

IBM WebSphere Business Integration Server Express
and Express Plus



Adapter for iSeries User Guide

Server Express Version 2.0.x

IBM WebSphere Business Integration Server Express
and Express Plus



Adapter for iSeries User Guide

Server Express Version 2.0.x

Note!

Before using this information and the product it supports, read the information in "Notices" on page 71.

15July2005

This edition of this document applies to connector version 2.0.x and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

© **Copyright International Business Machines Corporation 2004, 2005. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this document	v
Audience	v
Related documents	v
Typographic conventions	vi
New in this release	vii
Version 2.0	vii
Chapter 1. Overview	1
An overview of the iSeries i5/OS system	1
How the adapter works	4
Chapter 2. Installing the iSeries adapter	5
Adapter for iSeries environment	5
Installing the iSeries adapter and related files	6
Installed file structure	6
Post installation tasks	7
Chapter 3. Configuring the iSeries adapter	9
Configuring the connector	9
Chapter 4. Understanding business objects for the connector	27
Defining connector metadata	27
Overview of business object structure	27
Business Object structure for RPG programs	28
Business Object structure for iSeries data queues	32
Configuring meta objects for polling	33
Specifying business object attribute properties	34
Specifying business object attribute level application text	35
Chapter 5. Creating and modifying business objects	37
Overview of the ODA for iSeries	37
Generating business object definitions	37
Specifying business object information	43
Uploading business objects	45
Chapter 6. Troubleshooting and error handling	47
Error handling	47
Logging	47
Tracing messages	47
Appendix. Standard configuration properties for connectors	49
New properties	49
Standard connector properties overview	49
Standard properties quick-reference	51
Standard properties	55
Index	69
Notices	71
Programming interface information	72
Trademarks and service marks	73

About this document

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

The adapters 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 adapters for iSeries^(TM).

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
- i5/OS application architecture
- iSeries Integrated File System

Related documents

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

You can download, install, and view the documentation at the following site:

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

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

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

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

Typographic conventions

This document uses the following conventions:

<code>courier font</code>	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.
<i>italic, italic</i>	Indicates a variable name or a cross-reference.
<i>blue outline</i>	A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference.
{ }	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, <code>option[,...]</code> 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 <code><server_name><connector_name>tmp.log</code> .
/, \	In this document, backslashes (\) are used as the convention for directory paths. For Linux 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.
<code>%text%</code> and <code>\$text</code>	Text within percent (%) signs indicates the value of the Windows text system variable or user variable. The equivalent notation in a Linux environment is <code>\$text</code> , indicating the value of the <code>text</code> Linux environment variable.
<code>ProductDir</code>	Represents the directory where the product is installed.

New in this release

Version 2.0

- Support for accessing data queues on iSeries
- Polling Support to monitor data queues. The adapter is now bi-directional. This means that the adapter supports both Event and Request Processing
- ODA facility for generating BO Specifications from RPGLE Source Programs and data queues on iSeries Machines
- Facility to call a particular RPG PGM multiple times using a single request BO. The BO can now have values returned from multiple calls to the same RPG Program
- A problem where in the RPG PGM was called twice in a single call was also fixed in this version.

Chapter 1. Overview

This chapter describes the adapters for iSeries. The adapter, using the IBM Toolbox for Java (a set of Java[™] classes), provides the ability to execute RPG programs and access data queues on an iSeries or i5/OS system. IBM's Toolbox for Java provides a set of classes to access data queues and run programs. The adapter uses these classes and information from the incoming business object to build the parameter list for executing the program and read/write from/to data queues. The iSeries adapter currently supports both request processing and event processing.

Adapters 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 adapters, 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:

- 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 adapter.

For more information about the relationship of the integration broker, InterChange Server Express, to the adapter, see the *System Administration Guide* or the *System Implementation Guide*.

This chapter contains the following sections:

- "An overview of the iSeries i5/OS system"
- "How the adapter works" on page 4

An overview of the iSeries i5/OS system

The IBM iSeries (known as i5/OS) is a highly integrated, reliable server platform that allows businesses to run multiple operating environments simultaneously. Their integrity and security characteristics allow them to be used in many critical applications.

RPG has evolved from being a simple Report Program Generator, (from which it got its name) into a powerful application development procedural language on iSeries machines. Currently it is supported on the ILE (Integrated Language Environment) on iSeries.

Host servers handle requests from client PCs or other devices running an application as illustrated in Figure 1 on page 2 to enable printing a document and other tasks. The iSeries and i5/OS computers are full function servers capable of performing many tasks at once, including file, database, applications, mail, print, multimedia, fax and wireless communications. Each task server runs as a separate job on the system, and each server job sends and receives data streams on a socket connection.

One of these host servers is the Remote Command and Distributed program call server. This server runs the programs on an iSeries or i5/OS system.

IBM's Toolbox for JAVA has a multitude of packages that handle different functionalities. For example, Access classes manage sign-on information, create and maintain socket connections, and send and receive data, while Command Call classes run iSeries and i5/OS batch commands.

IBM's iSeries adapter uses the Access classes, and Program Call classes to call the RPG program. Data conversion classes provide the capability to convert numeric and character data between iSeries or i5/OS and Java formats.

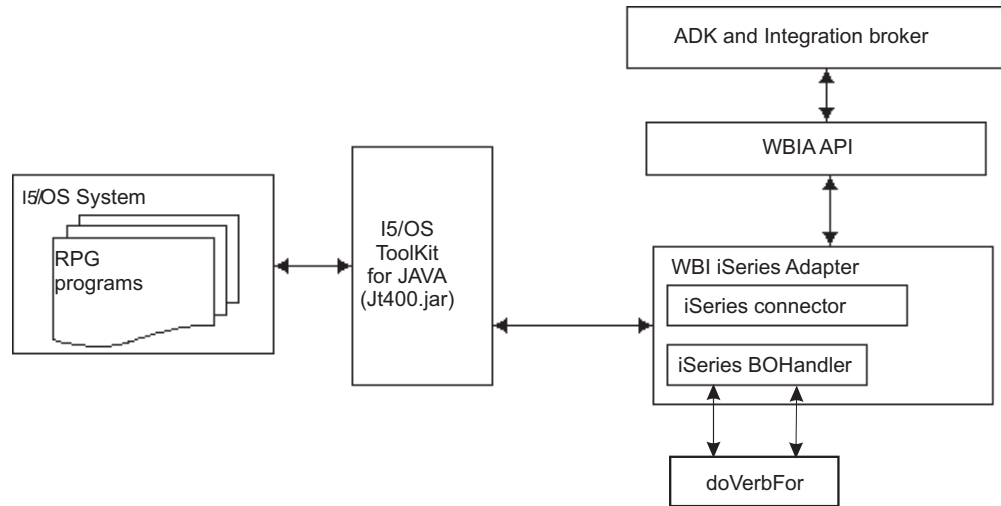


Figure 1. Overview of i5/OS client - server architecture

While i5/OS running on an i5/OS is capable of handling many types of tasks, the iSeries adapter only uses the Remote Command and Distributed program call server. This server runs the programs on the i5/OS system.

A diagram of the iSeries adapter connection of the client to the server is shown in Figure 2 on page 3.

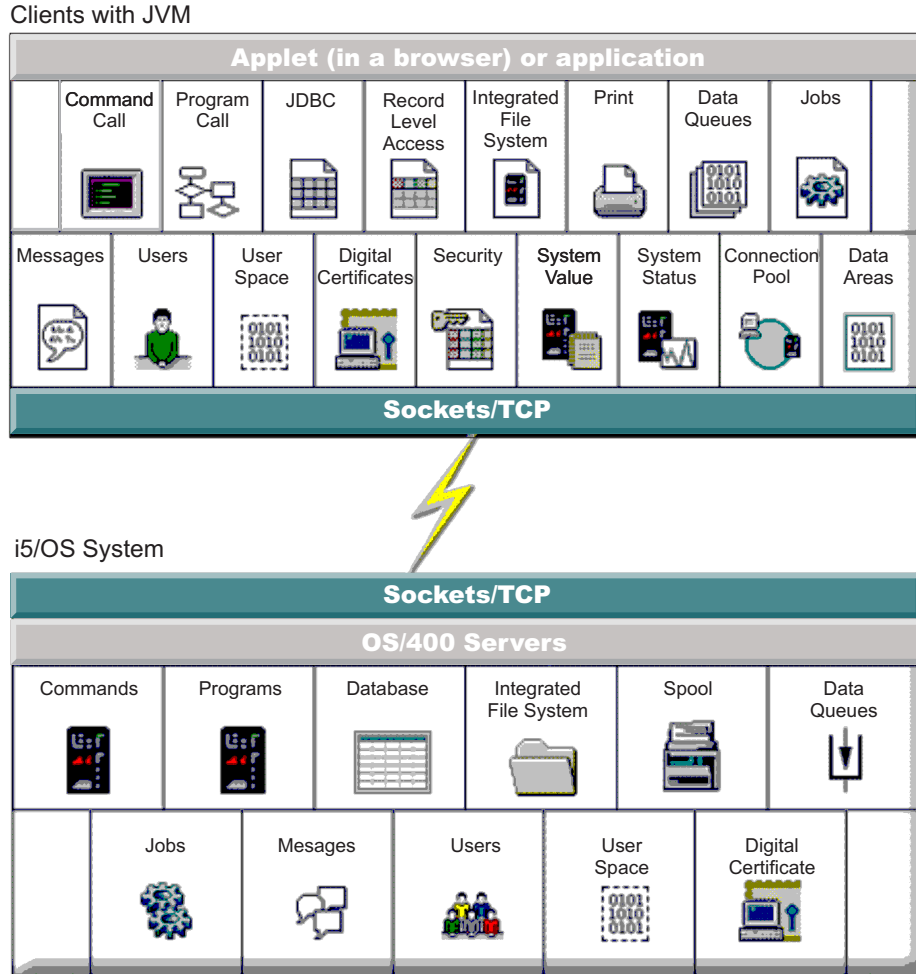


Figure 2. Diagram of the iSeries adapter's connections

Data queues

The data queues on iSeries allow fast communications between jobs. Therefore, it is an excellent way to synchronize and pass data between jobs. With data queues on iSeries:

- Many jobs can simultaneously access the data queues
- Messages on a data queue are free format
- The data queue can be used for either synchronous or asynchronous processing
- The messages on a data queue can be ordered in one of the following ways:
 - Last-in first-out (LIFO)
 - First-in first-out (FIFO)
 - Keyed

Each message on a keyed data queue has a key associated with it. A message can be taken off the queue only by specifying the key that is associated with it.

How the adapter works

The following sections describe how the adapter processes business objects.

Business object processing

The adapter receives business object requests from the integration broker and builds the parameter list for the RPG program, then establishes a connection with an iSeries or i5/OS system and executes the program.

The incoming business object contains the connection child attribute. The information in this attribute is used to connect to the iSeries or i5/OS system.

Connector operations

The connector passes business objects between the integration broker and a business object handler. The connector:

1. Registers the BOHandler with the framework.
2. The framework sends the BO request to the BOHandler.
3. The BOHandler uses the attribute information in the incoming business object to build the RPG program parameter list.
4. The BOHandler calls the RPG program running on the iSeries or i5/OS system.

Note: This is essentially a call to execute an RPG program on the iSeries or i5/OS system which then returns either a success or fail message.

5. The BOHandler then returns the results of the execution of the RPG program to the adapter framework. It also populates the business object with the returned parameters.

The adapter is written in Java and consists of two components:

- Connector
- Business object handler

Chapter 2. Installing the iSeries adapter

This chapter describes the process of installing and configuring the connector. It contains the following sections:

- “Adapter for iSeries environment”
- “Installing the iSeries adapter and related files” on page 6
- “Installed file structure” on page 6
- “Post installation tasks” on page 7

Adapter for iSeries environment

Before installing, configuring, and using the adapter, you must understand its environmental requirements:

- Broker compatibility
- Software prerequisites
- Adapter platforms

Broker compatibility

The adapter framework that an adapter uses must be compatible with the version of the integration broker with which the adapter is communicating. Version 2.0 of the adapter for iSeries is supported on the following versions of the adapter framework and with the integration broker InterChange Server Express, version 4.2.x.

Adapter framework: WebSphere Business Integration Adapter Framework, versions 2.1, 2.2, 2.3.x, 2.4, 2.5, and 2.6.

Note: For instructions on installing the integration broker and its prerequisites, see the *System Implementation Guide*.

Adapter platforms

The adapter runs on the following platforms:

- **Linux:**
Red Hat 2.1 Enterprise Linux WS/AS/ES 3.0 with Update 2, Intel (IA32)
SuSE Linux ES 8.1, Intel (IA32)
SuSE Linux ES 9.0, Intel (IA32)

Note: The TMTP (Tivoli Monitoring for Transaction Performance) component of the WebSphere Business Integration Adapter FrameworkV2.6 is not supported on Linux Red Hat.

- **Windows:**
Windows 2003 (Standard Edition or Enterprise Edition)
- OS/400 V5R2 and i5/OS V5R3. i5/OS refers to OS/400 and i5/OS unless explicitly stated.

Prerequisites

To use the connector, your environment must have the following:

1. Java and jar files:

- JDK 1.4 or later
- Java Secure Socket Extension 1.0 (JSSE)
- Jt400.jar file

Note: If the adapter is running on Windows, you can download the latest jt400.jar file from the Toolbox Web site at: <http://www-1.ibm.com/servers/eserver/series/toolbox/downloads.htm>. The jt400.jar needs to be copied to the %Product_dir%\connectors\iSeries directory.

When the adapter is running on iSeries, you can reference the jt400.jar file from the following location:
/QIBM/ProdData/HTTP/Public/jt400/lib/jt400.jar

- WBIA.jar file
 - CrossWorlds.jar file
 - BIA_iSeries.jar file
2. The iSeries adapter is designed to connect to an i5/OS with one of the following i5/OS versions:
 - Version 5, Releases 1 through 3
 - Version 4, Releases 1 through 3
 3. The Host Servers option of i5/OS must be installed and running.

Note: The i5/OS data queue server requires PTFs to correctly perform peek functions. You must have the appropriate PTF from the following link:
<http://www-1.ibm.com/servers/eserver/series/toolbox/hostservicepackdetail.htm>

4. You must be running RPG III or IV.

Installing the iSeries adapter and related files

For information on installing WebSphere Business Integration Server Express adapter products, refer to the *WebSphere Business Integration Server Express Installation Guide for Windows, for Linux, or for i5/OS*. The guide is located in the WebSphere Business Integration Server Express Adapters Infocenter at the following site: <http://www.ibm.com/websphere/wbiserverexpress/infocenter>.

Installed file structure

The adapter installation copies the standard files associated with the connector into your system. The utility installs the connector into the *ProductDir*\connectors\iSeries directory, and adds a shortcut for the connector to the Start menu.

Note: *ProductDir* represents the directory where the product is installed.

Table 1 describes the file structure used by the connector, and shows the files that are automatically installed when you install the connector through the Installer.

Table 1. File structure for the connector

Subdirectory of <i>ProductDir</i>	Description
\connectors\iSeries\BIA_iSeries.jar	Contains classes used by the iSeries connector only

Table 1. File structure for the connector (continued)

Subdirectory of <i>ProductDir</i>	Description
\connectors\iSeries\start_iSeries.bat	The startup script for the iSeries connector (Windows)
/connectors/iSeries/start_iSeries.sh	The startup script for the iSeries connector (Linux)
/connectors/iSeries/start_iSeries.sh	The startup script for the iSeries connector (i5/OS)
/connectors/iSeries/BIA_iSeries.jar	The startup script for the iSeries connector (i5/OS)
\connectors\iSeries\ext\	A directory where the ODA-generated .jar files can be saved. If you save to this directory, specify the directory in the startup script (start_iSeries.bat or start_iSeries.sh).
\connectors\messages\iSeriesAdapter.txt	Message file for the connector
\ODA\iSeries\BIA_iSeriesODA.jar	The iSeries ODA
\ODA\iSeries\start_iSeriesODA.bat	The ODA startup file (Windows)
/ODA/iSeries/start_iSeriesODA.sh	The ODA startup file (Linux)
/ODA/iSeries/start_iSeriesODA.sh	The ODA startup file (i5/OS)
repository\iSeries\CN_iSeries.txt	Repository definition for the connector.
\connectors\iSeries\start_iSeries_service.bat	The start-up script (Windows)

Post installation tasks

After installation and before startup, you must configure the adapter. For details, see Chapter 3, “Configuring the iSeries adapter,” on page 9.

Chapter 3. Configuring the iSeries adapter

This chapter describes the process of configuring the connector. It contains the following sections:

- “Configuring the connector”

Configuring the connector

The iSeries adapter uses standard connector properties for configuration as detailed in the next section and connector-specific properties as detailed in the following section.

This section includes the following topics:

- “Overview of Connector Configurator Express”
- “Starting Connector Configurator Express” on page 10
- “Running Connector Configurator Express from System Manager” on page 10
- “Creating a connector-specific property template” on page 11
- “Creating a new configuration file” on page 13
- “Setting the configuration file properties” on page 15
- “Saving your configuration file” on page 21
- “Completing the configuration” on page 21
- “Using Connector Configurator Express in a globalized environment” on page 22
- “Starting the connector” on page 22
- “Stopping the connector” on page 24
- “Creating multiple connector instances” on page 25

Overview of Connector Configurator Express

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

You use Connector Configurator Express to:

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

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

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

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

For **connector-specific properties**, however, you need first to define the properties and then set their values. You do this by creating a connector-specific property template for your particular adapter. There may already be a template set up in your system, in which case, you simply use that. If not, follow the steps in “Creating a new template” on page 11 to set up a new one.

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

Starting Connector Configurator Express

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

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

Running Connector Configurator Express in stand-alone mode

You can run Connector Configurator Express independently and work with connector configuration files.

To do so:

- From **Start>All Programs**, click **IBM WebSphere Business Integration Express>Toolset Express>Development>Connector Configurator Express**.
- Select **File>New>Connector Configuration**.
- When you click the pull-down menu next to **System Connectivity Integration Broker**, InterChange Server Express is displayed.

You may choose to run Connector Configurator Express independently to generate the file, and then connect to System Manager to save it in a System Manager project (see “Completing a configuration file” on page 15.)

Running Connector Configurator Express from System Manager

You can run Connector Configurator Express from System Manager.

To run Connector Configurator Express:

1. Open the System Manager.
2. In the System Manager window, expand the **Integration Component Libraries** icon and highlight **Connectors**.
3. From the System Manager menu bar, click **Tools>Connector Configurator Express**. The Connector Configurator Express window opens and displays a **New Connector** dialog box.

To edit an existing configuration file:

- In the System Manager window, select any of the configuration files listed in the Connector folder and right-click on it. Connector Configurator Express opens and displays the configuration file with the integration broker type and file name at the top.
- From Connector Configurator Express, 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.”
- 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 Express:

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 Express stores the information you have entered as an XML document, under \data\app in the\bin directory where you have installed Connector Configurator Express.

Creating a new configuration file

When you create a new configuration file, you must name it and select the integration broker, InterChange Server Express.

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

You also need to select the integration broker to determine the properties that appear in the configuration file. To select the broker:

- In the **Integration Broker** field, select InterChange Server Express.
- Use the drop-down menu to select values for 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 Express does not check the spelling of the name that you enter. You must ensure that the name is correct.

- **System Connectivity**

Click InterChange Server Express.

- **Select Connector-Specific Property Template**

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

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

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

Important: The directory path and name that you establish here must match the connector configuration file path and name that you supply in the startup file for the connector.

5. To complete the connector definition, enter values in the fields for each of the tabs of the Connector Configurator Express window, as described later in this chapter.

Using an existing file

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_iSeries.txt for the iSeries connector).
- An InterChange Server Express repository file.
Definitions used in a previous InterChange Server Express implementation of the connector may be available to you in a repository file that was used in the configuration of that connector. Such a file typically has the extension .in or .out.
- A previous configuration file for the connector.
Such a file typically has the extension *.cfg.

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

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

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

1. In Connector Configurator Express, click **File>Open>From File**.
2. In the **Open File Connector** dialog box, select one of the following file types to see the available files:

- Configuration (*.cfg)
- ICS Repository (*.in, *.out) (In this case, ICS stands for InterChange Server Express.)

Choose this option if a repository file was used to configure the connector in an InterChange Server Express environment. A repository file may include multiple connector definitions, all of which appear when you open the file.

- All files (*.*)

Choose this option if a *.txt file was delivered in the adapter package for the connector, or if a definition file is available under another extension.

3. In the directory display, navigate to the appropriate connector definition file, select it, and click **Open**.

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

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

Completing a configuration file

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

The title of the configuration screen displays the integration broker and connector name as specified in the file. Make sure InterChange Server Express is the 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 InterChange Server Express.
2. The Standard Properties tab will display the properties associated with the broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 18..
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 Express checks that values have been set for all required standard properties. If a required standard property is missing a value, Connector Configurator Express 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 Express displays a configuration screen with tabs for the categories of required configuration values.

Connector Configurator Express requires values for properties in these categories for connectors running InterChange Server Express:

- 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.

- Associated Maps

Important: Connector Configurator Express 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

Standard configuration properties provide information that all connectors use. See “Standard configuration properties for connectors,” on page 49 for documentation of these properties.

You must set at least the following standard connector configuration properties before running the connector:

- AgentTraceLevel
- ApplicationName

- ControllerStoreAndForwardMode
- ControllerTraceLevel
- DeliveryTransport

To change the value of a standard property:

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

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

Table 2 lists the connector-specific configuration properties for the connector. See the section that follows for explanations of the properties.

Table 2. Connector-specific configuration properties

Name	Possible values	Default value	Required
ApplicationName	iSeriesAdapter	None	Yes
UseDefaults	<i>default value</i>	None	Yes
MessageFileName	BIA_iSeriesAdapter.txt	BIA_iSeriesAdapter.txt	No
PollQuantity	an integer greater than 1	1	No

ApplicationName: This is a unique name that must be specified for each connector.

UseDefaults: For example, some of the input parameters to a program are constant. So these attributes can be designed to have default values. If there is no default value and the UseDefaults property is set to true, the adapter errors out and throws a VerbProcessingFailedException error message. If UseDefaults is not set or set to false, and there are no default values, the adapter builds a string of length MaxLength with padded spaces for the attribute values.

MessageFileName: This is the name and path of the error message file if it is not located in the standard message location %CROSSWORLDS%\connectors\messages. If the message file name is not in a fully qualified path, the message file is assumed to be located in the directory specified by the HOME environment variable or the startup parameter user.home. If a connector message file does not exist, the file BIA_iSeriesAdapter.txt is used as the message file.

PollQuantity: PollQuantity is an integer value above 1 specifying the number of items to poll from the data queues. Note that if n is specified as PollQuantity value then each queue configured using meta objects is polled n times. The default value is taken as 1.

Setting application-specific configuration properties

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

1. Right-click in the top left portion of the grid. A pop-up menu bar will appear. Click **Add** to add a property. To add a child property, right-click on the parent row number and click **Add child**.
2. Enter a value for the property or child property.
3. To encrypt a property, select the **Encrypt** box.
4. Choose to save or discard changes, as described for “Setting standard connector properties” on page 16.

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 “Configuration property values overview” on page 49.

Specifying supported business object definitions

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

Note: Some connectors require that certain business objects be specified as supported to perform event notification or additional configuration (using meta-objects) with their applications.

InterChange Server Express as your integration 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 Express 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 Express window, click **Delete Row**. The business object is removed from the list display.
3. From the **File** menu, click **Save to Project**.

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

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

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

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

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

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

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

Associated maps

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

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

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

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

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

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

- **Business Object Name**

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

- **Associated Maps**

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

- **Explicit**

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

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

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

To explicitly bind a map:

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

Setting trace/log file values

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

To change the logging and tracing values:

1. Click the **Trace/Log Files** tab.
2. For either logging or tracing, you can choose to write messages to one or both of the following:

- To console (STDOUT):
Writes logging or tracing messages to the STDOUT display.

Note: You can only use the STDOUT option from the **Trace/Log Files** tab for connectors running on the Windows platform.

- To File:
Writes logging or tracing messages to a file that you specify. To specify the file, click the directory button (ellipsis), navigate to the preferred location, provide a file name, and click **Save**. Logging or tracing message are written to the file and location that you specify.

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

Data handlers

The data handlers section is available for configuration only if you have designated a value of JMS for DeliveryTransport and a value of JMS for ContainerManagedEvents. 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.

Saving your configuration file

When you have finished configuring your connector, save the connector configuration file. Connector Configurator Express saves the file in the broker mode that you selected during configuration. The title bar of Connector Configurator Express always displays the broker mode 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`.

For details about using projects in System Manager and for further information about deployment, see the *System Implementation Guide*.

Completing the configuration

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

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

Using Connector Configurator Express in a globalized environment

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

Connector Configurator Express supports non-English characters in:

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

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

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

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

Starting the connector

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

On Linux systems the startup script should reside in the `ProductDir/bin` directory.

On i5/OS systems the startup script should reside in the `/QIBM/UserData/WBIServer44/<instance>/connectors/<ConnInstance/` with which the connector runs.

For InterChange Server Express:

```
/QIBM/UserData/WBIServer44/WebSphereICSName/connectors/iSeries/
start_iSeries.sh iSeries WebsphereICSName
```

For JMS,

```
/QIBM/UserData/WBIServer44/WebSphereAdapters/connectors/iSeries/
```



```
start_iSeries.sh iSeries
dummy -c/QIBM/UserData/WBIServer44/WbSphereAdapters/connectors
/iSeries/iSeriesConnector.cfg
```

The name of the startup script depends on the operating-system platform, as Table 3 shows.

Table 3. Startup scripts for a connector

Operating system	Startup script
Linux	connector_manager
i5/OS	start_iSeries.sh
Windows	start_iSeries.bat

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

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

- **Starting the connector on a Windows system:**

- From the **Start** menu, select **Programs>IBM WebSphere Business Integration Express>Adapters>Connectors**. By default, the program name is “IBM WebSphere Business Integration Server Express.” However, it can be customized. Alternatively, you can create a desktop shortcut to your connector.
- From the Windows command line: `start_connName connName brokerName {-configFile}`.
- 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).

- **Starting the connector on a Linux system:**

- From the command line type:
`connector_manager -start connName brokerName [-configFile]`

where *connName* is the name of the connector and *brokerName* identifies the integration broker.

- For InterChange Server Express, specify for *brokerName* the name of the InterChange Server Express instance.

- **Starting the connector on an i5/OS system:**

- From the Windows system where the WebSphere Business Integrations Server Express Console is installed, select **IBM WebSphere Business Integration Server Express>Toolset Express>Administrative>Console**. Then specify the i5/OS system name or IP address and a user profile and password that has *JOBCTL special authority. Select the connector from the list of connectors, and click Start
- To automatically start the adapter using the Console, use the `submit_adapter.sh` script. You can also set the adapter agent from the Console to start automatically. To set the option, right-click on the adapter agent and click to enable the **Autostart** check box.
- In Batch mode, from the i5/OS command line, you need to run the CL command QSH and from the QSHELL environment. Run
`/QIBM/ProdData/WBIServer44/bin/submit_adapter.sh connName`

WebSphereICSName pathToConnNameStartScript jobDescriptionName, where *connName* is the connector name, *WebSphereICSName* is the Interchange Server Express server name (default is QWIBDFT44), *pathToConnNameStartScript* is the full path to the connector start script, *jobDescriptionName* is the name of the job description to use in the QWIBSVR44 library.

- In interactive mode, you need to run the CL command QSH and from the QSHELL environment. Run
`/QIBM/UserData/WBIServer44/WebSphereICSName/connectors/iSeries/start_iSeries.sh iSeries WebSphereICSName [-cConfigFile]` where *connName* is the name of your connector and *WebSphereICSName* is the name of the InterChange Server Express instance.

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

Stopping the connector

The way to stop a connector depends on the way the connector was started.

- **Windows:**

- You can invoke the startup script which creates a separate “console” window for the connector. In this window, type “q” and press Enter to stop the connector.
- You can configure the connector to start as a Windows service. In this case, the connector stops when the Windows system shuts down.

- **i5/OS:**

- If you started the connector using the Console, or using the “submit_adapter.sh” script in QSHELL, then you can use one of two methods to stop the connector:
- From the Windows system where the WebSphere Business Integration Server Express Console is installed, select **IBM WebSphere Business Integration Express> Toolset Express>Administrative>Console**. Then specify the i5/OS system name or IP address and a user profile and password that has *JOBCTL special authority. Select the iSeries adapter from the list and select the Stop button. Use the CL Command WRKACTJOB SBS (QWIBSVR44) to display the jobs to the Server Express product. Scroll the list to find the job with the jobname that matches the job description for the connector. Use the CL Command WRKSMBJOB to display the jobs; the job name is QWBIISRSC. Select Option 4 on this job, and press F4 to get the prompt for the ENDJOB command. Then specify *IMMED for the Option parameter and press enter.

Note: The connector will end when the QWBISVR44 subsystem has ended.

- If you used the start_connName.sh script to start the adapter from QSHELL, press F3 to end the connector. You can also stop the agent, by using a script named `stop_adapter.sh` located in the `/QIBM/ProdData/WBIServer44/bin` directory.

- **Linux:**

Connectors run in the background so they have no separate window. Instead, run the following command to stop the connector:

```
connector_manager -stop connName
```

where *connName* is the name of the connector.

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

- **For Windows Platforms:**

ProductDir\connectors\connectorInstance

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

where *connectorInstance* uniquely identifies the connector instance.

You can specify the InterChange Server Express server name as a parameter of startup.bat; an example is: start_iseries.bat connName serverName.

- **For i5/OS Platforms:**

/QIBM/UserData/WBIServer44/WebSphereICSName/connectors/connectorInstance

where connectorInstance uniquely identifies the connector instance and where WebSphereICSName is the name of the Interchange Server Express instance with which the connector runs.

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:

/QIBM/UserData/WBIServer44/WebSphereICSName/repository
/connectorInstance where WebSphereICSName is the name of the Interchange Server Express

instance with which the connector runs.

- **For Linux Platforms:**

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. You can specify the InterChange

Server Express servername as a parameter of connector_manager; an example is connector_manager -start connName WebSphereICSName [-cConfigFile].

Create business object definitions

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

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

Use the following steps to create a configuration file (connector definition) for the connector instance in Connector Configurator Express.

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 business object definitions" on page 25.
3. (For Windows only.) Create a startup script shortcut.
4. (For Windows only.) 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.
5. (For i5/OS only.) Create a job description for the connector using the information below:
CRTDUPOBJ(QWBII SRSC) FROMLIB(QWBISVR44)OBJTYPE(*JOB D)TOLIB(QWBISVR44) NEWOBJ(newemailname) where newemailname is a 10-character name that you use for the job description for your new connector.
6. (For i5/OS only.) Add the new connector to the WebSphere Business Integration Server Express Console. For information about the WebSphere Business Integration Server Express console, refer to the online help provided with the Console.

Chapter 4. Understanding business objects for the connector

This chapter describes the structure of iSeries business objects, how the connector processes the business objects, and the assumptions the connector makes about them. Use this information as a guide to modifying existing business objects for iSeries or as suggestions for implementing new business objects.

This chapter contains the following sections:

- “Defining connector metadata”
- “Business Object structure for RPG programs” on page 28
- “Business Object structure for iSeries data queues” on page 32
- “Configuring meta objects for polling” on page 33
- “Specifying business object attribute properties” on page 34
- “Specifying business object attribute level application text” on page 35

For information on the Object Discovery Agent (ODA) utility that automates the creation of business objects for the adapter for iSeries, see Chapter 5, “Creating and modifying business objects,” on page 37.

Defining connector metadata

The iSeries connector is metadata-driven. In the WebSphere Business Integration system, metadata is application-specific information stored in a business object that helps the connector interact with the application. A metadata-driven connector handles each business object that it supports based on the metadata encoded in the business object definition, rather than on instructions hardcoded in the connector. Business object metadata includes the structure of the business object, the settings of its attribute properties, and the content of its application-specific information. Because the connector is metadata-driven, it can handle new or modified business objects without requiring modifications to the connector code.

The connector makes assumptions about the structure of its supported business objects and the format of the application-specific information.

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

Overview of business object structure

In the WebSphere Business Integration system, a business object definition consists of a type name, supported verbs, and attributes. An application business object is an instance of a business object definition. It reflects a specific application’s data structure and attribute properties.

Some attributes, instead of containing data, point to child business objects or arrays of child business objects that contain the data for these objects.

WebSphere Business Integration Adapter business objects can be flat or hierarchical. A flat business object contains only simple attributes, that is, attributes

that represent a single value (such as a String). A hierarchical business object contains both simple attributes and child business objects or arrays of child business objects that contain the values.

A cardinality 1 container object, or single-cardinality relationship, occurs when an attribute in a parent business object contains a single child business object. In this case, the child business object represents a collection that can contain only one record. The type of the attribute is the same as that of the child business object.

A cardinality n container object, or multiple-cardinality relationship, occurs when an attribute in the parent business object contains an array of child business objects. In this case, the child business object represents a collection that can contain multiple records. The type of the attribute is the same as the type of the array of child business objects.

A hierarchical business object can have simple attributes and can also have attributes that represent either a single-cardinality child business object or an array of child business objects. In turn, each of these business objects can contain single-cardinality child business objects and arrays of business objects, and so on.

Business Object structure for RPG programs

The business object for the iSeries adapter is a flat business object. The attributes can be input, output or inout parameters. One of the attributes needs to be a key for the Business Object Designer Express.

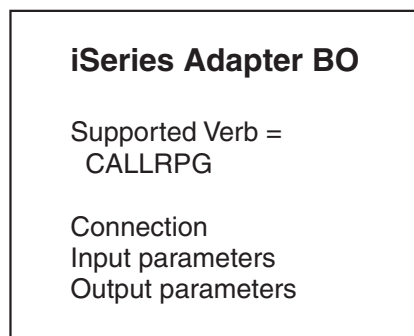


Figure 3. The iSeries parent business object

Also, there is a child attribute of type Connection. This contains information about the connection to the i5/OS machine; the HostName, UserName and Password. These attributes are all mandatory and thus `is Required` is set true for all of them. This Connection business object is a child attribute to all the iSeries business objects.

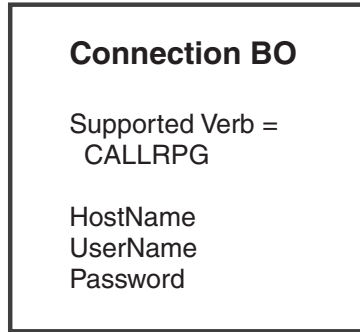


Figure 4. The iSeries child business object

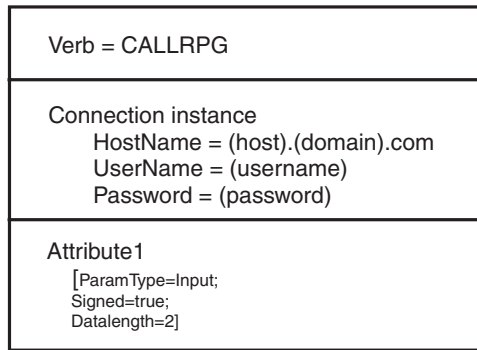


Figure 5. Example of a RPG business object

Note: The information within brackets in the above diagram represents the application-specific information.

The RPG business object consists of the verb CALLRPG and the program name (the full path of the IFSFile) is set as the verb's ASI. Apart from the Connection child attribute, the RPG business object has attributes corresponding to the RPG program parameters. The Name of the attribute is same as the name of the corresponding parameter specified in the source program. The MaxLength property of the attributes (representing an RPG Parameter) is derived from the corresponding parameter length specified in the PARM Spec of the source program. If the parameter is a number, the corresponding decimal part's length is also mentioned in the ASI as DecimalPositions=n and packedDec=true.

The adapter can be used to call a PGM multiple times with a single Request business object with multiple instances. An example is shown in Figure 6.

		General	Attributes								
	Pos	Name	Type	Key	Foreign Key	Required	Cardinality	Maximum Length	Default Value	Application Specific Information	
1	1	<input type="checkbox"/> Connection	Connection	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1			ParamType=Input	
1.1	1.1	UserName	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255	Rajesh		
1.2	1.2	Password	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255			
1.3	1.3	HostName	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255	abc.in.ibm.com		
1.4	1.4	ObjectEventId	String								
2	2	<input type="checkbox"/> MultiRecord	multi_child	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	N			NumRecords=4	
2.1	2.1	NAME1	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		ParamType=OUTPUT;DataLength=10	
2.2	2.2	SERIAL1	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		ParamType=OUTPUT;DataLength=10	
2.3	2.3	DEPT1	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		ParamType=OUTPUT;DataLength=10	
2.4	2.4	ObjectEventId	String								
3	3	ObjectEventId	String								
4	4			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255			

Figure 6. Single Request business object with multiple instances

Figure 7 shows the Parent business object with connection information and the Child business object with the PGM's parameter information. The connection information is available in the Connection business object and the Verb ASI is the same as the path of the program to be called.

		General	Attributes
Business Object Level Application-specific information:			
Supported Verbs:			
	Name	Application-specific information	
1	CALLRPG	/QSYS.LIB/PNPLIB.LIB/PRPG4.PGM	
2			

Figure 7. Parent business object with connection information

The child business object is shown in Figure 8 with attributes corresponding to parameter information of the program.

		General	Attributes								
	Pos	Name	Type	Key	Foreign Key	Required	Cardinality	Maximum Length	Default Value	Application Specific Information	
1	1	[-] Connection	Connection	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1			ParamType=Input	
1.1	1.1	UserName	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255	Rajesh		
1.2	1.2	Password	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255			
1.3	1.3	HostName	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255	abc.in.ibm.com		
1.4	1.4	ObjectEventId	String								
2	2	[-] MultiRecord	multi_child	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	N			NumRecords=4	
2.1	2.1	NAME1	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		ParamType=OUTPUT;DataLength=10	
2.2	2.2	SERIAL1	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		ParamType=OUTPUT;DataLength=10	
2.3	2.3	DEPT1	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		ParamType=OUTPUT;DataLength=10	
2.4	2.4	ObjectEventId	String								
3	3	ObjectEventId	String								
4	4			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255			

Figure 8. Child business object with corresponding parameter information

The example in Figure 9 shows a Business object with 2 instances of multi_child Business Object, thus the program will be executed twice. Make sure you use the appropriate XSD files for Multi record retrieval.

[-] Connection	Connection	
UserName	String	Rajesh
Password	String	password
HostName	String	abc.in.ibm.com
ObjectEventId	String	
[-] MultiRecord[n]	multi_child	
[-] [0]	multi_child	
NAME1	String	RAJ
SERIAL1	String	10
DEPT1	String	A
ObjectEventId	String	
[-] [1]	multi_child	
NAME1	String	PRABHU
SERIAL1	String	11
DEPT1	String	B
ObjectEventId	String	
ObjectEventId	String	

Figure 9. Business object with 2 instances of a multi-child business object

Connector business object processing

The connector passes business objects between the integration broker, InterChange Server Express, and the i5/OS system.

When the integration broker passes a business object to the connector, the connector performs the following operations:

1. Uses the information in the connection child attribute to connect to the i5/OS system.
2. Builds the parameter list for the RPG program based on the attributes in the business object.
3. Executes the RPG program corresponding to the business object.
4. Returns the result of executing the program: success or failure.

Use Business Object Designer Express ODA to create the business objects. Create the business object definition, and add the required attributes. Then configure the connector to support the business object. For more information regarding the Business Object Designer Express ODA, see Chapter 5, “Creating and modifying business objects,” on page 37.

Business Object structure for iSeries data queues

For the data queues business object, the attributes represent the data queue fields. Apart from these, it has a child Connection attribute. There can be parent-child kind of relationship if there is a AS400Structure as part of the data queue fields. The valid verbs supported are GetQueue and PutQueue. The application-specific information for the queues will be at the business object level. The value will be the absolute IFSFile path of the data queue. The total length of the attributes should be equal to the maximum length of an element in the queue. This value is defined when you create the queue on the iSeries machine.

The parameter type can be Input, Output or InOut. Both the Connection Object and all its attributes are set as required. The iSeries ODA generates all the attributes with ASI as ParamType=InOut by default. The user can however change them to Input or Output after making sure the change is validated against the program logic.

See Figure 10 for an example of a data queue business object.

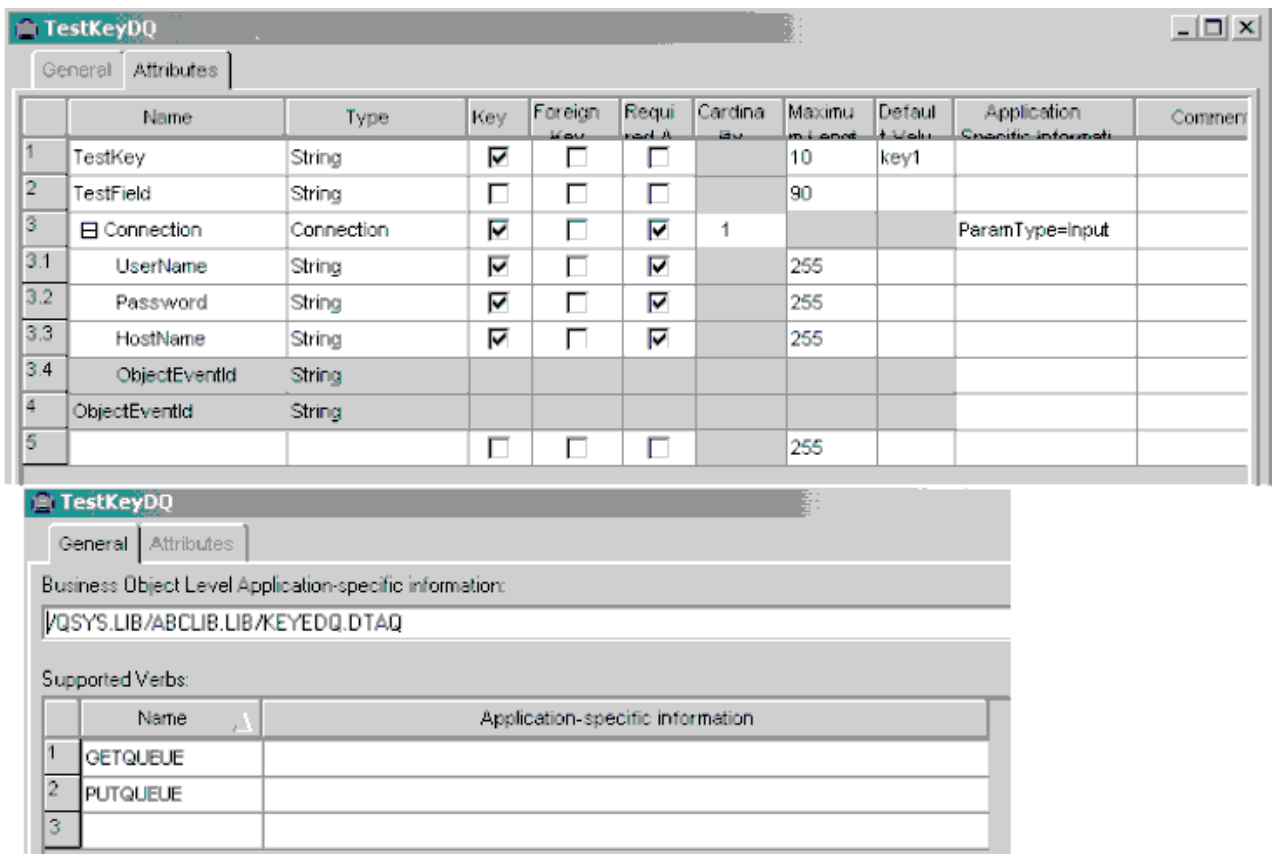


Figure 10. Data queue business object

Configuring meta objects for polling

The iSeries adapter uses meta-objects for all the data queues that receive complete data queue message for changes to either the database or to the data files. You need to configure the meta-objects for all the data queues that receive complete data queue messages.

The meta-object name always starts with MO_iSeries. Each meta-object holds information about the data queues. You need to add a dummy verb to all the meta-objects.

The attributes (hostname, username, and password) within the meta objects have a static default value. These default attributes cannot be changed dynamically once the connector is started, since these values are cached in the connector agent. To access the same data queue on a different machine, you must either change the default value and restart the iSeries adapter instance or configure another meta-object for the new machine information.

The verb in the Business object is set to appropriate string, while the DataQueueName attribute is set to the IFS File path of the data queue. The BusObjName attribute contains the name of the corresponding Business object (which contains the verb mentioned in the metaobject). The details read from the queue are filled in this Business object. The attributes of the meta object are shown in the example below (Figure 11). The corresponding SamplePollBO is shown in Figure 12.

	Name	Type	Key	Foreign Key	Required Attrib	Cardinality	Maximum Length	Default Value	Applic Specific Inform
1	UserName	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	user	
2	Password	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	passwd	
3	HostName	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	iseries.server.com	
4	Verb	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	Getqueue	
5	DataQueueName	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	/QSYS.LIB/PLIB.LIB/TESTDQ1.DTAQ	
6	BusObjName	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	SamplePollBO	
7	ObjectEventId	String							
8			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		

Figure 11. Example poll business object for a sequential data queue

	Pos	Name	Type	Key	Foreign Key	Required Attrib	Cardinality	Maximum Length	Default Value	Application Specific Inform
1	1	TestField	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		100		
2	2	ObjectEventId	String							
3	3			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		

Figure 12. Corresponding SamplePollBO

For keyed data queues, a key attribute called "key" will be added to the meta object as shown in Figure 13. The poll call will use the key information to get the corresponding message from the keyed data queue.

	Type	Key	Foreign Key	Required Attribute	Cardinality	Maximum Length	Default Value	Application Specific Information	Comments
1	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	user		
2	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	passwd		
3	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	iseries.machine.com		
4	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	Getqueue		
5	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	IGSYS.LIB/PNPUB.LIB/TESTDQ2KEY.DTAQ		
6	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	SamplePollKey		
7	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	Key1		
8	String								
9		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255			

Figure 13. Example poll business object for keyed data queue

The corresponding example poll business object for keyed data queue is shown in Figure 14.

	Pos	Name	Type	Key	Foreign Key	Required Attribute	Cardinality	Maximum Length	Default Value	Application Specific Information
1	1	TestKey	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		10		
2	2	TestField	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		90		
3	3	ObjectEventId	String							
4	4			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		

Figure 14. Sample poll business object for a keyed data queue

Specifying business object attribute properties

The iSeries connector has various properties that you can set on its business object attributes. This section describes how the connector interprets several of these properties and describes how to set them when modifying a business object.

The following table shows the properties for simple attributes.

Table 4. Business object attribute properties

Attribute	Description
Name	Unique name of the attribute
Type	All simple attributes should be of type String
MaxLength	If the attribute value length is greater than the MaxLength specified for the attribute and the attribute represents an input parameter, the value is trimmed to the Maxlength value. If the value length is less than MaxLength, then spaces are padded.

Table 4. Business object attribute properties (continued)

Attribute	Description
IsKey	Not used
IsForeignKey	Not used
IsRequired	All the input parameters need to have this attribute set to true.
AppSpecInfo	ParamType=<value>;Offset=<value>;Signed=<True/False>;DataLength=<value>;PackedDec=<True/False>;ZonedDec=<True/False>;DecimalPositions=<value>
DefaultValue	If set for the attribute, this value shall be used by the connector if one is not set for the input parameter.

Specifying business object attribute level application text

The following information is part of the business object attribute -level application text.

Table 5. Business object attributes

Property	Values	Description
ParamType	Input/Output/InOut	Indicates what type of parameter the attribute represents.
Offset	Any integer value	Indicates the offset in the byte array from where the parameter value starts.
Signed	True/false	For integer/short/long types, the property indicates if it is signed. If not set, the value is taken as unsigned.
DataLength	Any integer value	Applies to integer/short/long types. Used to distinguish the data length for the signed/unsigned types. If not set, a default of 4 is assumed.
DecimalPositions	Any integer value	Applies for zoned decimal and packed decimal types. The value represents the number of decimal positions.
PackedDec	True/false	If set to true, the attribute represents a packed decimal.
ZonedDec	True/false	If set to true, the attribute represents a zoned decimal.

Data conversion from the iSeries or i5/OS toolbox

The Toolbox for iSeries/AS400 has data conversion classes included. The following table matches the iSeries/AS400 data types with the corresponding IBM WebSphere Business Integration datatypes, along with the data conversion class used.

Table 6. Conversion datatypes and classes

iSeries/AS400 data type	IBM WBI datatype	Data conversion class
Signed two byte i5/OS number.	Integer - The app specific info - Signed=true; DataLength=2	AS400Bin2
Signed four byte i5/OS number.	Integer - The app specific info - Signed=true; DataLength=4	AS400Bin4
Signed two byte i5/OS floating point number.	Float	AS400Float4
Signed four byte i5/OS floating point number.	Double	AS400Float8
Unsigned two byte i5/OS number.	Integer - the app specific info Signed=false; DataLength=2	AS400UnsignedBin2
Unsigned four byte i5/OS number.	Integer - the app specific info Signed=false; DataLength=4	AS400UnsignedBin4
Packed-Decimal i5/OS number.	String - the MaxLength attribute property needs to have the number of digits. App specific info - DecimalPositions=<number of decimal positions>; PackedDec=true	AS400PackedDecimal
Zoned-Decimal i5/OS number.	String - the MaxLength attribute property needs to have the number of digits. App specific info - DecimalPositions=<number of decimal positions>; ZonedDec=true	AS400ZonedDecimal
Character data	String - MaxLength has the maximum length for the character data.	AS400Text
Date data	String - MaxLength has the maximum length for the date data.	AS400Text

Chapter 5. Creating and modifying business objects

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

This chapter contains the following sections:

- “Overview of the ODA for iSeries”
- “Generating business object definitions”
- “Specifying business object information” on page 43
- “Uploading business objects” on page 45

Overview of the ODA for iSeries

An Object Discovery Agent (ODA) enables you to generate business object definitions. A business object definition is a template for 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 Express provides a graphical interface to access the Object Discovery Agent and to work with it interactively.

The ODA for iSeries generates business object definitions for accessing RPGLE programs and data queue objects on the iSeries system. The Business Object Designer Express wizard automates the process of creating these definitions. You use the ODA to create business objects and Connector Configurator Express to configure the connector to support them.

Generating business object definitions

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

Starting the iSeries ODA

You can start the iSeries ODA using one of the following scripts:

- Windows – start_iSeriesODA.bat

Note: You can also start the iSeries ODA using the shortcut that the Installer automatically creates for Windows environments.

- Linux – start_iSeriesODA.sh
- i5/OS - start_iSeriesODA.sh

You select, configure, and run the iSeries ODA using Business Object Designer Express. Business Object Designer Express locates each ODA by the name specified in the AGENTNAME variable of each script or batch file.

Running Business Object Designer Express

Business Object Designer Express provides a wizard that guides you through the steps to generate a business object definition using the ODA.

Selecting the agent

You must first select the ODA agent.

1. Open Business Object Designer Express.
2. Click **File > New Using ODA**. The Business Object Wizard - Step 1 of 6 - Select Agent window opens.
3. Select the ODA/AGENTNAME (from the start_iSeriesODA script) in the Located agents list and click Next. (You may have to click Find Agents if desired agent is not listed.)

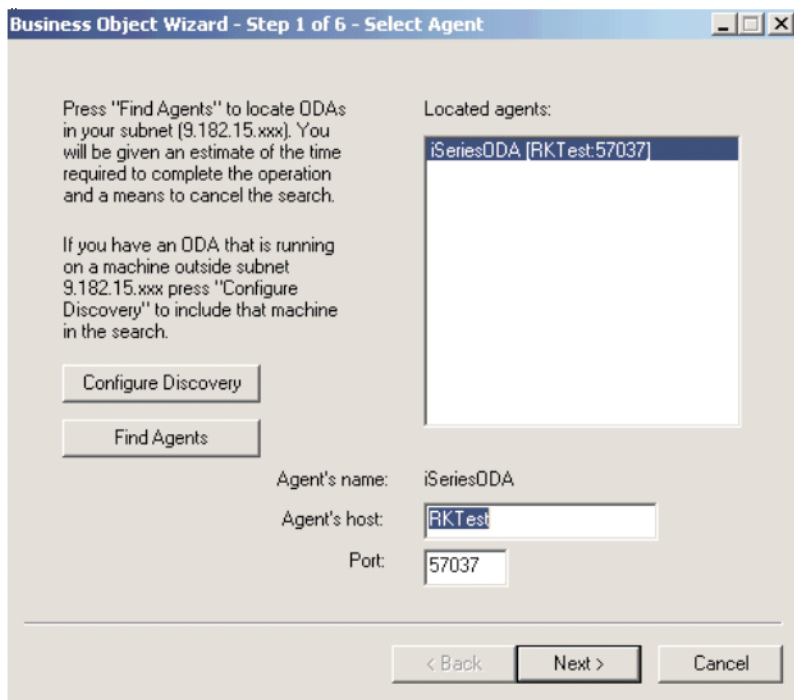


Figure 15. Select Agent window

Configuring the agent

After you click Next on the Select Agent window, the Business Object Wizard - Step 2 of 6 - Configure Agent window opens.

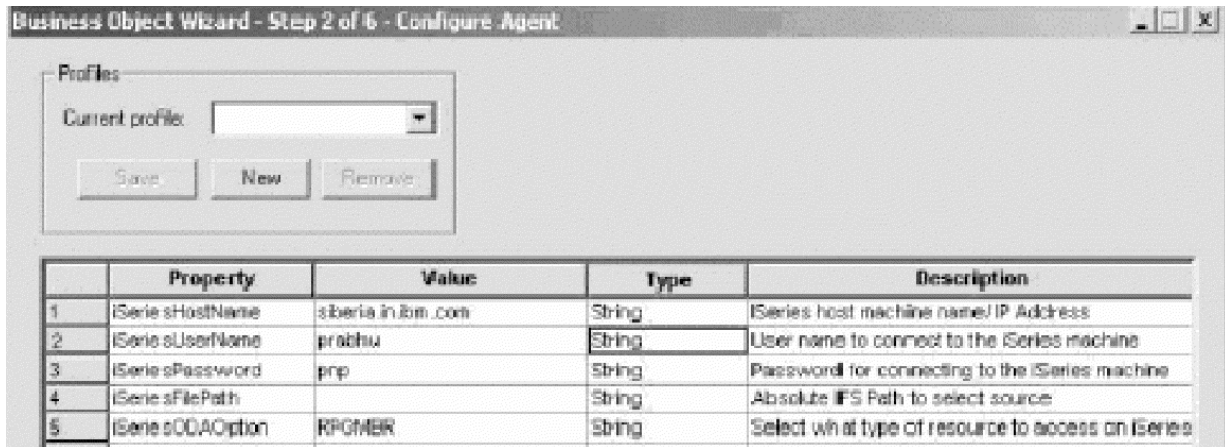


Figure 16. Configure Agent window

The properties you set on this screen are described in Table 5. You can save all the values you enter on this screen to a profile. Instead of retyping the property data the 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.

Table 7. Configure Agent properties

Property name	Default value	Type	Description
iSeriesHostName		String	(required) The iSeries host machine name.
iSeriesUserName		String	(required) The User name used to connect to the iSeries machine.
iSeriesPassword		String	(required) The Password used for connecting to the iSeries machine.
iSeriesFilePath	/QSYS.LIB/	String	Absolute IFS path to the select source. You must enter the correct path to retrieve the object. For example, for RPG source member, enter /QSYS.LIB/PNPLIB.LIB/QRPGSRC.FILE or for Data queue, enter /QSYS.LIB/PNPLIB/

Table 7. Configure Agent properties (continued)

Property name	Default value	Type	Description
iSeriesODAOption		String	The type of resource to access on iSeries. Currently there are 2 options: RPGMBR and DTAQ. Based on this selection, the source files are accessed to create corresponding business object definitions
TraceFileName	None	String	The name of the trace message file.
TraceLevel	5	Integer	(required) The tracing level (from 0 to 5) for the agent.
MessageFile	None	String	(required) The name of the message file that contains all the messages displayed by the ODA. If you do not correctly specify the name of the message file, the ODA will run without messages.

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. Type the value of each property, as defined in Table 7 on page 39.

Whenever any required field is left blank or during any error, (for example an invalid username), a pop up message is displayed with the corresponding error message.

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.

Selecting a business object

The Business Object Wizard - Step 3 of 6 - Select Source window opens, as illustrated in Figure 17.

This screen lists either the *.MBR for RPGLE source files or *.DTAQ files for data queues, where users can select the names of those files. The file type is decided by the iSeriesODAOption Agent property. The IFS Directories are represented as expandable tree nodes while the source names (MBRs and DTAQ) are shown as leaf nodes. You can select more than one source (leaf nodes only) either in the same IFS directory or in different IFS directories. Use this screen to select any number of source files for which the ODA will generate business object definitions.

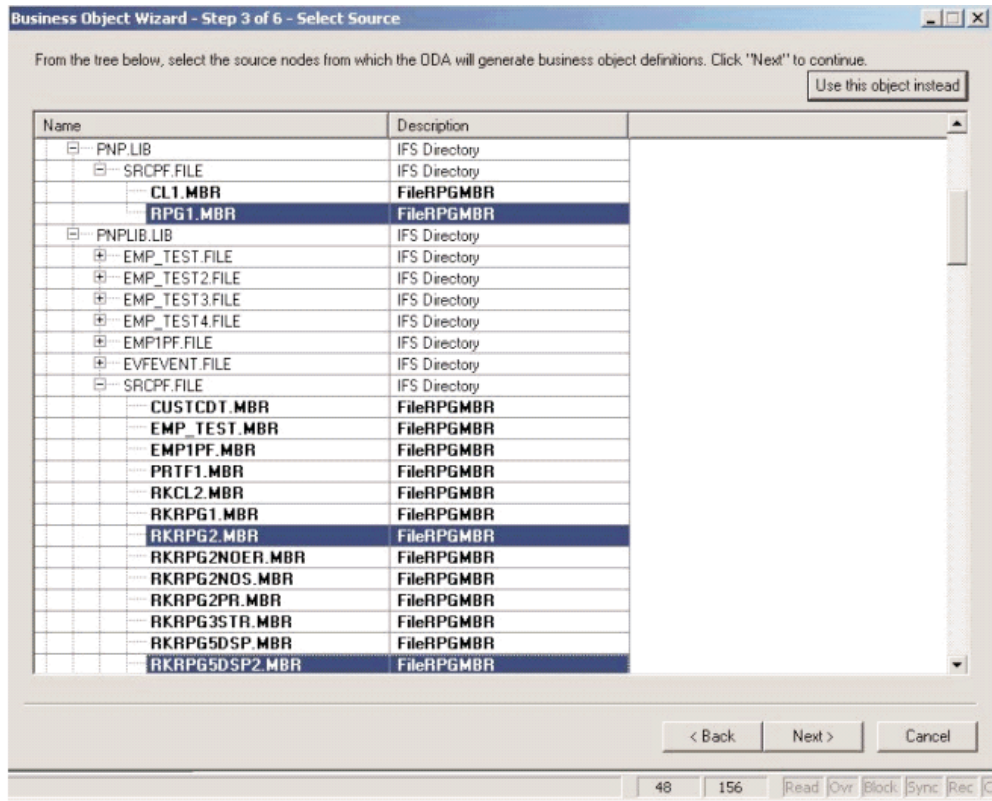


Figure 17. Select Source window

1. If necessary, expand a node to see a list of sub nodes.
2. Select the source file(s) you want to use. Click Next.
3. To select multiple nodes, see the *Business Object Development Guide* for more information regarding tree structures.

Confirming the object selection

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

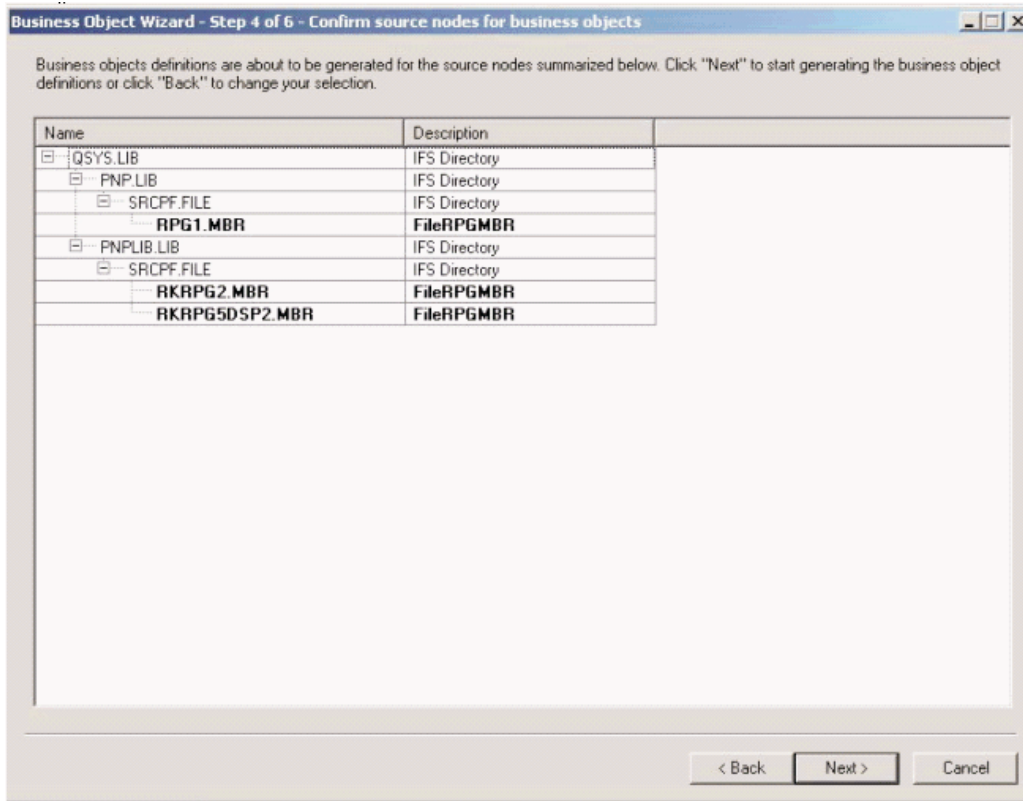


Figure 18. Confirm source node window

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... window opens with a message stating that the wizard is generating the business objects.

Generating the business objects

After you confirm your node sources, the iSeries ODA generates the business objects. The Business Object Wizard - Step 6 of 6 - Saving business object definitions... window opens.

1. Check either window Save a copy of the business object definitions to a separate file, or check Open the new business object definitions in separate windows. The latter choice launches the Business Object Designer Express and opens the business objects in that application.
2. If you are finished and want to close the ODA, check Shutdown ODA and click Finish.

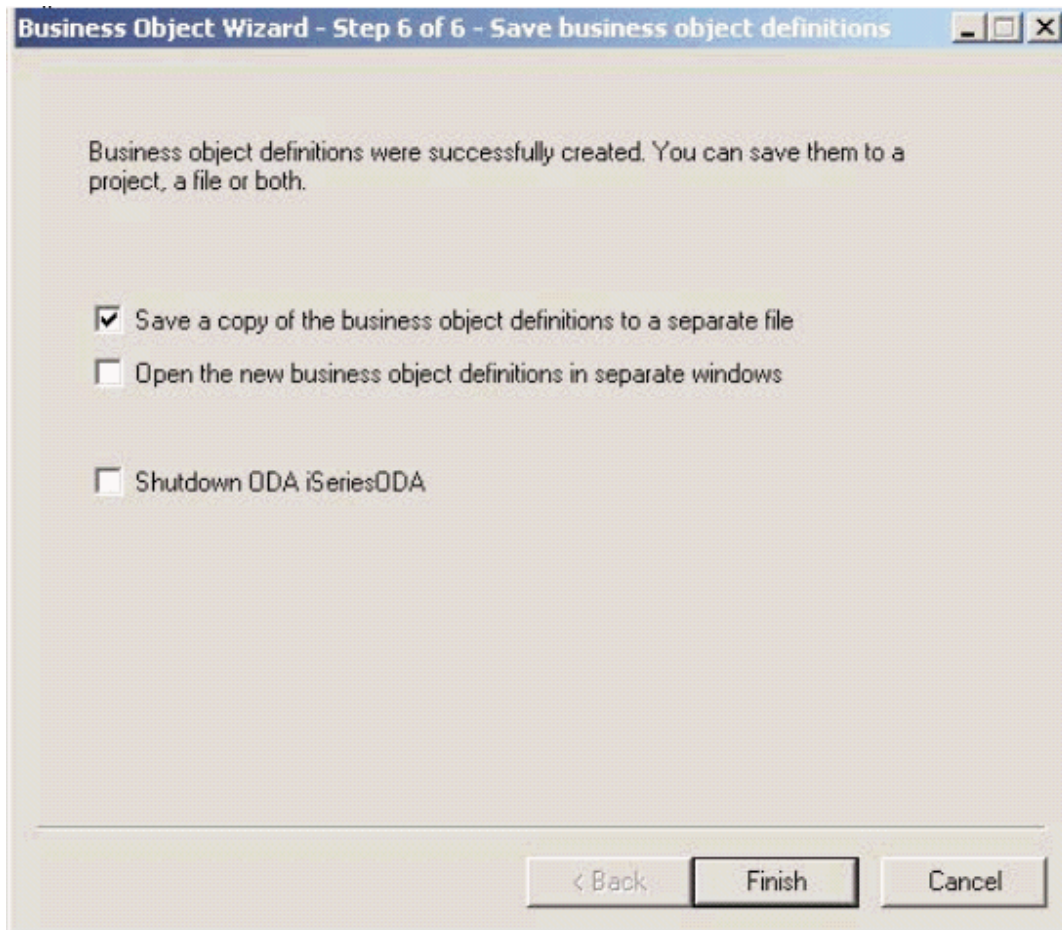


Figure 19. Save business objects window

Specifying business object information

After you create a business object, you can specify the business object-level ASI and the attribute-level ASI.

This section describes how to specify this information using the ODA with Business Object Designer Express. For a detailed description of these categories of information and what they mean for business object structure in the iSeries connector, see Chapter 4, “Understanding business objects for the connector,” on page 27.

Specifying the attribute-level ASI

Business Object Designer Express displays the attributes for the business object. For details about the attribute-level ASI in the iSeries connector, see “Specifying business object attribute level application text” on page 35.

The attributes are listed on the Attributes tab in the order in which they appear in the business object structure, as defined by the numeric value in the Pos column.

General		Attributes								
Pos	Name	Type	Key	Foreign Key	Required	Cardinality	Maximum Length	Default Value	Application Specific Information	
1	1	[-] Connection	Connection	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1			ParamType=Input
1.1	1.1	UserName	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	255	Rajesh		
1.2	1.2	Password	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	255			
1.3	1.3	HostName	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	255	abc.in.ibm.com		
1.4	1.4	ObjectEventId	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
2	2	[-] MultiRecord	multi_child	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	N			NumRecords=4
2.1	2.1	NAME1	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	255			ParamType=OUTPUT;DataLength=10
2.2	2.2	SERIAL1	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	255			ParamType=OUTPUT;DataLength=10
2.3	2.3	DEPT1	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	255			ParamType=OUTPUT;DataLength=10
2.4	2.4	ObjectEventId	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
3	3	ObjectEventId	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
4	4			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	255			

Figure 20. Setting the Attribute ASI

For each attribute, the window provides the name of the attribute, its type, and the ASI information. On this window, you need to specify a key (required by Business Object Designer Express to validate and save a business object) for each business object for which the ODA has not already specified a key.

In the business objects, passwords are not set as default value and not traced due to security reasons. The Business object generated for a sequential data queue has one attribute corresponding to its data length. For keyed data queues, there are 2 attributes of which the first one corresponds to the key while the other has the remaining length i.e. Data Length - Key Length. For a business object with an attribute with ASI DataLength = X and DecimalPositions=Y and when a BO is sent with the value of that attribute with greater than X digits (with the number of digits in Decimal Positions are greater than Y) the Connector will truncate this value to a length of decimal part to Y and maintain the Data Length to X and process the Business object successfully. If the integer part exceeds the length X-Y then Connector should throw an error. For example, if the ASI for an attribute is PackedDecimal=True;Datalength=10;DecimalPositions=2, then the values 112345678, 12345678.1, and 12345678.12 are accepted, but the extraneous decimal digits after the maximum limit of 2 in 12345678.123 will be truncated, and the value is taken as 12345678.12. A trace message is given for it as "Truncated String Value:<12345678.12> for DecimalPositions=2". The value 123456789.12 will log an error: "Length is not valid."

You can also use this window to set child object keys as needed and specify the following information:

- Is the attribute required for the connector to process the business object? If so, click the Required check box.
- Is the maximum length of the attribute different from the value that appears in the Maximum Length column.
- Does the attribute have a default value? If so, type the value in the Default column.

Specifying the business object-level ASI

After specifying the attribute-level ASI, you can view and modify the business object-level ASI. For details about business object-level ASI, see "Specifying business object attribute level application text" on page 35.

The business object-level ASI is listed on the General tab. The ASI value that appears in the field Business Object Level Application-specific information contains

the name of the proxy class that represents this business object. The connector uses this information to map a proxy class to a business object, and, in the case of a server-side business object (when the connector also runs as a server), the connector uses this information to map an implementation class to a business object.

This screen also lists all the verbs that are supported by the business object and provides the ASI for each verb. On this screen you can modify the ASI of the business object and its supporting verbs.

Note: You must check and change the PGM's path mentioned in the Verb-level ASI. The ODA assumes that the selected .MBR IFS file is a valid RPGLE source file. If it is not a RPGLE source file, the ODA generates a business object with no input parameters.

The screenshot shows a software interface with two tabs: 'General' (selected) and 'Attributes'. Below the tabs is a section titled 'Business Object Level Application-specific information:' followed by a text input field. Below that is a section titled 'Supported Verbs:' followed by a table.

	Name ▾	Application-specific information
1	CALLRPG	/QSYS.LIB/PNPLIB.LIB/PRPG4.PGM
2		

Figure 21. Setting the business object-level ASI

Uploading business objects

The newly created business object definition files must be uploaded to the integration broker, InterChange Server Express, once they have been created.

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 *System Implementation Guide*.

Chapter 6. Troubleshooting and error handling

This chapter describes how the adapter for iSeries handles errors. The adapter generates logging and tracing messages. The chapter contains the following sections:

This chapter contains the following sections:

- “Error handling”
- “Logging”
- “Tracing messages”

Error handling

All error messages generated by the connector are stored in a message file named `BIA_ISERIESAdapter.txt`. (The name of the file is determined by the `LogFile` standard connector configuration property.)

All errors translate to `VerbProcessingFailedException`.

Logging

The adapter logs an error message whenever it encounters an abnormal condition during processing, regardless of the trace level. When such an error occurs, the connector also prints a textual representation of the failed business object as it was received. It writes the text to the iSeries Adapter log file, whose file name corresponds to the connector property `LogFile`. The message contains a detailed description of the condition and the outcome and may also include extra information that may aid in debugging, such as business object dumps or stack traces (for exceptions).

The message IDs for the iSeries adapter range from 93000 to 94000.

Tracing messages

Tracing is an optional debugging feature you can turn on to closely follow a connector’s behavior. Tracing messages are configurable and can be changed dynamically. You set various levels depending on the desired detail. Trace messages, by default, are written to `STDOUT`.

You can also configure tracing to write to a file. The following table describes the types of tracing messages that the iSeries connector outputs at each trace level. All the trace messages appear in the file specified by the connector property `TraceFileName`. These messages are in addition to any tracing messages output by the IBM WebSphere Business Integration Adapter architecture. For more on configuring trace messages, see the connector configuration properties in “Configuring the connector.” For more information on tracing, including how to enable and set it, see the Connector Development Guide.

Table 6 lists the recommended content for connector tracing message levels.

Table 8. Tracing messages content level description

Tracing level	Tracing messages
Level 0	Use this trace level for trace messages that identify the connector version. No other tracing is performed at this level.
Level 1	N/A
Level 2	Use this trace level for trace messages that: <ul style="list-style-type: none"> • Identify the BO handler used for each object that the connector processes. • Log each time a business object is posted to the integration broker • Indicate each time a request business object is received
Level 3	N/A
Level 4	Use this trace level for trace messages that: <ul style="list-style-type: none"> • Identify application-specific information. Examples of this include the values returned by the methods that process the application-specific information fields in business objects. • Identify when the connector enters or exits a function. These messages help trace the process flow of the connector. • Record any thread-specific processing. For example, if the connector spawns multiple threads, a message logs the creation of each new thread.
Level 5	Use this trace level for trace messages that: <ul style="list-style-type: none"> • Indicate connector initialization. This type of message can include, for example, the value of each connector configurator property that has been retrieved from the broker. • Detail the status of each thread that the connector spawns while it is running. • Represent statements executed in the application. The connector log file contains all statements executed in the target application and the value of any variables that are substituted, where applicable. • Record business object dumps. The connector should output a text representation of a business object before it begins processing (showing the object that the connector receives from the collaboration) as well as after it finishes processing the object (showing the object that the connector returns to the collaboration).

Appendix. Standard configuration properties for connectors

This appendix describes the standard configuration properties for the connector component of WebSphere Business Integration Server Express adapters. The information covers InterChange Server Express.

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

New properties

These standard properties have been added in this release:

- AdapterHelpName
- ControllerEventSequencing
- jms.ListenerConcurrency
- jms.TransportOptimized
- TivoliTransactionMonitorPerformance

Standard connector properties overview

Connectors have two types of configuration properties:

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

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

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

Starting Connector Configurator Express

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

Connector Configurator Express and System Manager run only on the Windows system. If you are running the connector on a Linux system, you must have a Windows machine with these tools installed.

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

Configuration property values overview

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

1. Default

2. Repository for InterChange Server Express integration broker.
3. Local configuration file
4. Command line

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

A connector obtains its configuration values at startup. If you change the value of one or more connector properties during a run-time session, the property's update method determines how the change takes effect.

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

There are four update methods for standard connector properties:

- **Dynamic**
The new value takes effect immediately after the change is saved in System Manager. However, if the connector is in stand-alone mode (independently of System Manager).
- **Agent restart (InterChange Server Express only)**
The new value takes effect only after you stop and restart the connector agent.
- **Component restart**
The new value takes effect only after the connector is stopped and then restarted in System Manager. You do not need to stop and restart the agent or the server process.
- **System restart**
The new value takes effect only after you stop and restart the connector agent and the server.

To determine how a specific property is updated, refer to the **Update Method** column in the Connector Configurator Express window, or see the Update Method column in Table 9 on page 51.

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

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

Standard properties quick-reference

Table 9 provides a quick-reference to the standard connector configuration properties. Not all connectors require all of these properties, and property settings may differ. .

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

Note: In the Notes column in Table 9, the phrase “RepositoryDirectory is set to <REMOTE>” indicates that the broker is InterChange Server Express.

Table 9. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdapterHelpName	One of the valid subdirectories in <ProductDir>\bin\Data\App\Help that has a valid <Regional Setting> directory	Template name, if valid, or blank field	Component restart	Supported regional settings. Include chs_chn, cht_twn, deu_deu, esn_esp, fra_fra, ita_ita, jpn_jpn, kor_kor, ptb_bra, and enu_usa (default).
AdminInQueue	Valid JMS queue name	<CONNECTORNAME>/ADMININQUEUE	Component restart	This property is valid only when the value of DeliveryTransport is JMS
AdminOutQueue	Valid JMS queue name	<CONNECTORNAME>/ADMINOUTQUEUE	Component restart	This property is valid only when the value of DeliveryTransport is JMS
AgentConnections	1 through 4	1	Component restart	This property is valid only when the value of DeliveryTransport is MQ or IDL, the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
AgentTraceLevel	0 through 5	0	Dynamic for ICS; otherwise Component restart	
ApplicationName	Application name	The value specified for the connector application name	Component restart	
BrokerType	ICS	ICS	Component restart	
CharacterEncoding	Any supported code. The list shows this subset: ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437 .	ascii7	Component restart	This property is valid only for C++ connectors.
CommonEventInfrastructure	true or false	false	Component restart	

Table 9. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
CommonEventInfrastructureURL	A URL string, for example, corbaloc:iiop:host:2809.	No default value.	Component restart	This property is valid only if the value of CommonEventInfrastructure is true.
ConcurrentEventTriggeredFlows	1 through 32,767	1	Component restart	This property is valid only if the value of RepositoryDirectory is set to <REMOTE> and the value of BrokerType is ICS.
ContainerManagedEvents	Blank or JMS	Blank	Component restart	This property is valid only when the value of Delivery Transport is JMS.
ControllerEventSequencing	true or false	true	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ControllerStoreAndForwardMode	true or false	true	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ControllerTraceLevel	0 through 5	0	Dynamic	This property is valid only if the value of RepositoryDirectory is set to <REMOTE> and the value of BrokerType is ICS.
DeliveryQueue	Any valid JMS queue name	<CONNECTORNAME>/DELIVERYQUEUE	Component restart	This property is valid only when the value of Delivery Transport is JMS.
DeliveryTransport	IDL or JMS	IDL when the value of RepositoryDirectory is <REMOTE>, otherwise JMS	Component restart	If the value of RepositoryDirectory is not <REMOTE>, the only valid value for this property is JMS.
DuplicateEventElimination	true or false	false	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
EnableOidForFlowMonitoring	true or false	false	Component restart	This property is valid only if the value of BrokerType is ICS.
FaultQueue	Any valid queue name.	<CONNECTORNAME>/FAULTQUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory, CxCommon.Messaging.jms.SonicMQFactory, or any Java class name	CxCommon.Messaging.jms.IBMMQSeriesFactory	Component restart	This property is valid only if the value of DeliveryTransport is JMS.

Table 9. Summary of standard configuration properties (continued)

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

Table 9. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
LogAtInterchangeEnd	true or false	false	Component restart	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
MaxEventCapacity	1 through 2147483647	2147483647	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
MessageFileName	Valid file name	InterchangeSystem.txt	Component restart	
MonitorQueue	Any valid queue name	<CONNECTORNAME> /MONITORQUEUE	Component restart	This property is valid only if the value of DuplicateEventElimination is true and ContainerManagedEvents has no value.
OADAutoRestartAgent	true or false	false	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
OADMaxNumRetry	A positive integer	1000	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
OADRetryTimeInterval	A positive integer in minutes	10	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
PollEndTime	HH = 0 through 23 MM = 0 through 59	HH:MM	Component restart	
PollFrequency	A positive integer (in milliseconds)	10000	Dynamic if broker is ICS; otherwise Component restart	
PollQuantity	1 through 500	1	Agent restart	This property is valid only if the value of ContainerManagedEvents is JMS.
PollStartTime	HH = 0 through 23 MM = 0 through 59	HH:MM	Component restart	
RepositoryDirectory	<REMOTE> if the broker is ICS; otherwise any valid local directory.	For ICS, the value is set to <REMOTE>	Agent restart	
RequestQueue	Valid JMS queue name	<CONNECTORNAME> /REQUESTQUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS

Table 9. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
ResponseQueue	Valid JMS queue name	<CONNECTORNAME>/RESPONSEQUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
RestartRetryCount	0 through 99	3	Dynamic if ICS; otherwise Component restart	
RestartRetryInterval	A value in minutes from 1 through 2147483647	1	Dynamic if ICS; otherwise Component restart	
RHF2MessageDomain	mrm or xml	mrm	Component restart	This property is valid only if the value of DeliveryTransport is JMS and the value of WireFormat is CwXML.
SourceQueue	Any valid WebSphere MQ queue name	<CONNECTORNAME>/SOURCEQUEUE	Agent restart	This property is valid only if the value of ContainerManagedEvents is JMS.
SynchronousRequest Queue	Any valid queue name.	<CONNECTORNAME>/SYNCHRONOUSREQUEST QUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
SynchronousRequest Timeout	0 to any number (milliseconds)	0	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
SynchronousResponse Queue	Any valid queue name	<CONNECTORNAME>/SYNCHRONOUSRESPONSE QUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
TivoliMonitorTransaction Performance	true or false	false	Component restart	
WireFormat	CwXML or CwB0	CwXML	Agent restart	The value of this property must be CwXML if the value of RepositoryDirectory is not set to <REMOTE>. The value must be CwB0 if the value of RepositoryDirectory is set to <REMOTE>.
WsifSynchronousRequest Timeout	0 to any number (milliseconds)	0	Component restart	This property is not valid if the value of BrokerType is ICS..
XMLNameSpaceFormat	short or long	short	Agent restart	This property is not valid if the value of BrokerType is ICS.

Standard properties

This section describes the standard connector configuration properties.

AdapterHelpName

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

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

AdminInQueue

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

The default value is `<CONNECTORNAME>/ADMININQUEUE`

AdminOutQueue

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

The default value is `<CONNECTORNAME>/ADMINOUTQUEUE`

AgentConnections

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

The default value of this property is 1.

AgentTraceLevel

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

The default value is 0.

ApplicationName

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

The default is the name of the connector.

BrokerType

The BrokerType property identifies the integration broker type that you are using. The value is `ICS`, `,` or `.`

CharacterEncoding

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

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

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

ConcurrentEventTriggeredFlows

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

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

To implement concurrent processing for an entire flow (from a source application to a destination application), the following properties must be configured:

- The collaboration must be configured to use multiple threads by setting its `Maximum number of concurrent events` property high enough to use multiple threads.
- The destination application's application-specific component must be configured to process requests concurrently.

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

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

The default value is 1.

ContainerManagedEvents

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

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

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

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

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

- `MimeType = text/xml`

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

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

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

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

There is no default value.

ControllerEventSequencing

The ControllerEventSequencing property enables event sequencing in the connector controller.

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

The default value is true.

ControllerStoreAndForwardMode

The ControllerStoreAndForwardMode property sets the behavior of the connector controller after it detects that the destination application-specific component is unavailable.

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

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

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

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

The default value is true.

ControllerTraceLevel

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

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

The default value is 0.

DeliveryQueue

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

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

The default value is <CONNECTORNAME>/DELIVERYQUEUE.

DeliveryTransport

The DeliveryTransport property specifies the transport mechanism for the delivery of events. For Java Messaging Service, the value is JMS.

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

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

The default value is JMS.

JMS

The JMS transport mechanism enables communication between the connector and client connector framework using Java Messaging Service (JMS).

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

There may be a memory limitation if you use the JMS transport mechanism for a connector in the following environment when InterChange Server Express (ICS) is the integration broker.

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

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

This script is located in the `\bin` directory below the product directory (<ProductDir>). Using a text editor, add the following line as the first line in the CWSharedEnv.sh script:

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

This line restricts heap memory usage to a maximum of 768 MB (3 segments * 256 MB). If the process memory grows larger than this limit, page swapping can occur, which can adversely affect the performance of your system.

- Set the value of the IPCCBaseAddress property to 11 or 12. For more information on this property, see the *WebSphere Business Integration Server Express Installation Guide for Windows, for Linux, or for i5/OS*.

DuplicateEventElimination

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

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

The default value is false.

EnableOidForFlowMonitoring

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

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

The default value is false.

FaultQueue

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

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

jms.FactoryClassName

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

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

jms.ListenerConcurrency

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

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

The default value is 1.

jms.MessageBrokerName

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

When you connect to a remote message broker, this property requires the following values:

QueueMgrName:Channel:HostName:PortNumber

where:

QueueMgrName is the name of the queue manager.

Channel is the channel used by the client.
HostName is the name of the machine where the queue manager is to reside.
PortNumber is the port number used by the queue manager for listening

For example:

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

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

jms.NumConcurrentRequests

The `jms.NumConcurrentRequests` property specifies the maximum number of concurrent service call requests that can be sent to a connector at the same time. Once that maximum is reached, new service calls are blocked and must wait for another request to complete before proceeding.

The default value is 10.

jms.Password

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

There is no default value.

jms.TransportOptimized

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

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

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

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

The default value is `false`.

jms.UserName

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

There is no default value.

JvmMaxHeapSize

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

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

The default value is 128m.

JvmMaxNativeStackSize

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

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

The default value is 128k.

JvmMinHeapSize

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

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

The default value is 1m.

ListenerConcurrency

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

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

The default value is 1.

Locale

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

A locale name has the following format:

ll_TT.codeset

where:

ll is a two-character language code (in lowercase letters)

TT is a two-letter country or territory code (in uppercase letters)

codeset is the name of the associated character code set (may be optional).

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

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

The default value is en_US.

LogAtInterchangeEnd

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

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

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

The default value is false.

MaxEventCapacity

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

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

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

The default value is 2147483647.

MessageFileName

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

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

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

The default value is InterchangeSystem.txt.

MonitorQueue

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

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

The default value is <CONNECTORNAME>/MONITORQUEUE

OADAutoRestartAgent

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

This property must be set to true to enable the automatic and remote restart feature. For information on how to configure the WebSphere MQ-triggered OAD feature, see the *WebSphere Business Integration Server Express Installation Guide for Windows, for Linux or for i5/OS*.

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

The default value is false.

OADMaxNumRetry

The OADMaxNumRetry property specifies the maximum number of times that the WebSphere MQ-triggered Object Activation Daemon (OAD) automatically attempts to restart the connector after an abnormal shutdown. The OADAutoRestartAgent property must be set to true for this property to take effect.

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

The default value is 1000.

OADRetryTimeInterval

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

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

The default value is 10.

PollEndTime

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

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

If the adapter runtime detects:

- PollStartTime set and PollEndTime not set, or

- PollEndTime set and PollStartTime not set

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

PollFrequency

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

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

The following values are valid for this property:

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

The default is 10000.

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

PollQuantity

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

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

An e-mail message is also considered an event. The connector actions are as follows when it is polled for e-mail.

- When it is polled once, the connector detects the body of the message, which it reads as an attachment. Since no data handler was specified for this mime type, it will then ignore the message.
- The connector processes the first BO attachment. The data handler is available for this MIME type, so it sends the business object to Visual Test Connector.
- When it is polled for the second time, the connector processes the second BO attachment. The data handler is available for this MIME type, so it sends the business object to Visual Test Connector.
- Once it is accepted, the third BO attachment should be transmitted.

PollStartTime

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

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

If the adapter runtime detects:

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

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

RepositoryDirectory

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

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

When the integration broker is a WebSphere message broker or WAS, this value is set to `<ProductDir>\repository` by default. However, it may be set to any valid directory name.

RequestQueue

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

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

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

ResponseQueue

The ResponseQueue property specifies the JMS response queue, which delivers a response message from the connector framework to the integration broker. When the integration broker is InterChange Server Express (ICS), the server sends the request and waits for a response message in the JMS response queue.

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

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

RestartRetryCount

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

The default value is 3.

RestartRetryInterval

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

Possible values for the property range from 1 through 2147483647.

The default value is 1.

RHF2MessageDomain

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

This is an example header:

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

This property is not valid if the value of BrokerType is ICS. Also, it is valid only if the value of the DeliveryTransport property is JMS, and the value of the WireFormat property is CwXML.

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

SourceQueue

The SourceQueue property designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see “ContainerManagedEvents” on page 57.

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

The default value is <CONNECTORNAME>/SOURCEQUEUE.

SynchronousRequestQueue

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

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

The default value is <CONNECTORNAME>/SYNCHRONOUSREQUESTQUEUE

SynchronousRequestTimeout

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

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

The default value is 0.

SynchronousResponseQueue

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

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

The default is <CONNECTORNAME>/SYNCHRONOUSRESPONSEQUEUE

TivoliMonitorTransactionPerformance

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

The default value is false.

WireFormat

The WireFormat property specifies the message format on the transport:

- If the value of the RepositoryDirectory property is a local directory, the value is CwXML.
- If the value of the RepositoryDirectory property is a remote directory, the value is CwB0.

Index

A

- Application-specific configuration
 - properties
 - setting 18
- Attribute-level ASI
 - specifying 43

B

- Business Object Designer
 - running 37
- Business object-level ASI
 - specifying 44
- Business objects
 - creating and modifying 37
 - data queue structure 32
 - generating 42
 - generating definitions 37
 - processing 4
 - specifying attribute properties 34
 - specifying attribute-level application text 35
 - specifying information 43
 - specifying supported definitions 18
 - structure overview 27
 - Understanding 27
 - uploading 45
- Business Objects
 - RPG programs structure 28

C

- Configuration
 - completing 21
- Configuration file
 - completing 15
 - Creating a new 13
 - creating from a connector-specific template 13
 - saving 21
 - setting properties 15
 - using an existing file 14
- Connector Configurator
 - overview 9
 - running from System Manager 10
 - running in stand-alone mode 10
 - starting 10
 - using in globalized environments 22
- Connector instances
 - creating multiple 25
- Connector-specific properties 17
- Connector-specific property template
 - creating 11

D

- Data conversion
 - iSeries or AS/400 toolbox 35
- Data handlers 21

- Data queues
 - overview 3

E

- Error handling 47

H

- How the adapter works 4

I

- Installed file structure 6
- iSeries adapter
 - broker compatibility 5
 - configuring 9
 - environment 5
 - installing 5
 - installing and related files 6
 - operations 4
 - overview 1
 - platforms 5
 - prerequisites 5
 - starting the connector 22
 - stopping the connector 24
 - troubleshooting 47
- iSeries and AS/400 systems
 - overview 1

L

- Logging 47

M

- Maps 19
- Meta objects
 - configuring for polling 33
- Metadata
 - defining 27

O

- Object Discovery Agent (ODA)
 - overview 37

P

- Post installation
 - tasks 7

S

- Standard configuration properties for connectors 49

- Standard connector properties
 - setting 16

T

- Trace/log file values
 - setting 20
- Tracing messages 47

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other 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 grant 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.*

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan*

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 the 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 Corporation
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 measurements 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 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. All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples 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. COPYRIGHT LICENSE: This information contains sample application programs in source language, which illustrate 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. If you are viewing this information softcopy, the photographs and color illustrations may not appear.

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:

i5/OS
IBM
the IBM logo
AIX
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
IMS
Informix
iSeries
Lotus
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both. Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Linux is a trademark of Linus Torvalds in the United States, other countries, or both. Other company, product or service names may be trademarks or service marks of others.

WebSphere Business Integration Server Express and Express Plus include software developed by the Eclipse Project (<http://www.eclipse.org/>).



WebSphere Business Integration Server Express, Version 4.4, and WebSphere Business Integration Server Express Plus, Version 4.4



Printed in USA