects all have a similar set of attributes and properties. You can use one meta-object as a model for creating all others, with changes to just a few values.

The following example shows the MO_MetaSolv_Customer_Create meta-object, with italics distinguishing the values that you can change if you use this text as a model for another meta-object:

```
[BusinessObjectDefinition]
Name = MO_MetaSolv_Customer_Create
Version = 1.0.0
[Attribute]
Name = methodToCall
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = importNewCustomerAccount
IsRequiredServerBound = false
[End]

[Attribute]
Name = javaClass
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = PSRCustomerAccount
Type = MetaSolv_Connector_Object
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = 7:MetaSolv.CORBA.WDIPSR.data.PSRCustomerAccount
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
```

```
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Update
[End]

[End]
```

Setting property values for meta-objects

The following table describes the attribute properties that have different values for different meta-objects, and how the connector interprets them:

Property	Meaning
Name (For top-level Business Object Definition)	Name of this meta-data business object
AppSpecificInfo (For the methodtoCall attribute)	The name of the MetaSolv API to be called by the connector, such as: *ImportNewCustomerAccount*
Name (For attribute that has Type=MetaSolv_Connector_Object; creates the business object for a Create or Update)	The name of the business object being referenced by this meta-object, such as: *PSRCustomerAccount*
Name (for the primary key in a Retrieve)	A string that is the primary key value for retrieving the business object. For example, for PSRCustomerAccount, the value is: custAcctID
AppspecificInfo (For attribute that has Type=MetaSolv_Connector_Object)	The Java class for the business object being referenced, such as: *7:MetaSolv.CORBA.WDIPSR.data_v2. PSRCustomerAccount*

Creating the customer account and other business objects

This section describes the basic procedure for creating a WebSphere Business Integration Adapter business object that is application-specific for the PSR (Product Service Request) feature of the MetaSolv application.

All MetaSolv PSR business tasks require that a MetaSolv Customer Account data entity exist. Consequently, a WebSphere Business Integration Adapter application-specific business object that corresponds to the Customer Account data entity must be created before any other business objects can be used for PSR tasks.

Creating the business object requires obtaining the PSRTypes_v2.idl file from the MetaSolv web site. After installing the .idl, consult Chapter 4, "The MetaSolv Object Discovery Agent (ODA)," on page 25for detailed information on creating business objects with the MetaSolv ODA.

Using application-specific data

Data transformations: The connector extracts data from a business object in an all-String data format. In that process, the following data transformations take place:

- The connector converts all dates to date structures.

- For Enumerated types, the connector uses the value definitions in the IDL files to convert String values from the business object into valid CORBA enumerations.
- Primary data attributes are grouped together in object and sub-object structures.

The following table shows special considerations for using application-specific data types:

Data type	Comments
String	The String is copied. Single quotation marks are replaced with a space, because the API cannot handle single quotation marks.
Long	If the string representing the attribute value is not of the correct number format, an error message is logged and processing of the business object stops.
Float	If the string representing the attribute value is not of the correct number format, an error message is logged and processing of the business object stops.
Character	The first character of the value of the business object attribute is used.
Enumeration	The connector uses the values defined in the IDL files and converts them into a valid CORBA enumeration, and also converts a business object sent through CORBA back into a string representation.

AppSpecificInfo format: The connector requires that the text strings in the `AppSpecificInfo` field be structured in the following format:

- A numeral represents the intended data type.
- A colon follows the numeral.
- Data other than the data type (such as a class package or an Enumerated type) is appended after the colon.

The syntax for this format is:

`data_type:class_package (if required)`

The following table shows the numerals that indicate specific data types for the MetaSolv API, and indicates whether additional data needs to be appended:

Numeral	Data type	Additional data (if required)
0	Integer/Long	none
1	Float/Double	none
2	Character	none
3	String	none
4	Enumerated Type	class package
5	Date Structure	class package
6	Object/Seq	class package
7	Union	class package
8	Other	

For example, to represent a long data type, in the AppSpecificInfo field of the business object, you enter:

```
0:
```

Similarly, the character and string data types are represented in the AppSpecificInfo field by the following:

```
2:
3:
```

If the data type is an enumerated type, a date structure, or a MetaSolv object or union, the full class package must appear in the AppSpecificInfo field. The class package tells the connector how to locate the associated Java file that it will be working with. For example, for a date structure, enter `5:MetaSolv.CORBA.WDIPSR.data.DateStruct` in the AppSpecificInfo field, because that is the full class package to the DateStruct class that enables MetaSolv to handle date structures.

Not all attributes or sub-objects within the MetaSolv API are supported for both exporting from and importing into the MetaSolv API. In order to allow the use of the same business object in both directions, it is necessary to avoid the setting of non-supported fields. This is achieved by an additional parameter in the application-specific information, as described in the following table:

Character	Meaning
N	Not supported in the API
I	Supported only during import
E	Supported only during export
A	Supported during import and export (Default)

This attribute is specified at the third place in the AppSpecificInfo field (for example, `0::I` indicates an integer value that is supported only during inbound operations).

If this meta attribute is not specified, the connector assumes that the attribute is supported during import and export.

Business object structure for MetaSolv

WebSphere Business Integration Adapter business objects are hierarchical: parent business objects can contain child business objects, which can in turn contain child business objects, and so on.

For the connector, the containment relationship between a parent and a child business object can have cardinality 1 or cardinality n. In addition, the connector supports a parent/child object relationship of type Union:

- A cardinality 1 container occurs when an attribute in a parent business object references a single child object.
- A cardinality n container object occurs when an attribute in the parent business object references an array of child business objects. The array can contain zero or more child business objects.
- A container that has the type Union can contain child business objects of different types. The child object that is being used in the relationship is defined by a type attribute of its parent. This type is usually used to implement a child object that has a cardinality of 1, but that is optional for its parent.

Business object attribute properties

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

Property	Meaning
Default Value	Not used.
Max Length	Not used.
Name	Name of the attribute as it appears in the MetaSolv API.
Type	All simple types are of type String.
Foreign Key	Not used.
Key	Defines the Key value of the business object.
Required	

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.

Special attribute values

IsIgnore, IsBlank

The MetaSolv API requires that every attribute be specified when the API object is sent to the server. For both `CxIgnore` and `CxBlank`, the connector uses the WebSphere Business Integration Adapter-defined blank value.

The connector uses IsBlank as the business object equivalent of a fixed-length string, up to the maximum length of the attribute.

Date Format

The connector expects dates in the WebSphere Business Integration Adapter generic date format (YYYYMMDD hhmmss). However, the time information is ignored, because the MetaSolv API does not support time.

Connector interactions with the integration broker

The connector handles business objects in response to requests from the integration broker.

Business object retrieval

When the connector receives an integration broker's request to retrieve a business object, it returns a business object that corresponds exactly to the current MetaSolv application database representation of that business object. That is, for each simple attribute of each individual business object within a hierarchical business object, there is a value from a corresponding attribute in the MetaSolv database. In addition, the connector populates hierarchical business object containers with the exact number of individual business objects that exist in the application.

Verb determination

In the WebSphere Business Integration Adapter business object structure, each individual business object within a hierarchical business object contains its own verb. If the connector receives from the integration broker a business object that has different verbs for parent and child business objects, the connector uses only

the verb of the top-level parent business object in processing the business object. Verbs of child business objects are ignored by the connector if they differ from the verb of the parent business object.

After-images versus deltas

The connector is implemented to handle after-image business objects.

Chapter 4. The MetaSolv Object Discovery Agent (ODA)

This chapter describes the Object Discovery Agent (ODA) for MetaSolv, and how to use it to generate business object definitions for the IBM WebSphere Business Integration Adapter for MetaSolv.

This chapter contains the following sections:

Overview of the ODA for MetaSolv

An Object Discovery Agent (ODA) enables you to generate business object definitions. A business object definition is a template which describes a business object. The ODA examines specified application objects, "discovers" the elements of those objects that correspond to business object attributes, and generates business object definitions to represent the information. Business Object Designer provides a graphical interface to access the Object Discovery Agent and to work with it interactively.

The ODA for MetaSolv runs only on Windows 2000 and generates business object definitions from meta-data contained in MetaSolv IDL files. The Business Object Designer wizard automates the process of creating these definitions. You use the ODA to create business objects and Connector Configurator to configure the connector to support them. For information about Connector Configurator, see Appendix B, "Using Connector Configurator," on page 55.

Mapping MetaSolv application data types

The data types used by the MetaSolv application are mapped to WBI data types according to the following table.

Table 5. Mapping MetaSolv data types

Application data types	**WebSphere data types**
Int, long	String
Char	String
String	String
Float	String
DateStruct	String
Boolean	String
Object	Object

Generating business object definitions

This section describes how to use the MetaSolv ODA in Business Object Designer to generate business object definitions. For information on launching and using Business Object Designer, see *IBM WebSphere Business Integration Adapters Business Object Development Guide*.

Starting the ODA

The ODA runs on Windows 2000 on a machine on which the meta-data repository (that is, the IDL files) resides, using the `start_MetaSolvODA.bat` from the Windows 2000 command line.

The ODA for MetaSolv has a default name of `MetaSolvODA`. The name can be changed by changing the value of the `AGENTNAME` variable in the start script.

To start the ODA, from a Windows 2000 console window run this command:

```
start_MetaSolvODA
```

You configure and run MetaSolvODA using Business Object Designer. Business Object Designer 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 MetaSolvODA. You can run multiple instances of the ODA, either on the local host or a remote host in the network. Each instance has to run on a unique port if multiple instances are being run on the same machine.

Running the business object designer

Business Object Designer provides a wizard that guides you through the steps to generate a business object definition using the ODA. The steps are as follows:

Selecting the agent

1. Start Business Object Designer.
2. Click **File > New Using ODA**.

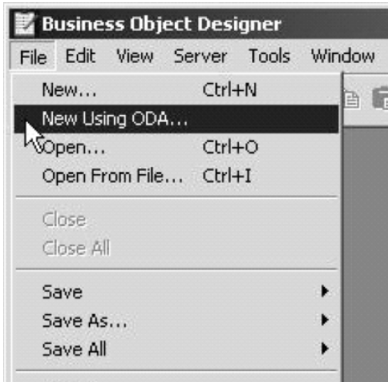

Figure 2. Select **New Using ODA**

3. The *Business Object Wizard - Step 1 of 6 - Select Agent* screen appears.

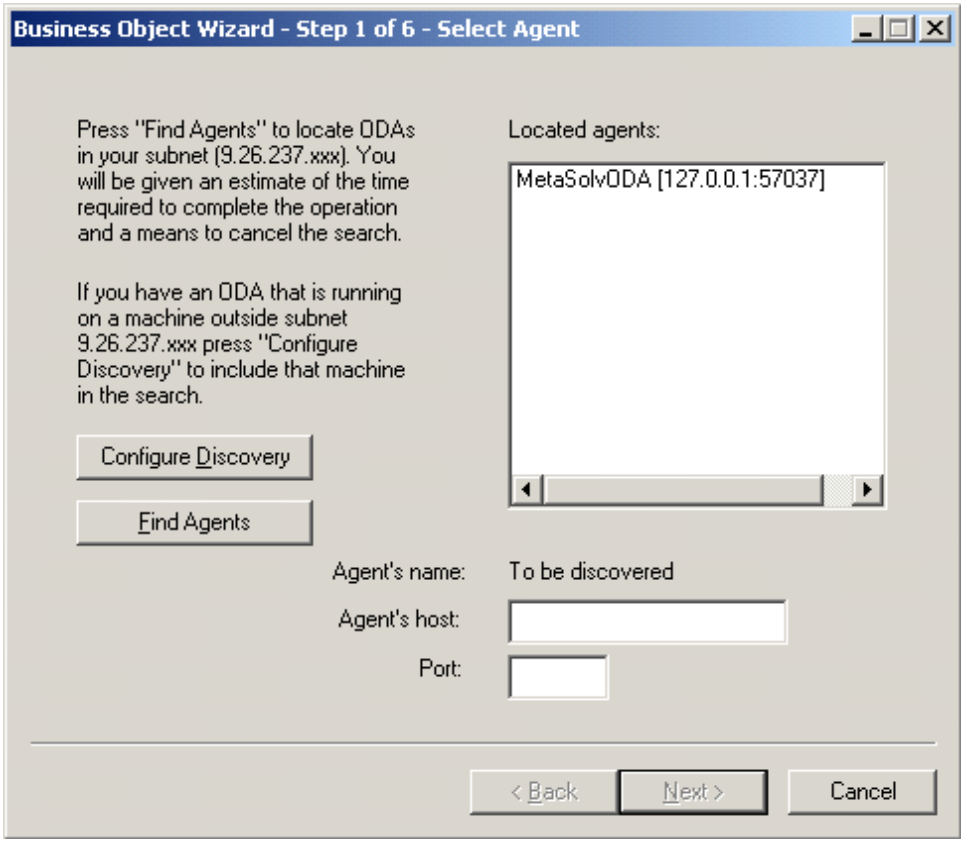

Figure 3. Step 1 - Selecting the agent

- Use the **Find Agents** button
- or type the **Agent's host:** name in the text entry box and type the **Port:** number in its text entry box.

4. Click **Next**.

Configure the agent

After you click **Next** on the Select Agent screen, the *Business Object Wizard - Step 2 of 6 - Configure Agent* screen appears which illustrates sample values.

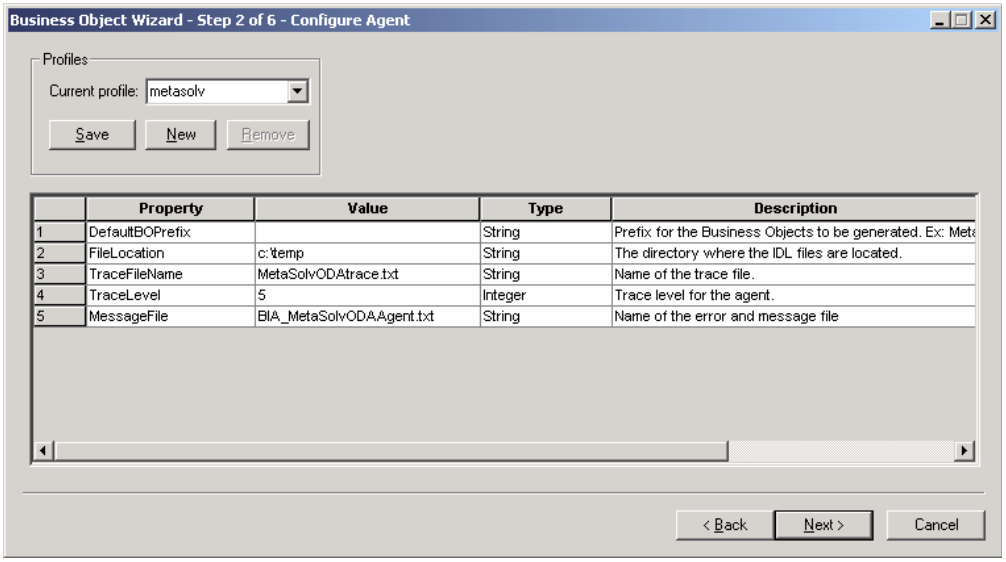

Figure 4. Step 2 - Configure the agent

You can save all the values you enter on this screen to a profile. Instead of retyping the property data next time you run the ODA, you simply select a profile from the drop-down menu and re-use the saved values. You can save multiple profiles, each with a different set of specified values.

1. Use the **New** and **Save** buttons in the Profiles group box any time you want the ODA to create a new profile. When you use the ODA again, you can select an existing profile.
2. Type the value of each property, as defined in Table 6.

   Note: If you use a profile, the property values are filled in for you, though you can modify the values as needed. You can also save new values.

Table 6. Configure agent properties

Property name	Default value	Type	Required	Description
`DefaultBOPrefix`	None	String	Yes	The BO prefix that is used by by the ODA for BO definitions.
`FileLocation`	None	String	Yes	Directory where IDL files are located.
`TraceFileName`	`MetaSolvODA trace.txt`	String	Yes	The name of the trace message file; for example, `MetaSolvODAtrace.txt`.
`TraceLevel`	5	Integer	Yes	The tracing level (from 0 to 5) for the Agent.

Table 6. Configure agent properties (continued)

Property name	Default value	Type	Required	Description
`MessageFile`	`BIA_MetaSolvODA Agent.txt`	String	Yes	The name of the message file that contains all the messages displayed by the ODA. For MetaSolv, the name of this file is `BIA_MetaSolvODAAgent.txt`. If you do not correctly specify the name of the message file, the ODA will run without messages.

Select MetaSolv classes

The *Business Object Wizard - Step 3 of 6 - Select Source* screen appears, as illustrated in Figure 5 on page 30.

This step provides the list of MetaSolv IDL files present in the folder specified by the `File Location` on the previous screen. Clicking on the file which is to be used for generating BO definitions displays the hierarchical structure of the MetaSolv classes present in the IDL file. The hierarchy can be traversed until the leaf nodes are reached in the class tree. Multiple classes can be selected for generation and they could belong to any level in the hierarchy or could be part of a separate hierarchy. Only the classes which are selected will be generated by the ODA.

Children of a class will not be generated by default, they have to be explicitly selected. This can be done by holding the shift key and clicking the child objects which have to be generated. When a parent storable class is selected it will generate the parent to child relationship for all the children within the parent class but the definitions of only the selected child objects will be generated.

For resolving the dependencies within IDL files (i.e. a class has an attribute of a type which has the definition in some other IDL file) there are two methods to resolve these issues:

- Generate the required object from the IDL while which is imported before generating BO's from the main IDL file.
- Include the text from the imported IDL file within the main IDL file.

If PSR_Types2.idl needs WDIUtil.idl file then either the definitions of objects need can be copied into PSRTypes2.idl or the ODA could be run on WDIUtil.idl file first and generate the dependent objects before the objects are generated from PSRTypes2.idl.

The generated BO's have some properties which have to be set individually before the BO can be used by the Connector. The key fields for the BO have to be marked manually. The name of the key field in the BO has to be specified as BO ASI for the respective BO. For the MetaSolv adapter the BO's expect only one key field. The name of that key field must be set as BO ASI. For In MetaSolv_Customer BO the key attribute **`custAcctId`** is set as the BO ASI. The adapter does not use required and foreign key properties of the attributes and so are not set by the ODA.

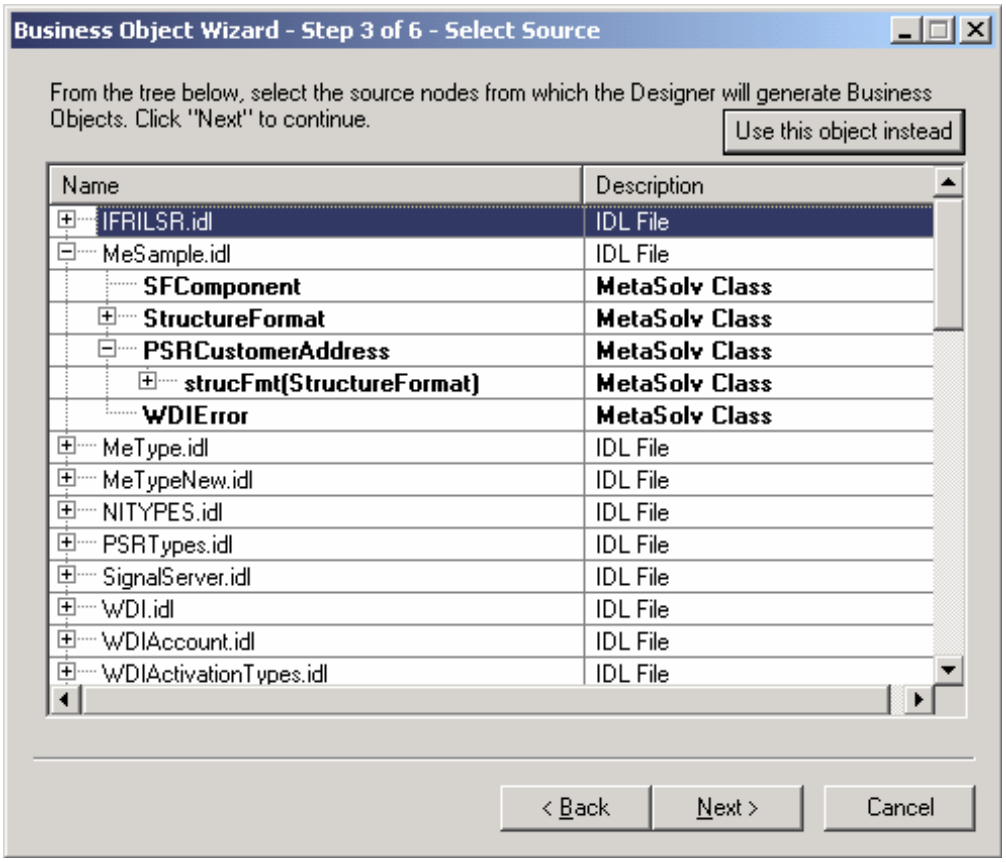

Figure 5. Step 3 - Select Source

1. If necessary, expand a MetaSolv node to see a list of sub-objects.
2. Select the MetaSolv object(s) you want to use. In Figure 5, the **PSRCustomerAccount** object is selected
3. Click **Next**.

Confirm the object selection

The *Business Object Wizard - Step 4 of 6 - Confirm source nodes for business object definitions* screen appears. It shows the object(s) you selected.

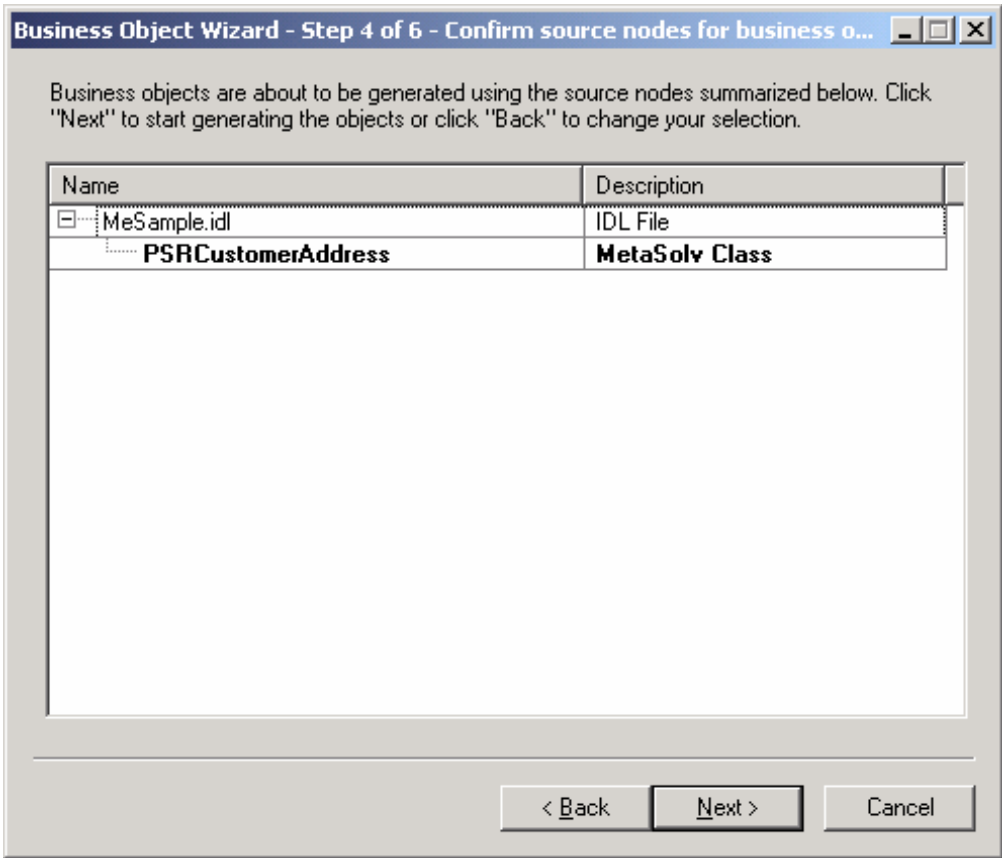

Figure 6. Step 4 - Confirm source node

Click **Back** to make changes or **Next** to confirm that the list is correct.

The *Business Object Wizard - Step 5 of 6 - Generating business objects...* screen appears with a message stating that the wizard is generating the business objects.

Specifying business object properties

This step will be used to gather additional information needed to generate BO from selected MetaSolv classes. After the information has been gathered, an IBM WBI business object is generated. This is a two step process. The first step selected the common properties like BO prefix and supported verbs for the selected MetaSolv classes and if the ODA should allow the user to generate verb level `AppSpecificInfo`. Second step is to get the verb level Application Specific Information for each generated BO.

Step 1: Common BO properties screen

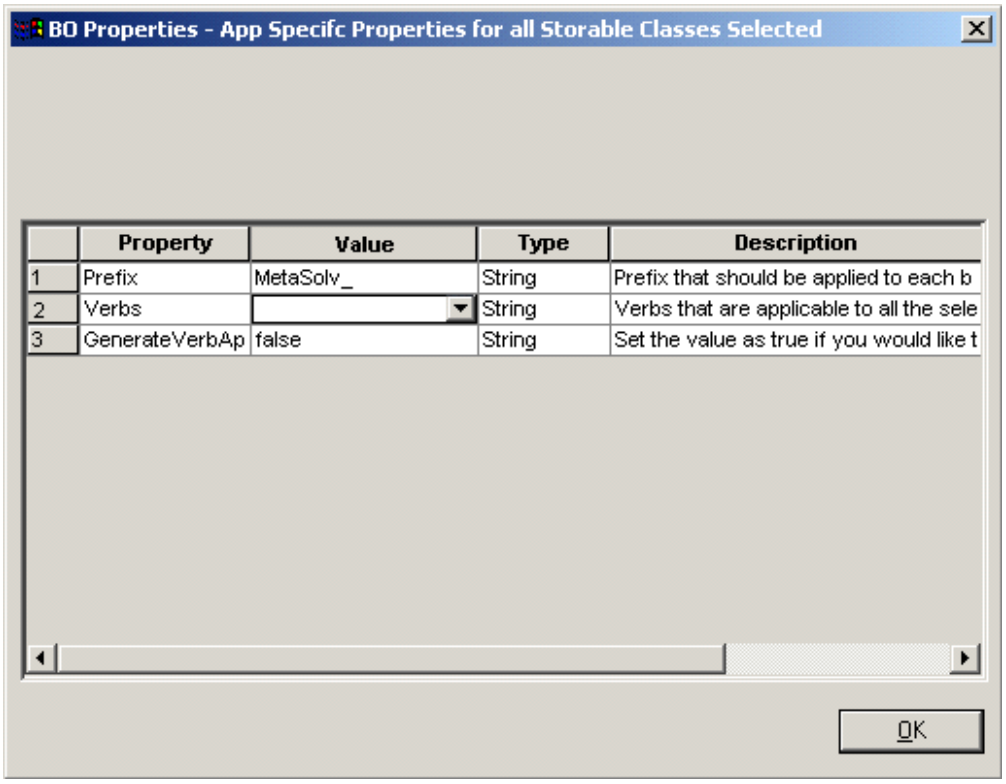

Figure 7. Step 5 - Select verbs

`Prefix` is used to set the prefix for the Business Objects to be generated. In this example it is MetaSolv_

`Verbs` is a drop down check list of verbs that will be applied to all the selected objects.

`GenerateVerbAppInfo` Set this value to **`true`** if you would like to specify verb level **AppSpecificInfo** during generation.

For details about business object verbs for the MetaSolv connector, see Chapter 3, "Creating or modifying business objects," on page 17

1. In the **Value** list for the Verbs property, select the verbs that you want the business object to support. You can select one or more verbs. You can also deselect a verb at any time.

Figure 8. Selected verbs

2. Click **OK**.

Step 2: Capturing verb application specific information

For each class selected in "Step 1: Common BO properties screen" on page 32, a separate window appears where you specify the method sequence that must be executed for the verb. This window is displayed only if the property `GenreateVerbApp` is set to true in the previous step

Figure 8 illustrates this screen for the selected verbs of the **`PSRCustomerAccount`** business object created in Figure 5 on page 30 and Figure 6 on page 31.

Verb level ASI

The format for verb level App Specific Info is `MO_MetaSolv_<verb name>_<BOName>` where `<verb name>` is the name of the selected verb, `Create`, `Retrieve` etc.

`<BOName>` is the name of a business object like `MetaSolv_PSRCustomerAccount`.

Attribute level ASI

The format of the attribute level `App Specific Info` is `<MetaSolv Data Type>:<Class Name>` where `<MetaSolv Data Type>` is an integer corresponding to MetaSolv data type in the following table.

Table 7. MetaSolv data types

MetaSolv Application Data type	ASI Integer valule
`Int, long`	0
`float`	1
`char`	2
`string`	3
`DateStruct`	4
`Enumeration`	5
`Object`	6
`Union`	7
`Sequence`	8
`boolean`	9

`<Class Name>` is the name of the class including the full package name. This is required only if the attribute is a complex attribute. If the attribute maps to a class like `PSRBillingAccountUnion` the ASI would be:

```
7:MetaSolv.CORBA.WDIPSR.PSRBillingAccountUnion
```

The BO ASI will not be set by the ODA. It must be manually set by the users. Please refer to the MetaSolv adapter guide for information on format of BO ASI.

Open the business object in a separate window

The *Business Object Wizard - Step 6 of 6 - Save business objects* screen appears.

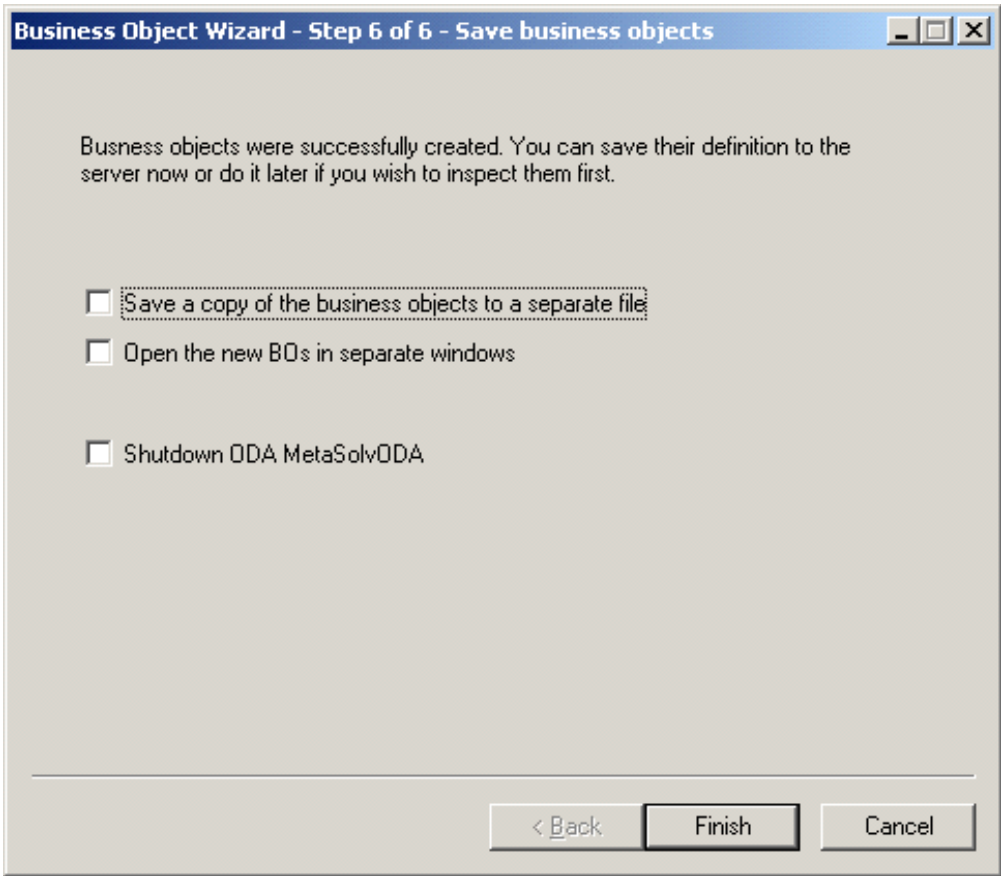

Figure 9. Step 6 - Save business objects screen

You must open the business object in a separate window within Business Object Designer, and, after specifying a key for the top-level business object, you can save the generated business object definitions to a file.

To open the business objects in separate windows:

1. Select **Open the new BOs in separate windows**. A dialog box appears.
2. Click **Finish**. Each business object appears in a separate window where you can view and set the ASI information for the business objects. For details, see "Specifying business object properties" on page 31.

To save the business objects to a file (only after you a have specified a key for the parent-level business object, as illustrated in Figure 9):

1. Select **Save a copy of the business objects to a separate file**. A dialog box appears.
2. Type the location in which you want the copy of the new business object definitions to be saved.

Business Object Designer saves the files to the specified location.

If you have finished working with the ODA, you can shut it down by checking **Shutdown ODA MetaSolv ODA** before clicking **Finish**.

Uploading business object files

The newly created business object definition files must be uploaded to the integration broker once they have been created. The process depends on whether you are running WebSphere InterChange Server, WebSphere MQ Integrator Broker, or WebSphere Application Server.

- **WebSphere InterChange Server**: If you have saved your business object definition files to a local machine and need to upload them to the repository on the server, refer to the *Implementation Guide for WebSphere InterChange Server*.
- **WebSphere MQ Integrator Broker**: You must export the business object definitions out of Business Object Designer and into the integration broker. For details, refer to *Implementing Adapters with WebSphere MQ Integrator Broker*
- **WebSphere Application Server**: For details, see *Implementing Adapters with WebSphere Application Server*

Appendix A. Standard configuration properties for connectors

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

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

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

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

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

New and deleted properties

These standard properties have been added in this release.

New properties

- XMLNameSpaceFormat

Deleted properties

- RestartCount

Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties
- Connector-specific configuration properties

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

Using Connector Configurator

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

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

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

Setting and updating property values

The default length of a property field is 255 characters.

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

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

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

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

To determine how a specific property is updated, refer to the **Update Method** column in the Connector Configurator window, or see the Update Method column in Table 8 on page 39 below.

Summary of standard properties

Table 8 on page 39 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker, as standard property dependencies are based on `RepositoryDirectory`.

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

Note: In the "Notes"column in Table 8 on page 39, the phrase "Repository directory is REMOTE" indicates that the broker is the InterChange Server. When the broker is WMQI or WAS, the repository directory is set to LOCAL

Table 8. Summary of standard configuration properties

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

Table 8. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
DuplicateEventElimination	`true or false`	`false`	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	Component restart	JMS transport only
jms.FactoryClassName	`CxCommon.Messaging.jms .IBMMQSeriesFactory` or `CxCommon.Messaging .jms.SonicMQFactory` or any Java class name	`CxCommon.Messaging. jms.IBMMQSeriesFactory`	Component restart	JMS transport only
jms.MessageBrokerName	If FactoryClassName is `IBM`, use `crossworlds.queue. manager`. If FactoryClassName is `Sonic`, use `localhost:2506`.	`crossworlds.queue.manager`	Component restart	JMS transport only
jms.NumConcurrentRequests	Positive integer	`10`	Component restart	JMS transport only
jms.Password	Any valid password		Component restart	JMS transport only
jms.UserName	Any valid name		Component restart	JMS transport only
JvmMaxHeapSize	Heap size in megabytes	`128m`	Component restart	Repository directory is <REMOTE> (broker is ICS)
JvmMaxNativeStackSize	Size of stack in kilobytes	`128k`	Component restart	Repository directory is <REMOTE> (broker is ICS)
JvmMinHeapSize	Heap size in megabytes	`1m`	Component restart	Repository directory is <REMOTE> (broker is ICS)
ListenerConcurrency	`1- 100`	`1`	Component restart	Delivery Transport must be `MQ`
Locale	`en_US`, `ja_JP`, `ko_KR`, `zh_CN`, `zh_TW`, `fr_FR`, `de_DE`, `it_IT`, `es_ES`, `pt_BR` **Note:** This is a subset of the supported locales.	`en_US`	Component restart	

Table 8. Summary of standard configuration properties (continued)

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

Table 8. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
RepositoryDirectory	Location of metadata repository		Agent restart	For ICS: set to <REMOTE> For WebSphere MQ message brokers and WAS: set to C:\crossworlds\ repository
RequestQueue	Valid JMS queue name	*CONNECTORNAME*/REQUESTQUEUE	Component restart	Delivery Transport is JMS
ResponseQueue	Valid JMS queue name	*CONNECTORNAME*/RESPONSEQUEUE	Component restart	Delivery Transport is JMS: required only if Repository directory is <REMOTE>
RestartRetryCount	0-99	3	Dynamic	
RestartRetryInterval	A sensible positive value in minutes: 1 - 2147483547	1	Dynamic	
RHF2MessageDomain	mrm, xml	mrm	Component restart	Only if Delivery Transport is JMS and WireFormat is CwXML.
SourceQueue	Valid WebSphere MQ name	*CONNECTORNAME*/SOURCEQUEUE	Agent restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	Component restart	Delivery Transport is JMS
SynchronousRequestTimeout	0 - any number (millisecs)	0	Component restart	Delivery Transport is JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	Component restart	Delivery Transport is JMS
WireFormat	CwXML, CwBO	CwXML	Agent restart	CwXML if Repository Directory is not <REMOTE>: CwBO if Repository Directory is <REMOTE>
WsifSynchronousRequestTimeout	0 - any number (millisecs)	0	Component restart	WAS only
XMLNameSpaceFormat	short, long	short	Agent restart	WebSphere MQ message brokers and WAS only

Standard configuration properties

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

AdminInQueue

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

The default value is `CONNECTORNAME/ADMININQUEUE`.

AdminOutQueue

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

The default value is `CONNECTORNAME/ADMINOUTQUEUE`.

AgentConnections

Applicable only if `RepositoryDirectory` is <REMOTE>.

The AgentConnections property controls the number of ORB (Object Request Broker) connections opened by `orb.init[]`.

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

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

ConcurrentEventTriggeredFlows

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

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

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

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

ContainerManagedEvents

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

There is no default value.

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

- PollQuantity = 1 to 500
- SourceQueue = /SOURCEQUEUE

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

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

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

The fields for these values in the Data Handler tab will be displayed only if you have set ContainerManagedEvents to JMS.

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

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

ControllerStoreAndForwardMode

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

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

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

The default is true.

ControllerTraceLevel

Applicable only if RepositoryDirectory is <REMOTE>.

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

DeliveryQueue

Applicable only if DeliveryTransport is JMS.

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

The default value is CONNECTORNAME/DELIVERYQUEUE.

DeliveryTransport

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

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

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

WebSphere MQ and IDL

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

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

JMS

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

This property can also be set to `false`.

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

FaultQueue

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

The default value is CONNECTORNAME/FAULTQUEUE.

JvmMaxHeapSize

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

The default value is 128m.

JvmMaxNativeStackSize

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

The default value is 128k.

JvmMinHeapSize

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

The default value is 1m.

jms.FactoryClassName

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

The default is CxCommon.Messaging.jms.IBMMQSeriesFactory.

jms.MessageBrokerName

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

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

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

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

jms.NumConcurrentRequests

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

The default value is 10.

jms.Password

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

There is no default.

jms.UserName

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

There is no default.

ListenerConcurrency

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

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

Locale

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

A locale name has the following format:
ll_TT.codeset

where:

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

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

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

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

LogAtInterchangeEnd

Applicable only if RespositoryDirectory is <REMOTE>.

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

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

MaxEventCapacity

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

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

MessageFileName

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

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

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

MonitorQueue

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

The default value is CONNECTORNAME/MONITORQUEUE

OADAutoRestartAgent

Valid only when the RepositoryDirectory is <REMOTE>.

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

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

The default value is `false`.

OADMaxNumRetry

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

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

The default value is `1000`.

OADRetryTimeInterval

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

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

The default is `10`.

PollEndTime

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

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

PollFrequency

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

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

Set `PollFrequency` to one of the following values:

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

The default is 10000.

Important: Some connectors have restrictions on the use of this property. Where they exist, these restrictions are documented in the chapter on installing and configuring the adapter.

PollQuantity

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

FIX

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

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

PollStartTime

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

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

RequestQueue

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

The default value is CONNECTOR/REQUESTQUEUE.

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

The default is 3.

RestartRetryInterval

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

The default is 1.

RHF2MessageDomain

WebSphere message brokers and WAS only.

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

A sample header would look like this:

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

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

SourceQueue

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

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

The default value is `CONNECTOR/SOURCEQUEUE`.

SynchronousRequestQueue

Applicable only if `DeliveryTransport` is JMS.

Delivers request messages that require a synchronous response from the connector framework to the broker. This queue is necessary only if the connector uses synchronous execution. With synchronous execution, the connector framework

sends a message to the SynchronousRequestQueue and waits for a response back from the broker on the SynchronousResponseQueue. The response message sent to the connector bears a correlation ID that matches the ID of the original message.

The default is CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE

SynchronousResponseQueue

Applicable only if DeliveryTransport is JMS.

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

The default is CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE

SynchronousRequestTimeout

Applicable only if DeliveryTransport is JMS.

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

The default value is 0.

WireFormat

Message format on the transport.

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

WsifSynchronousRequestTimeout

WAS integration broker only.

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

The default value is 0.

XMLNameSpaceFormat

WebSphere message brokers and WAS integration broker only.

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

The default value is short.

Appendix B. Using Connector Configurator

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

You use Connector Configurator to:

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

Note:

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

The topics covered in this appendix are:

Overview of Connector Configurator

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

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

You use Connector Configurator to:

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

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see "Running Configurator in stand-alone mode" on page 56).

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 57 to set up a new one.

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

Starting Connector Configurator

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

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

Running Configurator in stand-alone mode

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

To do so:

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

You may choose to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see "Completing a configuration file" on page 61.)

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 57.
- To use an existing file, simply modify an existing template and save it under the new name. You can find existing templates in your `\WebSphereAdapters\bin\Data\App` directory.

Creating a new template

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

To create a template in Connector Configurator:

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

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

Specifying general characteristics

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

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

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

Specifying values

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

1. Click the **Value** tab. The display panel for Value replaces the display panel for General.
2. Select the name of the property in the **Edit properties** display.
3. In the fields for **Max Length** and **Max Multiple Values**, enter your values.

To create a new property value:

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

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

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

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

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

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

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

Setting dependencies

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

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

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

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

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

Creating a new configuration file

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

- In the System Manager window, right-click on the **Connectors** folder and select **Create New Connector**. Connector Configurator opens and displays the **New Connector** dialog box.
- In stand-alone mode: from Connector Configurator, select **File>New>Connector Configuration**. In the New Connector window, enter the name of the new connector.

You also need to select an integration broker. The broker you select determines the properties that will appear in the configuration file. To select a broker:

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

Creating a configuration file from a connector-specific template

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

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

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

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

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

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

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

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

Using an existing file

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

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

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

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

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

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

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

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

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

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

Completing a configuration file

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

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

1. Under the **Standard Properties** tab, select the value field for the `BrokerType` property. In the drop-down menu, select the value `ICS`, `WMQI`, or `WAS`.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 64..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

Setting the configuration file properties

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

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

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

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

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

- Associated Maps
- Resources
- Messaging (where applicable)

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

Standard properties differ from connector-specific properties as follows:

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

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

- A field with a grey background indicates a standard property. You can change the value but cannot change the name or remove the property.

- A field with a white background indicates an application-specific property. These properties vary according to the specific needs of the application or connector. You can change the value and delete these properties.
- Value fields are configurable.
- The **Update Method** field is displayed for each property. It indicates whether a component or agent restart is necessary to activate changed values. You cannot configure this setting.

Setting standard connector properties

To change the value of a standard property:

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

Setting application-specific configuration properties

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

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

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

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

Encryption for connector properties

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

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

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

Update method

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

Specifying supported business object definitions

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

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

If ICS is your broker

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

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

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

To delete a business object from the supported list:

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

Deleting a business object from the supported list changes the connector definition and makes the deleted business object unavailable for use in this implementation

of this connector. It does not affect the connector code, nor does it remove the business object definition itself from System Manager.

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

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

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

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

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

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

If a WebSphere Message Broker is your broker

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

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

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

If WAS is your broker

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

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

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

Associated maps (ICS only)

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

The list of business objects contains the application-specific business object which the agent supports and the corresponding generic object that the controller sends

to the subscribing collaboration. The association of a map determines which map will be used to transform the application-specific business object to the generic business object or the generic business object to the application-specific business object.

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

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

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

- **Business Object Name**

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

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

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

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

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

  To explicitly bind a map:

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

Resources (ICS)

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

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

Messaging (ICS)

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

Setting trace/log file values

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

To change the logging and tracing values:

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

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

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

Data handlers

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

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

Saving your configuration file

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

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

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

- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder. By default, the file is saved to `\WebSphereAdapters\bin\Data\App`.
- You can also save it to a WebSphere Application Server project if you have set one up.

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

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

Changing a configuration file

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

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

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

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

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

Index

Special characters

A

B

C

E

G

M

O

S

U

V

Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

COPYRIGHT LICENSE

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

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

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

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

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

WebSphere Business Integration Adapter Framework V2.4.0

Printed in USA