WebSphere Business Integration
Server Express and Express Plus

Adapter for XML User Guide

Adapter Version 3.5.x

WebSphere Business Integration
Server Express and Express Plus

Adapter for XML User Guide

Adapter Version 3.5.x

Note!

Before using this information and the product it supports, read the information in "Notices" on page 81.

22April2005

This edition of this document applies to the WebSphere Business Integration Server Express Adapter for XML (5724-H07), version 3.5.x.

To send us your comments about IBM WebSphere Business Integration Server Express documentation, 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 2000, 2004. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2000, 2004

About this document

The products IBM(R) WebSphere(R) Business Integration Server Express and IBM WebSphere 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 the Toolset Express help you to crate, 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.

This document describes installation, configuration, business object development, and troubleshooting for the Adapter for XML.

Except where noted, all the information in this guide applies to both IBM WebSphere Business Integration Server Express and IBM WebSphere Business Integration Server Express Plus. The term ″WebSphere Business Integration Server Express″ and its variants refer to both products.

What this document includes

This document describes installation, connector property configuration, business object development, and troubleshooting for the IBM WebSphere Business Integration Server Express Adapter for XML.

What this document does not include

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

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

Audience

This document is for WebSphere 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
- HTTP and HTTPS based application architecture

Prerequisites for this document

You need to be familiar with the WebSphere Business Integration Server Express Adapters system, business object development, and data handlers. You also need to be familiar with the XML markup language and a schema language, either document type definition (DTD) or XSDL (for schema documents).

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 WebSphere Business Integration Server Express documentation at the following site: http://www.ibm.com/websphere/wbiserverexpress/infocenter.

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, filename, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
italic, *`italic`*	Indicates a variable name or a cross-reference.
blue text	Blue text, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click any blue text to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
`[ ]`	In a syntax line, square brackets surround an optional parameter.
`...`	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.
`< >`	In a naming convention, angle brackets surround individual elements of a name to distinguish them from each other, as in `<server_name><`*`connector_name`*`>tmp.log`.
`/, \`	In this document, backslashes (\) are used as the convention for directory paths for Windows. For Linux, i5/OS installations, substitute slashes (/) for backslashes. All WebSphere Business Integration Server Express system product pathnames are relative to the directory where the product is installed on your system.
`%text%` and *`$text`*	Text within percent (%) signs indicates the value of the Windows `text` system variable or user variable. The equivalent notation in a Linux environment is *`$text`*, indicating the value of the *`text`* Linux environment variable.
ProductDir	Represents the directory where the product is installed. The defaults for each platform are as follows: Windows: IBM\WebSphereServer i5/OS: /QIBM/ProdData/WBIServer44/product Linux: /home/${username}/IBM/WebSphereServer

New in this release

Version 3.5.x

The Adapter for XML has been updated with the following items:

- The adapter, running on Windows, provides bidirectional script support
- Support for GB18030 Chinese code has been added for this release

The 3.5.x version of the adapter is supported on the following platforms:

- WIN 2003
- IBM i5/OS V5R3 and OS/400 V5R2

  Note: Unless explicitly stated, i5/OS refers to OS/400 and i5/OS.
- Linux:
  RedHat 2.1 Enterprise Linux WS/AS/ES 3.0 Update 2, Intel (IA32)
  SuSE Linux ES 8.1, Intel (IA32)
  SuSE Linux ES 9.0, Intel (IA32)

Chapter 1. Overview of the XML adapter

This chapter describes the connector component of the WebSphere Business Integration Server Express Adapter for XML. The connector enables an integration broker to exchange business objects with URLs by using HTTP and HTTPS protocols. A URL can be any destination such as a remote application or a servlet on a Web server. The connector supports XML version 1.0.

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

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

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

For more information about the relationship of the integration broker to the connector, see the *System Administration Guide*.

Note: When working in an XML environment, you can use the product-delivered connector or you can build custom modules. See "Planning a connector implementation" on page 21 for guidelines on deciding which to use.

This chapter contains the following sections:

- "Connector components"
- "How the connector works" on page 4

Connector components

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

- Connector
- XML data handler
- Protocol handler (HTTP and HTTPS)

The connector interacts with the XML data handler. For more on data handlers, see the *Data Handler Guide*.

Figure 1 illustrates the architecture of the connector components. The modular design of the connector enables you to design custom components to replace the product-delivered functionality.

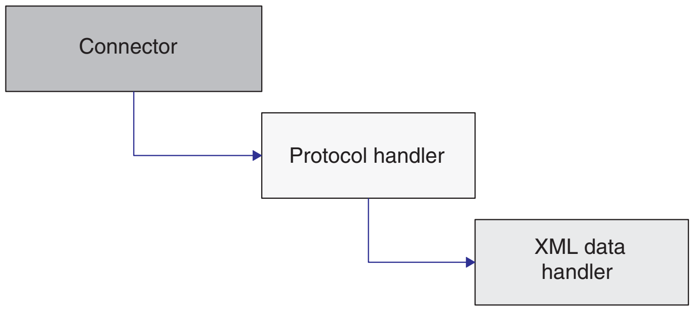


Figure 1. Connector architecture

Connector

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

- Receives business object requests from the integration broker.
- Calls the Protocol Handler Framework and passes the URL string of a business object to invoke the appropriate instance of the protocol handler.
- Passes business object requests to protocol handlers.
- Receives business object responses or success/failure return code from protocol handlers. If the connector is using a synchronous protocol handler, then it receives business object responses. If the connector is using an asynchronous protocol handler, then it reports success or failure based on return code.

The primary methods used by the connector are `init()`, `doVerbFor()`, and `pollForEvents()`. The `init()` method reads all configuration values from the integration broker's repository, sets proxy names (HTTP and HTTPS) and respective ports, and reads the Java class package names for the protocol handler (`JavaProtocolHandlerPkgs`) and XML data handler (`JavaDataHandlerPkgs`), and the value of properties of the data handler and protocol handler.

The `doVerbFor()` method handles the business object request/response operations. When the connector receives a top-level business object from an integration broker, the `doVerbFor()` method extracts the request business object and the destination URL. The `doVerbFor()` method then creates the appropriate protocol handler instance.

When the connector receives a response from the destination URL, the `doVerbFor()` method populates the response business object as a child of the top-level business object and then returns the result to the integration broker. In the connector, all errors are propagated as exceptions and are handled through the connector, where `BON_FAIL` is returned and Return Status Descriptor is set.

The `pollForEvents()` method is used for event notification. The connector has the ability to check for events from a URL by using business objects. For more information on event notification, see "Event notification" on page 6.

The connector sets a static property to contain the name of the top-level data handler meta-object as it is specified in the DataHandlerConfigMO connector configuration property.

Protocol handler (HTTP and HTTPS)

The protocol handler enables the connector to communicate with a URL by using the HTTP and HTTPS protocols. The protocol handler is an abstract base class that extends the Java `URLConnection` class. This class contains abstract methods that can be implemented to provide support for specific protocols, such as HTTP and HTTPS. An instance of the protocol handler is created by the Protocol Handler Framework, which is called by the connector.

The WebSphere Business Integration Server Express Adapter for XML includes asynchronous and synchronous protocol handlers. Synchronous protocol handlers return business objects from synchronous responses. Asynchronous protocol handlers do not expect a response business object; they return only a success or failure message based on the return code from the post operation. The asynchronous protocol handler does not support event notification.

Note: By using the Protocol Handler Framework, you can add support for other protocols such as FTP. The Protocol Handler Framework is an abstract base class called `CWURLConnection`.

The Protocol Handler Framework creates an instance of a protocol handler, and the connector passes a business object to the created instance. The protocol handler extracts the content type (such as `text/plain` or `text/xml`) from the business object and uses it to create an instance of the XML data handler.

When the protocol handler calls the `createHandler()` method, it passes in a content type. The data handler create method passes the content type by replacing the forward slash (/) characters with periods (.) and replacing all non-alphanumeric characters with an underscore (_). Then the create method looks for an attribute in the data handler top-level meta-object that matches the parsed string for the content type. If it does not t find a match, the method builds the class name as `com.crossworlds.DataHandlers.`*`modified content_type`*. [1]

The protocol handler performs the following operations:

- Receives a business object from the connector and passes it to the XML data handler. The protocol handler parses the MimeType attribute to determine which data handler instance to create.
- Receives an XML stream from the XML data handler and then passes it to the appropriate URL. The XML stream represents the request business object.

  If the data handler parses an XML string, the protocol handler converts the XML string into an XML stream before passing it to a URL.
- If it is synchronous, it then receives a response stream from a URL and passes it back to an XML data handler, which converts it back into a business object.
- If it is asynchronous, then it reports success or failure based on return code from the request operation to the URL.
- Sends the response business object back to the connector.

If your implementation of the connector needs to support additional protocols, you must build a custom protocol handler. For information on how to create a custom protocol handler, see Chapter 4, "Building a custom protocol handler," on page 27

1.

How the connector works

The following sections describe how the connector processes business objects, how meta-objects are used for configuration, and how the connector handles event notification.

Business object processing

The connector uses request/response operations to pass data between it and a URL. The connector receives business object requests from an integration broker and converts the requests into XML streams. The request stream is passed to a URL by using the POST method, and a response stream is returned that may or may not have similar content. The response stream is converted into a response business object and returned with the original top-level business object to the integration broker. Note that the type of business object request can differ from that of the business object response.

The complete request-response cycle is illustrated in Figure 2.

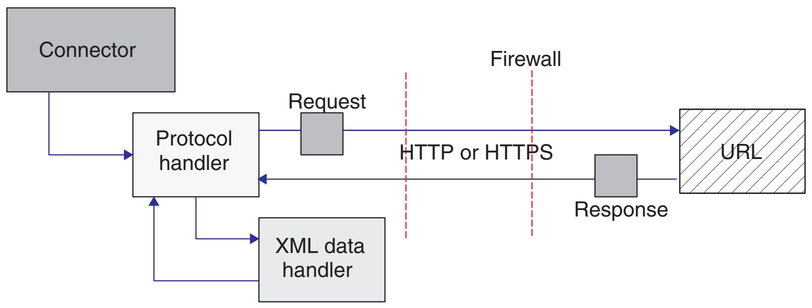


Figure 2. Business object event processing

Request

When the connector receives a business object request from an integration broker, it must convert it into a request stream that can be passed by using the appropriate protocol. The protocol handler and the XML data handler are used to convert and send a request business object to a URL. Figure 3 illustrates the request process.

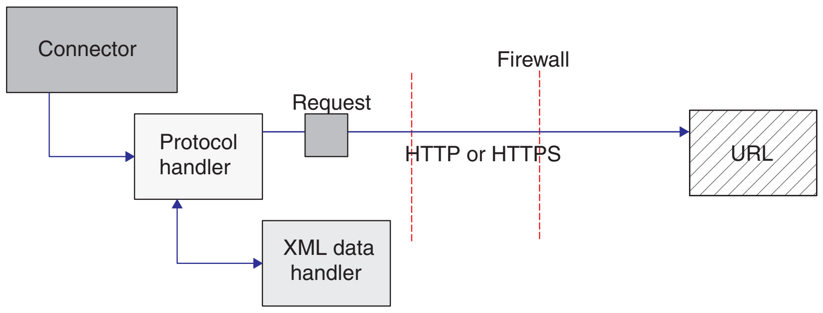


Figure 3. Request processing

Specifically, when the connector receives a top-level business object from the integration broker, the following process flow occurs:

1. The connector calls `getAttrValue` (“`URL`”) and retrieves the URL. It also calls `getAttrValue` (“`MimeType`”) and `getAttrValue` (“`BOPrefix`”) to retrieve the MimeType and BOPrefix attributes values from the business object.

2. The connector extracts the request business object from the top-level business object.
3. The connector calls the appropriate protocol handler (HTTP or HTTPS) based on the protocol specified in the URL field of the top-level business object, and the protocol handler package name specified.
4. The protocol handler calls the appropriate data handler based on the MimeType and BOPrefix attributes of the top-level business object (as configured in the top-level meta-object).
5. The data handler converts the business object into a request stream and then passes it back to the protocol handler.
6. The protocol handler sends the request stream to the destination URL specified in the top-level business object or passes the return code.

Response

If you are using a synchronous protocol handler, then when the response business object is returned from a URL, it is returned in the form of a response stream. If you are using an asynchronous protocol handler, then return code is simply passed back. The response processing is similar to the request processing, except the response stream must be translated back into a business object.

Note: The response stream may not always be represented by the same business object type as the request stream.

Figure 4 illustrates the process flow of the response business object returning to the connector.

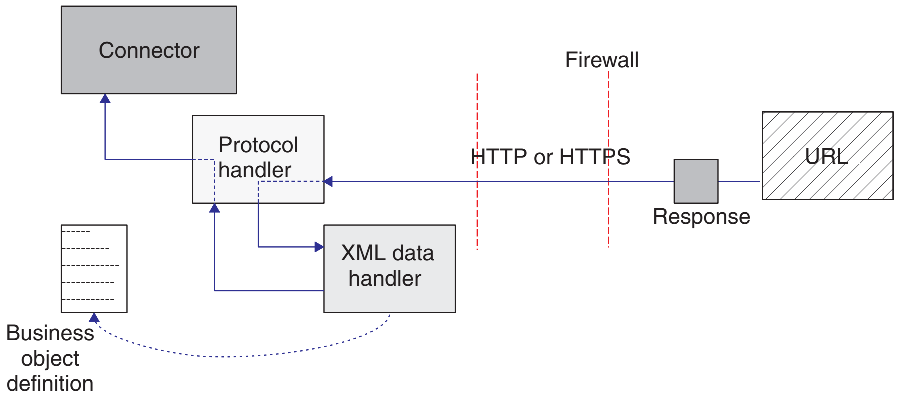


Figure 4. Process flow of data returning from a URL

Specifically, when the protocol handler receives a response stream from a URL, the following process flow occurs if the MIME type is `text/xml`:

1. The protocol handler calls the `getContentType ()`method to retrieve the MIME type value to determine which data handler to use.
2. The protocol handler calls the `DataHandler` class to create an instance of the XML data handler.

   Note that the data format in the response stream can be different from the data format in the original request business object.
3. The protocol handler converts the response stream into a string, which the protocol handler passes to the XML data handler.

4. The XML data handler obtains the business object name based on the message content and extracts the data from the request stream (XML document) into a business object.
5. The XML data handler passes the completed request business object to the protocol handler.
6. The protocol handler passes the request business object to the connector, which adds it to the original top-level business object.
7. The connector passes the original top-level business object containing the request business object back to the integration broker.

Event notification

For event notification, the connector uses business objects to retrieve events from a URL. The connector polls a URL by sending a request XML document that is returned as a response XML document. The response contains child business objects that the connector passes to the integration broker as events. Each child business object is processed as a single event. The asynchronous protocol handler does not support event notification.

Note: Poll for events processing is the same as business object request processing except that there is an additional step to extract event objects from the response business object and send them to the integration broker.

An event notification business object follows the same business object processing operations as the request and response business objects of an XML business object. All unsubscribed events are archived to a file in the WebSphere Business Integration Server Express Adapters standard business object dump format.

To enable event notification, you need to define event notification business objects and set up your URL (such as a Web servlet or `cgi-bin` script) to handle these business objects. The connector uses the POST method to send an XML event request document as a stream to the URL. The URL should read the XML document as a stream from STDIN and write an XML document which contains one or more event objects as a stream to STDOUT.

Figure 5 illustrates the basic process of event notification.

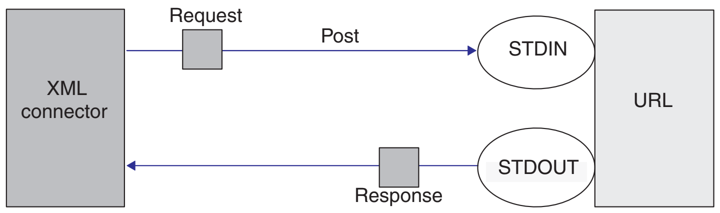


Figure 5. Event notification process

For more information on defining business objects, see Chapter 3, “Developing business objects for the connector,” on page 21

Chapter 2. Installing and configuring the connector

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

Adapter environment

Adapter platforms

The adapter is supported on the following platforms:

- **Linux**:
  RedHat 2.1 Enterprise Linux WS/AS/ES 3.0 Update 2, Intel (IA32)
  SuSE Linux ES 8.1, Intel (IA32)
  SuSE Linux ES 9.0, Intel (IA32)
- **i5/OS V5R3 and OS/400 V5R2**
- **Windows:**
  Windows XP with Service Pack 1A, for WebSphere Business Integration Server Express Adapter Framework (administrative tools only)
  Windows 2003 (Standard Edition or Enterprise Edition)

Adapter prerequisites

To use the connector, your environment must have:

- Java Secure Socket Extension 1.0.3 (JSSE)

  The WebSphere Business Integration Server Express Adapter delivers the international version. You might want to download one with domestic grade encryption and add it to the `\connector\Xml\dependencies` directory.
- Access to the destination URLs
- XML Data Handler

Locale-dependent data

The connector has been internationalized so that it can support double-byte character sets, and deliver message text in the specified language. When the connector transfers data from a location that uses one character code to a location that uses a different code set, it performs character conversion to preserve the meaning of the data. The Java runtime environment within the Java Virtual Machine (JVM) represents data in the Unicode character code set. Unicode contains encodings for characters in most known character code sets (both single-byte and multibyte). Most components in the WebSphere business integration system are written in Java. Therefore, when data is transferred between most integration components, there is no need for character conversion. To log error and informational messages in the appropriate language and for the appropriate country or territory, configure the `Locale` standard configuration property for your environment. For more information on configuration properties, See Appendix A, "Standard configuration properties for connectors," on page 33.

This adapter supports the processing of bidirectional script data for languages such as Arabic, Hebrew, Urdu, Farsi and Yiddish. To use the bidirectional capacity, you must configure the bidirectional standard properties. For more information, refer to the standard configuration properties for connectors inAppendix A, "Standard configuration properties for connectors," on page 33.

Installing the XML adapter

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.

Verifying installation

The sections below describe the paths and filenames of the product after installation and how to verify your adapter installation.

Note: All product pathnames are relative to the directory where the product is installed on your system.

Verifying installation on a Windows system

Before you begin, install the adapter. The Installer copies the standard files associated with the adapter into your system. The utility installs the connector into the *ProductDir*`\connectors\XML` directory, and adds a shortcut for the connector to the Start menu.

Perform the following steps to verify adapter installation on a Windows system, change to the directory where you installed the adapter `ProductDir\` and compare the contents to those listed in Table 1.

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

Table 1. Windows file structure for the connector

Subdirectory of *ProductDir*	**Description**
`\connectors\XML`	Contains the adapter application-specific component class files CWXML.jar, CwProtocolHandler.jar, and the adapter start file start_XML.bat
`\connectors\messages\XMLConnector.txt`	Contains the error codes and description of the errors used by the connector
`\connectors\XML\Samples`	Contains the files used for testing the connector

Table 1. Windows file structure for the connector (continued)

Subdirectory of *ProductDir*	**Description**
`\connectors\XML\Dependencies\IBM`	Contains the .jar files for implementing security
\connectors\XML\start_XML_service.bat	The startup script for the connector service.

Verifying installation on a Linux system

Before you begin, install the adapter. The Installer copies the standard files associated with the adapter into your system. The utility installs the connector agent into the `*ProductDir*/connectors/XML` directory.

Perform the following steps to verify adapter installation on a Linux system, change to the directory where you installed the adapter `ProductDir/` and compare the contents to those listed in Table 2.

Table 2 describes the Linux file structure used by the adapter, and shows the files that are automatically installed when you choose to install the adapter through Installer.

Table 2. Linux file structure for the connector

Subdirectory of *ProductDir*	**Description**
/QIBM/UserData/WBIServer44	Contains the adapter application-specific component class files CWXML.jar, CwProtocolHandler.jar, and the adapter start file start_XML.sh
`/connectors/messages/XMLConnector.txt`	Contains the error codes and description of the errors used by the connector
`/connectors/XML/Samples`	Contains the files used for testing the connector
`/connectors/XML/Dependencies/IBM`	Contains the .jar files for implementing security

Verifying the installation on an i5/OS system

Before you begin, install the adapter. The Installer copies the standard files associated with the adapter into your system. The utility installs the connector agent into the `*ProductDir*/connectors/WBIMB` directory.

Perform the following steps to verify adapter installation on an i5/OS system, change to the directory where you installed the adapter `ProductDir/` and compare the contents to those listed in Table 3.

Table 3 describes the i5/OS file structure used by the adapter, and shows the files that are automatically installed when you choose to install the adapter through Installer.

Table 3. i5/OS file structure for the connector

Subdirectory of *ProductDir*	**Description**
/QIBM/UserData/WBIServer44	Contains the connector CWXML.jar, CWProtocolHandler.jar and start_XML.sh files

Table 3. i5/OS file structure for the connector (continued)

Subdirectory of *ProductDir*	Description
/QIBM/UserData/WBIServer44	Location for jcert.jar, jnet.jar, and jsse.jar files that are required for https connections. To use the Adapter for XML with your i5/OS system, you must obtain appropriate jsse.jar, jnet.jar, and jcert.jar files for use with i5/OS, and add them to this directory. These files are available as downloads from the Sun web site at: http://java.sun.com/products/jsse/. To set up these files, see the steps following this table.
/connectors/XML/Samples	Contains java and xml files and sample business objects used for testing.
/connectors/XML/Dependencies/Sun	Contains the xml.jar file.
/connectors/messages	Contains the XMLConnector.txt file.
repository/XML	Contains the CN_XML.txt file.
/lib	Contains the WBIA.jar file.
/bin	Contains the CWConnEnv.sh file.

To use the XML adapter with encryption on i5/OS, you must manually install some jar files that are not shipped with this product.

Perform the following steps to install the jar files in /QIBM/UserData/WBIServer44 to use the adapter with i5/OS.

1. Obtain the appropriate jsse.jar, jnet.jar, and jcert.jar files for use with IBM i5/OS and from the sun web site at: http://java.sun.com/products/jsse/
2. Unzip the jsse jar to hard-disk which creates a jsse project directory. Under the jsse directory there is a /lib directory (for example: \jsse1.0.3_03\lib) that contains the necessary jsse jars to run the adapter.
3. Copy the three jar files in the /lib directory to the i5/OS IFS in the following location: /QIBM/UserData/WBIServer44 (you can use Windows Explorer and copy to the IFS mapped drive).
4. After the jars are copied to IFS, the owner of the jar files should be changed to QWBISVR44 with the following CL commands:
   CHGOWN OBJ ('/QIBM/UserData/WBIServer44/jnet.jar')
   NEWOWN (QWBISVR44) RVKOLDAUT (*YES)
   CHGOWN OBJ ('/QIBM/UserData/WBIServer44/jcert.jar')
   NEWOWN (QWIBSVR44 RVKOLDAUT (*YES)
   CHGOWN OBJ ('/QIBM/UserData/WBIServer44/jsse.jar')
   NEWOWN (QWBISVR44) RVKOLDAUT (*YES)
5. The start_XML.sh script must be edited to include the jar files in the classpath. Edit the following file:
   /QIBM/UserData/WBIServer44/<ICSName>/connectors/
   <XMLBasedConnector> is an XML connector
   for example: "XMLConnector' (for the default), or 'MyXMLConnector' if the user created a custom XML adapter named MyXML Connector)
   export JAVA_JSSE=/QIBM/UserData/WBIServer44/jcert.jar:/QIBM/UserData/WBIServer44/jnet.jar:/QIBM/UserData//WBIServer44/jnet.jar:/QIBM/UserData/WBIServer44/jsse.jar export JCLASSES=${JCLASSES}:${JAVA_JSSE}.

Configuring the connector

To use InterChange Server Express as the integration broker, configure connector properties in Connector Configurator Express, which you access from the System Manager.

Configuring the data handler

Configure the meta-objects used for the XML data handler. Note that `repository\DataHandlers\MO_DataHandler_DefaultXMLConfig.xsd` is provided as the default meta-object. For information on configuring meta objects, see "Configuring top-level meta-objects for the data handler" on page 15.

Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, "Standard configuration properties for connectors," on page 33 for documentation of these properties.

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

Table 4 provides information specific to this connector about a configuration property in the appendix.

Table 4. Property information specific to this connector

Property	Note
`CharacterEncoding`	This connector does not use this property.
`Locale`	Because this connector has been internationalized, you can change the value of this property.

You must provide a value for the `ApplicationName` configuration property before running the connector. You also must set at least the following standard connector configuration properties before running the connector.

- AgentTraceLevel
- ApplicationName
- ControllerStoreAndForwardMode
- ControllerTraceLevel
- DeliveryTransport

Connector-specific properties

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

Table 5 lists the connector-specific configuration properties for the connector. See the section that follows for explanations of the properties.

Table 5. Connector-specific configuration properties

Name	Possible values	Default value	Required?
ArchiveDirectory	*archive directory name*	`\connectors\xml\ archive`	
DataHandlerConfigMO	*data handler meta-object name*	MO_DataHandler_ Default	YES
HttpProxyHost	*http host name*		
HttpProxyPort	*http proxy port*	80	
HttpTimeout	Time that the connector waits before timing out an http session.	0	
HttpsDebug	Set the property to one of the values in Table 6 on page 13.		
HttpsProxyHost	https host name		
HttpsProxyPort	https proxy port	443	
JavaProtocolHandlerPkgs	protocol handler name	com.crossworlds. connectors.utils. ProtocolHandlers	YES
Login	user login		
MaxNumRetries	positive integer	10	
Password	user password		
PollingBusinessObjects	business object names		
ReturnBusObjResponse	true or false	true	
ResponseTLO	true or false	false	
SecurityProvider	SSL implementation	com.sunnet.sslinternalssl. Provider,sun.security.provider.Sun .jsse.JSSEProvider (valid?) (To use the Adapter for XML with your i5/OS system, you must obtain the appropriate jsse.jar, jnet.jar, and jcert.jar files for used with IBM i5/OS and add them to the /QIBM/UserData/WBIServer44 directory. these files are available as downloads from the Sun web site at: http://java.sun.com/products/jsse.)	YES
UseCaches	true or false	false	
UseDefaults	true or false	false	
UseDigitalSignature	true or false	false	

ArchiveDirectory

Directory containing archived events. Each event can be identified by its business object name and verb. By default, the create verb is appended to the business object name. The default is `\connectors\xml\ archive`. This property is BiDi supported.

DataHandlerConfigMO

Name of the top-level meta-object that the XML connector uses to determine its data handler support. This meta-object must contain the name of the child meta-object that the XML data handler uses to set configuration properties. This property is also used by the DataHandler base class to determine which DataHandler class to instantiate for a particular content type. The default is MO_DataHandler_ Default. For more information, see "Configuring top-level meta-objects for the data handler" on page 15.

HttpProxyHost

Name of the host that acts as the proxy for HTTP. This property is required only if the network uses a proxy server that uses the HTTP protocol.

HttpProxyPort

Proxy port number used to connect HTTP. This property is required only if the network uses a proxy server that uses the HTTP protocol. The default port number is `80`.

HttpTimeout

The amount of time, specified in milliseconds, that the connector waits for a response from the server before timing out the HTTP session.

The default value is set to `0.`

HttpsDebug

Setting that determines what debugging information is generated for the HTTPS session. Table 6 lists the HTTPS debug values for the HTTPS protocol handler.

Table 6. HttpsDebug Values

Name	Meaning
`all`	Turn on all debugging
`data`	Hex dump of each handshake message. It can be used to widen handshake debugging.
`handshake`	Print each handshake message. It can be used with SSL.
`keygen`	Print key generation data. It can be used with SSL.
`plaintext`	Hex dump of record plain text. It can be used to extend record debugging.
`record`	Enable per-record tracing. It can be used with SSL.
`session`	Print session activity. It can be used with SSL.
`ssl`	Turn on SSL debugging only.
`verbose`	Print verbose handshake message. It can be used to extend record debugging.

HttpsProxyHost

HTTPS proxy machine name. This property is required only if the network uses a proxy server that uses the HTTPS protocol.

HttpsProxyPort

Proxy port number used to connect HTTPS. This property is required only if the network uses a proxy server that uses the HTTPS protocol.

JavaProtocolHandlerPkgs

If this attribute is present, it specifies the packages that are used as protocol handlers, instead of the default Java handler. These classes must conform to Java's Protocol Handler Framework. For example, to use a protocol handler named `com.mycompany.http` (for HTTP), set this field to `com.mycompany`. Also make sure that the `.jar` files of the corresponding class are in your classpath.

For more information on Java Protocol Handlers, see the tutorial at the following website:
`http://developer.java.sun.com/developer/onlineTraining/protocolhandlers/`

It is also possible to specify multiple packages for this value separated by vertical bars, "|". For example, `com.crossworlds.Protocol Handlers|com.mycompany`

WebSphere Business Integration Server Express Adapter delivers two packages:

- `com.crossworlds.connectors.utils.ProtocolHandlers` (synchronous protocol handler)
- `com.crossworlds.connectors.utils.ProtocolHandlers.async` (asynchronous protocol handler)

The default is `com.crossworlds.connectors.utils.ProtocolHandlers`.

Login

The user's login identity, which is passed as an authentication credential in support of base-64 authentication.

MaxNumRetries

Specifies the number of retries to attempt after the asynchronous protocol handler does not receive a response from the destination URL. This property internally uses the Httptimeout for each retries. For each retries the adapter will wait for the amount of time specified in the HttpTimeout. This property is used only by the asynchronous protocol handler. If you do not specify a value, this property defaults to`0`.

Password

The user's password, which is passed as an authentication credential in support of base-64 authentication.

PollingBusinessObjects

Business objects used for event notification. Separate multiple entries with a comma (For example, XMLPoll_Cust, XMLPoll_Order). Each business object must be supported by the connector. This property is required if the connector is set up for event notification.

ReturnBusObjResponse

Determines whether the connector expects a business object to be returned from the protocol handler.If the value is set to true, then the connector expects a business object. If the value is set to false, then the connector does not expect a business object. It expects only a response of success or failure. The default is true.

Note: If you are configuring the asynchronous protocol handler, then the value must be set to false, because the asynchronous protocol handler does not expect a business object response.

ResponseTLO

Used during polling, and determines whether the adapter picks up a container object that subsumes other objects, or each individual object. When it is set to to `true`, the adapter passes the container object. When it is set to `false`, the adapter passes the individual objects.

The default setting is false.

SecurityProvider

Used by HTTPS during the SSL handshake. Comma-separated values of this attribute determine which SSL implementation to use when connecting to HTTPS URLs. If no value is set, HTTPS connections do not work. The default value is `com.sun.net.ssl.internal.ssl.Provider, sun.security.provider.Sun.`

UseCaches

If this attribute is set to `false`, the connector attempts to retrieve a non-cached version of XML documents. This is merely a request; it cannot be strictly enforced by the connector. Set this value to `true` to retrieve cached XML documents only.

UseDefaults

On a Create operation, if UseDefaults is set to `true`, the connector checks whether a valid value or a default value is provided for each isRequired business object attribute. If a value is provided, the Create operation succeeds. If the parameter is set to `false`, the connector checks only for a valid value and causes the Create operation to fail if it is not provided. The default is `false`.

UseDigitalSignature

Specifies whether a digital signature length (0 in binary) is added to the end of a message sent by using the HTTP or HTTPS protocol. If your connector supports digital signatures, set this property to `true`. The default is `false`.

Note: The product-provided connector does not support digital signatures.

Configuring top-level meta-objects for the data handler

Meta-objects are business objects that contain configuration information. A data handler meta-object contains information to configure a data handler. The connector uses the information in the data handler meta-object to create an instance of the XML data handler.

Before running the connector, you must set up a data handler meta-object that specifies what data handler the connector will use based on the MIME type. When the connector receives a business object request, it uses information from the meta-object to dynamically create an instance of the appropriate data handler.

The connector gets the name of the top-level data handler meta-object from the connector configuration property DataHandlerConfigMO. The top-level meta-object is a hierarchical business object that can contain any number of child objects. Each child object is a flat object that represents a specific data handler instance. Child meta-objects have attributes that provide configuration values that enable a data handler instance to do its work. Different types of data handlers require different configuration properties, so the child meta-objects that support specific handlers have different attributes.

The XML connector uses the XML data handler to convert between business objects and XML documents. To configure the XML data handler for the connector, do the following:

- Set up the top-level data handler meta-object to have an attribute for each MIME type that the connector supports. The attribute name should be the name of the MIME type. The attribute represents a child meta-object for a data handler instance.

  For the XML connector, make sure that top-level meta-object contains an attribute for the `text/xml` MIME type. This attribute must also list the name of the child meta-object for the XML data handler.
- Set the default attribute values in each child meta-object. The configuration properties for the WebSphere Business Integration Adapter Server Express data handlers are described in the *Data Handler Guide*.

  In the child meta-object for the XML data handler, set the appropriate default attribute values.

For detailed information on setting up meta-objects for individual data handlers, see the *Data Handler Guide*.

Note: For the connector to instantiate a data handler, the data handler top-level meta-object must be part of the connector's supported objects list.

Common configuration tasks

This section describes common configuration tasks for the connector.

Setting up event notification

To enable event notification for the connector, follow these steps:

1. Create a top-level business object containing a child request and a response business object.
2. Configure your URL to handle the structure of the request and response business objects. For more information on defining business objects, see Chapter 3, "Developing business objects for the connector," on page 21
3. After you have defined your event notification business objects, set the PollingBusinessObjects and ArchiveDirectory configuration properties by using Connector Configurator Express. Select InterChange Server Express as the integration broker, by using Connector Configurator Express, which you access from System Manager.

Specifying a data handler

To specify a data handler to be used by the XML connector, use the following steps.

1. Determine the types of data formats that the connector will support.

   By default, the connector uses the XML data handler for the `text.xml` MIME type. If you are converting between business objects and some other MIME type, make sure that the MIME type is an attribute in the top-level data-handler meta-object (by default, `MO_DataHandler_Default`). Only one data handler can be used to convert any given format type.
2. Determine which data handler (or data handlers) the connector will use.

   The top-level data-handler meta-object associates MIME types with a child data-handler meta-object. The child data-handler meta-object determines which data handler is instantiated.
3. Use the Business Object Designer Express to modify the data-handler meta-objects.

   Note: With InterChange Server Express as the integration broker, you can launch Business Object Designer Express from within System Manager.
4. In Connector Configurator Express or System Manager, add the data handler top-level meta-object to the list of supported objects for the connector. If the connector is not subscribed to the top-level data-handler meta-object, the connector does not load the meta-object at startup.
5. Specify the name of the top-level data-handler meta-object in the connector `DataHandlerConfigMO` configuration property. The product-delivered default is the `MO_DataHandler_Default` meta-object.

For more information about the data-handler meta-objects, see the *Data Handler Guide*.

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_XML.bat connName serverName.
- **For i5/OS Platforms:**

  `/QIBM/UserData/WBIServer44/WebShereICSName/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

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

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

Create a start-up script

To create a startup script:

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

   dirname
2. Put this startup script in the connector directory you created in "Create business object definitions" on page 17.
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(QWBIXML) FROMLIB(QWBISVR44)OBJTYPE(*JOBD)
   TOLIB (QWBISVR44) NEWOBJ(newxmlname) where newxmlname 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.

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\connNamewhere *connName* identifies the connector.

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.

The name of the startup script depends on the operating-system platform, as Table 7 shows.

Table 7. Startup scripts for a connector

Operating system	Startup script
Linux-based systems	`connector_manager`
i5/OS	`start_connName.sh`
Windows	`start_`*`connName`*`.bat`

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

Note: You need a local configuration file if the adapter is using JMS transport.
Starting the connector on a Windows system:

- From the **Start** menu, select **Programs>IBM WebSphere Business Integration Server 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 {-cconfigFile}.
- 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-based systems:

- From the command line type:
  ```
  connector_manager -start connName brokerName [-cconfigFile ]
  ```
  where *connName* is the name of the connector and *brokerName* identifies your 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 OS/400 or 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. This is the only way the adapter will start using the subsystem within the autostart job entry for the server.
- 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/connName/ start_connName.sh connNameWebsphereICSName* [-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 that 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 OS/400 or i5/OS system name or IP address and a user profile and password that has *JOBCTL special authority. Select the XML 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. For example, for the XML connector the jobname is QWBIXMLC. 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.

Chapter 3. Developing business objects for the connector

This chapter describes the structure of a top-level business object passed by the connector. It also describes the required attributes and describes how the connector processes the top-level business object. It contains the following sections:

Planning a connector implementation

The connector is designed in a modular fashion, so that components can be replaced or added without compromising the integrity of the connector. Before you begin configuring the connector and its components, take the time to analyze the system you need to develop.

Use the following information to determine whether you can use the connector components unchanged as they are delivered with the product. If the functionality of a component of the connector does not meet your needs, you can replace it with a custom component. For example, you might need to implement a custom data handler if your application expects data types other than XML.

Use Table 8 to determine whether you can use the connector components as delivered or whether you have to create a custom component.

Table 8. Using a WebSphere Business Integration Adapter Server Express component or building a custom component?

Use WebSphere Business Integration Server Express Adapter-delivered component...	**If all conditions are true...**	**Otherwise, build a...**
Synchronous protocol handler (HTTP/HTTPS)	• Using the HTTP or HTTPS protocol • Can use a user/password exchange • Do not require certification details • Require a response business object from a URL	Custom protocol handler, See *Chapter 4, "Building a custom protocol handler," on page 27*
Asynchronous protocol handler (HTTP/HTTPS)	• Using the HTTP or HTTPS protocol • Can set User/password exchange • Do not require certification details • Require only success or failure return code from a URL (do not require a response business object)	Custom protocol handler See *Chapter 4, "Building a custom protocol handler," on page 27*
XML data handler	Data format is XML 1.0 See *Data Handler Guide*	Custom data handler See *Data Handler Guide*
Name Resolver (XML data handler)	The business object name is determined by using the root element name in the XML document and the `BOPrefix` attribute in a child meta-object for the XML Data Handler (configurable).	Custom name resolver See *Data Handler Guide*

Table 8. Using a WebSphere Business Integration Adapter Server Express component or building a custom component? (continued)

Use WebSphere Business Integration Server Express Adapter-delivered component...	If all conditions are true...	Otherwise, build a...
Entity Resolver (XML data handler)	• Ignore external entities • Search local file system for external entities	Custom entity resolver See *Data Handler Guide*
SAX Parser (XML data handler)	Data format is XML	Custom parser

Connector business object processing

The connector passes business objects between the integration broker and protocol handlers. It sends request business objects to a protocol handler and receives response business objects from protocol handlers. However, it does not process any of the data in the business objects.

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

1. Extracts the request business object from the top-level business object. The connector expects the request business object to be the first child business object that does not have a value of `CxIgnore` or `CxBlank`.
2. Sends the request business object to a protocol handler.
3. When the protocol handler returns the response business object, the connector adds the response business object to the top-level business object and returns the complete top-level business object to the integration broker.

Connector business object structure

The connector requires a hierarchical business object. The top-level business object contains attributes whose values are the destination URL string, the MIME type of the data, the business object prefix, and the request and response business objects.

Figure 6 illustrates the required basic structure of a top-level business object for the WebSphere Business Integration Server Express Adapter for XML.

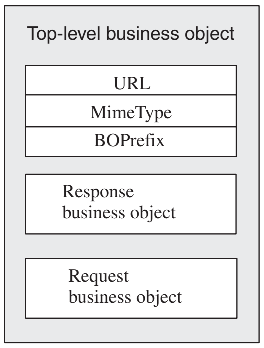


Figure 6. Basic Structure of a top-level business object

For example, if you create two business objects, XMLApp_CustCreateRequest and XMLApp_CustCreateResponse, then a top-level business object definition for the connector might look like:

```
XMLApp_CustCreate

URL      String
MimeType String
BOPrefix String
Response XMLApp_CustCreateResponse
Request  XMLApp_CustCreateRequest
```

Use Business Object Designer Express to create the request and response business objects. Create the top-level business object definition, and add the required attributes and the request and response business objects. Then configure the connector to support the top-level business object.

Required attributes for the top-level business object

The top-level business object must have at least one attribute for the URL string, MIME type, BOPrefix, request business object, and response business object. Each of these attributes must be marked `IsRequired = True`.

Table 9 describes the required attributes of a top-level business object. See the sections that follow for more information.

Table 9. Required attributes for a top-level XML business object

Attribute	Type	Description
URL	String	Destination URL.
MimeType	String	MIME type to be used for the transaction.
BOPrefix	String	Used with the MIME type to create an instance of the XML data handler.
Response	Business object	Business object that represents a response message. See "Request and response business objects" on page 24.
Request	Business object	The business object that represents a request message. In the top-level business object, place this attribute after the attribute for the response business object.

Note: The connector requires that you set at least one attribute as the key attribute. However, the connector does not need any attributes to be set as the key.

URL

The URL string defines the destination of the data in the business object and the protocol to use to pass the data. The string contains the entire destination, including the protocol (such as HTTP or HTTPS); therefore, a separate attribute that specifies the protocol is not required.

The URL string is used by the connector to open a connection to the destination URL. When a connection is opened, the connector uses the URL string to create an instance of the appropriate protocol handler.

For example, the string `http://www.ibm.com` specifies that the HTTP protocol is being used, and an instance of the HTTP Protocol Handler is created.

MIME type

The MIME type defines the content type and format for the data that is being passed to a URL. The connector uses the MIME type to invoke the appropriate data handler. A meta-object identifies the data handler instance for the MIME type/BOPrefix combination. If the data handler implementation handles only one MIME type, the BOPrefix attribute in a child meta-object is optional. In a top-level business object, it is required.

The connector expects the MIME type to be `text/xml` by default, but you can configure the connector to use other MIME types.

BOPrefix

The connector uses the BOPrefix with the MimeType attribute to invoke the appropriate data handler instance. This attribute is required to guarantee the uniqueness of a business object name. For example, you can have two Purchase Order business objects for different applications: AppA_PO and AppB_PO.

Note: The BOPrefix attribute in a top-level business object is different from the BOPrefix attribute in a child meta-object for the XML data handler. For more information on XML data handler child meta-objects, see the *Data Handler Guide.*

When an XML stream returns from a URL, the XML data handler maps the root element name in the XML stream to the business object definition `BOPrefix_name`. The value of the root element name is always placed after the value of BOPrefix.

For example, if you have the root element as <Customer> in the XML document and `BOPrefix`=AppA, then the `BOPrefix_Name` is AppA_Customer.

Request and response business objects

The request and response business objects contain the actual data to be passed to a destination URL. When the connector receives a top-level business object, only the request business object is populated; the response business object is populated with the data returned from the destination URL.

Note these guidelines for defining request and response business objects in the top-level business object:

- Place the response business object before the request business object if the following are true:
  - The request and response business objects are of the same type
  - The business object is to be used for collaboration requests (only when WebSphere npnp is the integration broker)
  - The data in the request business object needs to be preserved (not overwritten by the response from the URL)
- In the top-level business object, set the attribute value for the response business object to `CxIgnore` or `CxBlank`. The connector passes the first non-null attribute value to the protocol handler.
- If the business object that represents the request is identical to the business object that represents the response, the types of the request and response attributes should be the same.
- The request and response business objects can be different. For example, you can send out a customer purchase order business object and receive an order status business object.

- You can define multiple response business objects to support each response XML document that is to be returned to the top-level business object. Multiple response business objects enable the connector to handle the possibility that different types of XML documents (corresponding to different business object types) can be returned from a Web server.

Business object conformance with data handler requirements

Although you can include any WebSphere Business Integration Server Express Adapter business object in the top-level wrapper business object for the connector, the contained business objects must deliver data in a form that is compliant with the requirements of the data handler used to convert the data.

For example, for the BySize data handler, a business object definition must specify a value for the MaxLength attribute property for each business object attribute. For the XML data handler, the business object definition must include application-specific information that enables the data handler to generate an XML document.

Therefore, a good practice is to create your own business object for each type of data to be processed. In the business object, provide only the data required by the application and the information required by the data handler. You can then include these business objects in the top-level connector business object.

See the *Data Handler Guide* for information specific to each data handler.

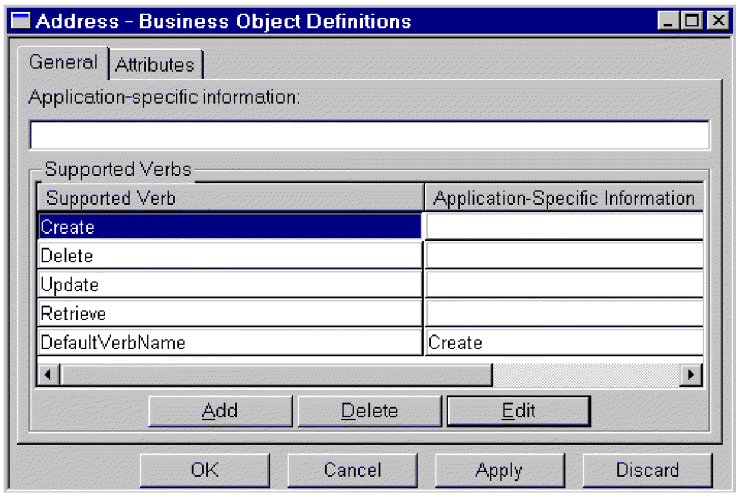
Business objects for event notification

The structure of an event notification business object is similar to the structure of a request business object in that they both require attributes for a URL, MIME type, BOPrefix, response business object, and request business object. The only difference in business object processing is how the connector handles the contents of the response business object. For event notification, the connector expects the response business object to contain child business objects that represent events.

When defining an event notification business object, keep the following in mind:

- The top-level business object needs to have both a request and a response attribute. They both need to be required, and they need to be of different types.
- Place the request business object before the response business object.
- A response business object can return multiple child business objects of the same type. For example, you can design the response business object to return only customer events.
- A response business object can return multiple child business objects of different types. For example, you can design the response business object to return order and customer events.
- All unsubscribed child business objects are archived to the archive directory.
- The business object must have the '`DefaultVerbName`' verb added to the Supported Verb column along with the default verb in the Application-Specific Information column of the business object definition. The default verb is the verb to be used for event notification so that subscriptions are checked properly. You must set the verb for each business object that is to be sent to the integration broker.

Figure 7 illustrates the placement of the 'DefaultVerbName' in the Business Object Definitions.



Application-specific information:

Supported Verbs

Supported Verb	Application-Specific Information
Create	
Delete	
Update	
Retrieve	
DefaultVerbName	Create

Figure 7. Placement of the 'DefaultVerbName' in the business object definitions

Business objects based on XML DTDs or schema documents

If you are creating request and response business objects based on an XML DTD or schema document, you must create a business object definition for each type of XML document to be processed. The business object definition contains structure information that is contained in the XML document's DTD or schema document. For example, if there is one request stream (a single DTD or schema document), but four possible response stream types (four separate DTDs or schema documents), you must define five business object definitions. On the other hand, if the request and response stream use the same schema, you need only one business object definition. You can use the XML Object Discovery Agent (ODA) to generate business object definitions based on DTDs or schema documents.

Note: When reading a DTD or schema, the XML ODA ignores FIXED attributes since the value of these attributes are optional in an xml instance, and the value is always fixed. If you would like these FIXED values to exist in the xml instances created from and read to BOs, you must manually add the FIXED attributes as BO attributes. You must check to ensure that these values are not changed at runtime.

For information about how to define business object definitions for XML documents, either using the XML ODA or manually, see the *Data Handler Guide*.

Chapter 4. Building a custom protocol handler

This chapter describes the Protocol Handler Framework and how to use it to build a custom protocol handler. It contains the following sections:

Protocol handler framework

The WebSphere Business Integration Server Express Adapter Protocol Handler Framework enables developers to write protocol handlers for different types of protocols in a uniform manner. The Protocol Handler Framework has a class called `CWURLConnection`, which contains the abstract methods that need to be implemented to create a custom protocol handler. The framework is part of the `com.crossworlds.protocolhandler` package.

Protocol handler framework classes

Every custom protocol handler must have at least two classes:

- `Handler`
- `cw_`*`protocol`* `connection` (`cw_httpconnection` for HTTP protocol)

The `connection` class extends the `CWURLConnection` class.

Figure 8 illustrates the hierarchy of the `com.crossworlds.connectors.utils.protocolhandler` base class.

© Copyright IBM Corp. 2000, 2004

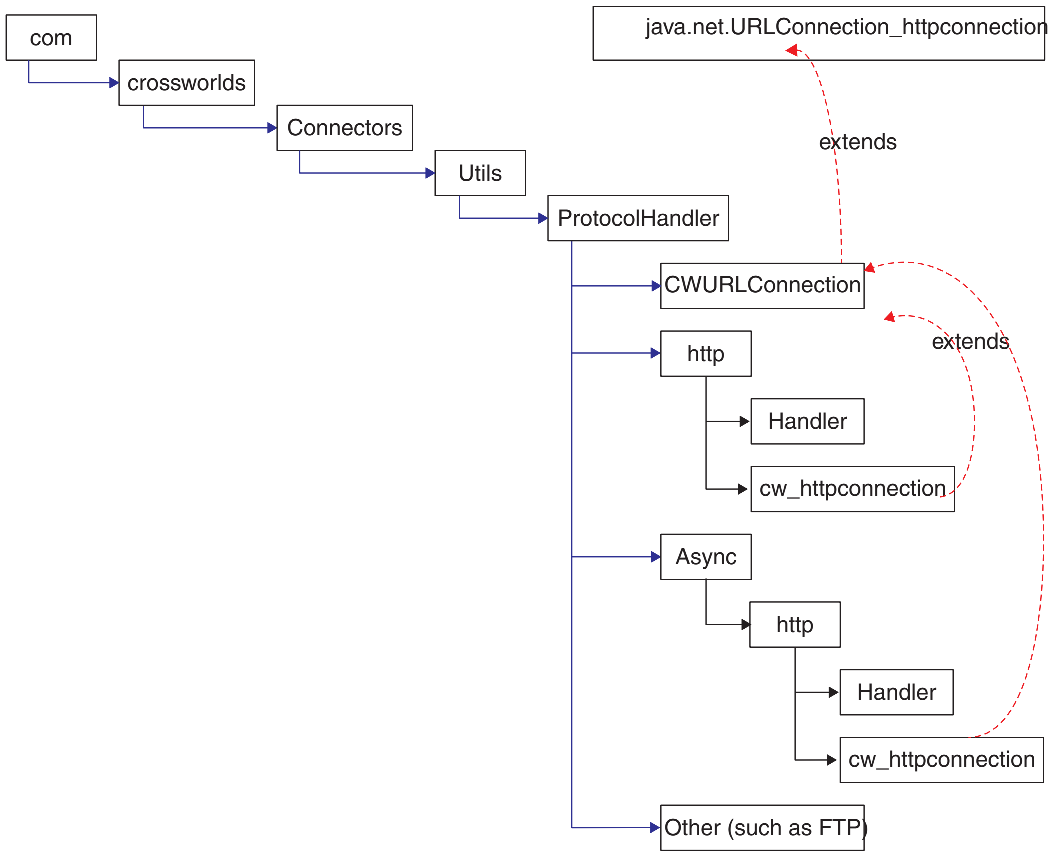


Figure 8. Protocol handler class hierarchy

To use the Protocol Handler Framework to develop a custom protocol handler, do the following:

- Create a ProtocolNameConnection Class, where ProtocolName is the name of your protocol.
- Provide at least one implementation of the `getContent()` method in the `connection` class.
- Create a Handler class.

Handler class summary

```
Public URLConnection openConnection(URL url); throws IOException
```

Connection class summary

```
public String getContent (object input, String mimeType,
Sting BOPrefix, Long ServerTimeout, String Credentials) throws IOException
public String getContentType()
public synchronized void connect() throws IOException
```

Creating a protocol handler class

When you install the connector, stub code and make files for the protocol handler are installed. The stub file contains Java code that defines an empty class that lists all the methods you must implement. You can use the stub file as a template for generating your custom protocol handler.

To implement your new protocol handler:

1. Modify (and rename) the `stubProtocolHandler.java` file.
2. Edit the makefile to contain the name of your source file.
3. Run the `makeProtocolHandler.bat` file to compile the class. The makefile compiles only the class. It does not add the class to the `.jar` file.
4. Add the new class to the `.jar` file. Use the following command:

   ```
   jar cvf MyProtocolHandler.jar <classes>.
   ```

   where:
   - `MyProtocolHandler.jar` is the protocol handler `.jar` file. This file must be in the classpath where the connector startup batch file, `start_xml.bat` (`start_xml.sh` for Linux), resides.
   - `<classes>` are all of the classes for your protocol handler. List all classes and separate each entry with a space.
5. Make sure that the connector can pick up the new classes. Edit the `start_xml.bat` (`start_xml.sh` for Linux) so that the new .jar file is included in the CLASSPATH.

Protocol handler framework methods

The following section describes the Protocol Handler Framework method used when designing a new protocol handler or modifying an existing protocol handler.

getContent ()

The `getContent()` method is used for business object processing. It does the following:

- Uses the MimeType and BOPrefix business object attributes to determine the appropriate data handler instance to create.
- Sends a business object to the appropriate data handler for conversion, and then and sends the file to a URL.
- Receives a response stream from the destination URL and invokes a data handler instance to convert the stream into a business object.
- Returns a business object back to the original caller (such as the connector).

Syntax

```
public abstract Object getContent (Object input, String mimeType,
String BOprefix,Long ServerTimeout String Credentials) throws IOException
```

Parameters

input	Specifies the business object interface (the business object to send)
mimeType	Specifies the MIME type of the data being passed to the data handler
BOprefix	Specifies the BOPrefix of the data being passed to the data handler
Long ServerTimeout	Specifies the amount of time the server waits before timing out
Credentials	Authentication credentials

Return values

Returns a business object interface

Calling the WebSphere Business Integration Adapter-provided protocol handler

The following code example illustrates how to call the WebSphere Business Integration Adapter-provided protocol handler.

```
try
{
// set the system property, so that Java knows where to look for
// the protocol handlers. You only need to do it once.
Properties prop = System.getProperties();
prop.put("java.protocol.handler.pkgs",
"com.crossworlds.connectors.utils.ProtocolHandlers");

URL url = new URL("http://www.crossworlds.com");
CWURLConnection uc = (CWURLConnection) url.openConnection();
BusinessObjectInterface respBO = (BusinessObjectInterface)
uc.getContent (input, mime, prefix, Long ServerTimeout, Credentials);

}
catch (Exception XX)
{
//flag error
}
```

Custom protocol handler sample code

The following sample code can be used as a guide when developing a custom protocol handler.

```
/**
 * This package hierarchy is used to write the Protocol Handler.
 * [ProtocolName] should be substituted with the name of the protocol
 * for which the handler is being written.
 * For example com.crossworlds.connectors.utils.ProtocolHandlers.ftp
   or com.crossworlds.connectors.utils.ProtocolHandlers.http
 */
package com.crossworlds.connectors.utils.ProtocolHandlers.[ProtocolName];


import CxCommon.BusinessObjectInterface;
import com.crossworlds.connectors.utils.ProtocolHandlers.CWURLConnection;
import com.crossworlds.DataHandlers.DataHandler;

import AppSide_Connector.JavaConnectorUtil;

import java.net.*;
import java.io.*;

/**
 * The handler class creates a ProtocolNameConnection class instance
 * It is invoked indirectly via Java's URL getContent() mechanism.
 *
 * how to use it:
 * System.setProperty ("java.protocol.handler.pkgs",
 *"com.crossworlds.ProtocolHandler");
 *URL url = new URL ("the URL");
 *CWURLConnection uc  = (CWURLConnection) url.openConnection ();
* /
public class Handler
{
   // this will return the appropriate URLConnection
   // But the constructor takes only one argument - the URL.  As this
   // is called by Java Networking Framework.
   public URLConnection openConnection(URL url) throws IOException
   {
      // you can pass in any parameters here.
        return new MyURLConnection (url);
   }
}

class MyURLConnection extends CWURLConnection {

   /**
    * This is instantiated by URL.openConnection()
    */
   public MyURLConnection(URL url)
   {
      // store this URL some where
   }

   /**
    * This method returns the content type of the data
    */
   public String getContentType()
   {
      // here is where you have to determine the content Type (aka
      // Mimetype) of URL streams
   }

   /**
    *  This method is used to create a connection
```

```
     */
    public synchronized void connect() throws IOException
    {
       // you might call super().connect as it suffices most of the
       // time.
       // If it is custom protocol, do the handshaking stuff here
    }

    /**
     * getContent () : The getContent method used by CrossWorlds.
     * This method takes in 5 parameters
     *   - input Object,
     *   - content type for the data &
     *   - Business Object Prefix to * be used to create the Business
     *   Object name
     *       - Long ServerTimeout
     *   - Credentials
     *   It returns an appropriate Object back to the caller.  This
     *   method interacts with the DataHandler using the exposed APIs
     *   for the DataHandler.
     */
    public Object getContent(Object input,  String mimeType,
  String BOprefix, Long ServerTimeout, String Credentials)
                                 throws IOException
    {
       // log a message
       JavaConnectorUtil.logMessage
       ("logging a message", JavaConnectorUtil.XRD_INFO);

       // write a trace
       if (JavaConnectorUtil.isTraceEnabled (JavaConnectorUtil.LEVEL3))
          JavaConnectorUtil.traceWrite  (JavaConnectorUtil.LEVEL3,
          "Level 3 trace msg");


       // get a datahandler
       DataHandler dh = DataHandler.createHandler (null, mimeType, BOprefix);

       InputStream in = dh.getStreamFromBO
       ( (BusinessObjectInterface) input, null);

       // Send this to URL
          - read data from Input Stream
          - write to URL
          - repeat until input stream is drained.

       // Now read the response
       String replyString = // some how read the reply from URL
       String outputType = // get the mime of reply some how

 // remember to get a fresh DH, as the incoming data may be of
 // different mime type than was originally received by the
 // protocol handler
       DataHandler dh2 = DataHandler.createHandler
       (null, outputType, BOprefix);

       BusinessObjectInterface replyBO = dh2.getBO
       (replyString, outputType);

       return replyBO;  // DONE!

    }

 }
```

Appendix A. 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.

If your adapter supports DB2 Information Integrator, use the WMQI options and the DB2 II standard properties (see the Notes column in Table 10 on page 35.)

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

New properties

These standard properties have been added in this release:

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

Standard connector properties overview

Connectors have two types of configuration properties:

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

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

This section describes how to start Connector Configurator 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.

© Copyright IBM Corp. 2000, 2004

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 10 on page 35.

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 10 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 10, the phrase "RepositoryDirectory is set to <REMOTE>" indicates that the broker is InterChange Server Express.

Table 10. 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	

Table 10. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
BiDi.Application	Any valid combination of these bidirectional attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (five letters)	Component restart	This property is valid only if the value of BiDi.Transforma tion is true
BiDi.Broker	Any valid combination of these bidirectional attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (five letters)	Component restart	This property is valid only if the value of BiDi.Transformation is true. If the value of BrokerType is ICS, the property is read-only.
BiDi.Metadata	Any valid combination of these bidirectional attributes: 1st letter: I,V 2nd letter: L,R 3rd letter: Y, N 4th letter: S, N 5th letter: H, C, N	ILYNN (five letters)	Component restart	This property is valid only if the value of BiDi.Transformation is true.
BiDi.Transformation	true or false	false	Component restart	This property is valid only if the value of BrokerType is not WAS
BrokerType	ICS	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.
CommonEventInfrastruc ture	true or false	false	Component restart	
CommonEventInfrastruc tureURL	A URL string, for example, corbaloc:iiop: host:2809.	No default value.	Component restart	This property is valid only if the value of CommonEvent Infrastructure is true.
ConcurrentEventTrig geredFlows	1 through 32,767	1	Component restart	This property is valid only if the value of RepositoryDirectory is set to <REMOTE> and the value of BrokerType is ICS.
ContainerManagedEvents	Blank or JMS	Blank	Component restart	This property is valid only when the value of Delivery Transport is JMS.

Table 10. Summary of standard configuration properties (continued)

Property name	**Possible values**	**Default value**	**Update method**	**Notes**
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`.
jms.ListenerConcurrency	`1 through 32767`	`1`	Component restart	This property is valid only if the value of jms.TransportOptimized is `true`.
jms.MessageBrokerName	If the value of jms.FactoryClassName is `IBM`, use `crossworlds.queue. manager`.	`crossworlds.queue. manager`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS` .
jms.NumConcurrentRequests	Positive integer	`10`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS` .

Table 10. Summary of standard configuration properties (continued)

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

Table 10. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
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`
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	

Table 10. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
ResultsSetEnabled	`true` or `false`	`false`	Component restart	Used only by connectors that support DB2II.
ResultsSetSize	Positive integer	0 (means the results set size is unlimited)	Component restart	Used only by connectors that support DB2II. This property is valid only if the value of ResultsSetEnabled is `true`.
RHF2MessageDomain	`mrm` or `xml`	`mrm`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS` and the value of WireFormat is `CwXML`.
SourceQueue	Any valid WebSphere MQ queue name	*<CONNECTORNAME>* `/SOURCEQUEUE`	Agent restart	This property is valid only if the value of ContainerManagedEvents is `JMS`.
SynchronousRequest Queue	Any valid queue name.	*<CONNECTORNAME>* `/SYNCHRONOUSREQUEST QUEUE`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS`.
SynchronousRequest Timeout	`0` to any number (milliseconds)	`0`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS`.
SynchronousResponse Queue	Any valid queue name	*<CONNECTORNAME>* `/SYNCHRONOUSRESPONSE QUEUE`	Component restart	This property is valid only if the value of DeliveryTransport is `JMS`.
TivoliMonitorTransaction Performance	`true` or `false`	`false`	Component restart	
WireFormat	`CwXML` or `CwBO`	`CwXML`	Agent restart	The value of this property must be `CwXML` if the value of RepositoryDirectory is not set to <REMOTE>. The value must be `CwBO` if the value of RepositoryDirectory is set to <REMOTE>.
WsifSynchronousRequest Timeout	`0` to any number (milliseconds)	`0`	Component restart	This property is 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.

BiDi.Application

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

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

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

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

BiDi.Broker

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

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

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

BiDi.Metadata

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

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

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

BiDi.Transformation

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

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

The default value is `false`.

BrokerType

The BrokerType property identifies the integration broker type that you are using. The 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 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 Espress (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 OS/400*.

DuplicateEventElimination

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

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

The default value is `false`.

EnableOidForFlowMonitoring

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

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

The default value is `false`.

FaultQueue

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

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

jms.FactoryClassName

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

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

jms.ListenerConcurrency

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

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

The default value is 1.

jms.MessageBrokerName

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

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

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

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

jms.NumConcurrentRequests

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

The default value is `10`.

jms.Password

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

There is no default value.

jms.TransportOptimized

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

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

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

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

The default value is `false`.

jms.UserName

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

There is no default value.

JvmMaxHeapSize

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

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

The default value is 128m.

JvmMaxNativeStackSize

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

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

The default value is 128k.

JvmMinHeapSize

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

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

The default value is 1m.

ListenerConcurrency

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

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

The default value is 1.

Locale

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

A locale name has the following format:

`ll_TT.codeset`

where:
ll is a two-character language code (in lowercase letters)
TT is a two-letter country or territory code (in uppercase letters)
codeset is the name of the associated character code set (may be optional).

By default, only a subset of supported locales are listed. To add other supported values to the list, you modify the \Data\Std\stdConnProps.xml file in the <*ProductDir*>\bin directory. For more information, refer to the Connector Configurator 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 OS/400*.

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.

ResultsSetEnabled

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

The default value is `false`.

ResultsSetSize

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

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

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

RHF2MessageDomain

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

This is an example header:

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

This property is 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 43.

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

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

SynchronousRequestQueue

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

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

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

SynchronousRequestTimeout

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

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

The default value is `0`.

SynchronousResponseQueue

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

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

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

TivoliMonitorTransactionPerformance

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

The default value is `false`.

WireFormat

The WireFormat property specifies the message format on the transport:

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

Appendix B. Connector Configurator Express

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

You use Connector Configurator Express to:
- Create a connector-specific property template for configuring your connector
- Create a configuration file
- Set properties in a configuration file

The topics covered in this appendix are:
- "Overview of Connector Configurator Express"
- "Creating a connector-specific property template" on page 57
- "Creating a new configuration file" on page 59
- "Setting the configuration file properties" on page 62

Overview of Connector Configurator Express

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

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

© Copyright IBM Corp. 2000, 2004

your system, in which case, you simply use that. If not, follow the steps in "Creating a new template" on page 57 to set up a new one.

Running connectors on Linux

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.

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

Starting Connector Configurator Express

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

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

Running Configurator in stand-alone mode

You can run Connector Configurator Express without running System Manager and work with connector configuration files, irrespective of your broker.

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**, you can select ICS.

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 62.)

Running Configurator 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.
4. When you click the pull-down menu next to **System Connectivity Integration Broker**, you can select ICS.

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" on page 57.
- To use an existing file, simply modify an existing template and save it under the new name. You can find existing templates in your `\ProductDir\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
  Property Subtype
  Updated Method
  Description
- **Flags**
  Standard flags
- **Custom Flag**
  Flag

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

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

Specifying values

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

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

To create a new property value:

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

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

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

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

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

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

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

Setting dependencies

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

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

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

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

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

Setting pathnames

Some general rules for setting pathnames are:

- The maximum length of a filename in Windows and Linux is 255 characters.
- In Windows, the absolute pathname must follow the format [Drive:][Directory]\filename: for example, C:\WebSphereAdapters\bin\Data\Std\StdConnProps.xml
  In Linux the first character should be /.
- Queue names may not have leading or embedded spaces.

Creating a new configuration file

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

You also select an operating system for extended validation on the file. The toolbar has a droplist called **Target System** that allows you to select the target operating system for extended validation of the properties. The available options are: Windows, Linux, i5/OS, Other (if not Windows or Linux), and None-no extended validation (switches off extended validation). The default on startup is Windows.

To start Connector Configurator Express:

- In the System Manager window, select **Connector Configurator Express** from the **Tools** menu. Connector Configurator Express opens.

- In stand-alone mode, launch Connector Configurator Express.

To set the operating system for extended validation of the configuration file:
- Pull down the **Target System:** droplist on the menu bar.
- Select the operating system you are running on.

Then select **File>New>Connector Configuration**. In the New Connector window, enter the name of the new connector.

You also need to select an integration broker. The broker you select determines the properties that will appear in the configuration file. To select a broker:
- In the **Integration Broker** field, select ICS.
- Complete the remaining fields in the **New Connector** window, as described later in this chapter.

Creating a configuration file from a connector-specific template

Once a connector-specific template has been created, you can use it to create a configuration file:
1. Set the operating system for extended validation of the configuration file using the **Target System:** droplist on the menu bar (see "Creating a new configuration file" above).
2. Click **File>New>Connector Configuration**.
3. The **New Connector** dialog box appears, with the following fields:
   - **Name**

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

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

     Click ICS.
   - **Select Connector-Specific Property Template**

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

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

file, and click **Save**. The status display in the message panel of Connector Configurator 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.

6. 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_XML.txt for the XML connector).
- An ICS repository file.
  Definitions used in a previous ICS implementation of the connector may be available to you in a repository file that was used in the configuration of that connector. Such a file typically has the extension `.in` or `.out`.
- A previous configuration file for the connector.
  Such a file typically has the extension *.cfg.

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

To use an existing file to configure a connector, you must open the file in Connector Configurator 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`)

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

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

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

1. Start System Manager. A configuration can be opened from or saved to System Manager only if System Manager has been started.

2. Start Connector Configurator 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 you have the correct broker. If not, change the broker value before you configure the connector. To do so:

1. Under the **Standard Properties** tab, select the value field for the `BrokerType` property. In the drop-down menu, select the value `ICS`.
2. The Standard Properties tab will display the connector properties associated with the selected broker. The table shows **Property name**, **Value**, **Type**, **Subtype** (if the Type is a string), **Description**, and **Update Method**.
3. You can save the file now or complete the remaining configuration fields, as described in "Specifying supported business object definitions" on page 65..
4. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

   Before you created the configuration file, you used the **Target System** droplist that allows you to select the target operating system for extended validation of the properties.

   Before it saves the file, Connector Configurator 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.

   If you have elected to use the extended validation feature by selecting a value of Windows, Linux, and i5/OS, or Other from the **Target System** droplist, the system will validate the property subtype s well as the type, and it displays a warning message if the validation fails.

Setting the configuration file properties

When you create and name a new connector configuration file, or when you open an existing connector configuration file, Connector Configurator 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 on all brokers:

- Standard Properties
- Connector-specific Properties
- Supported Business Objects
- Trace/Log File values

- Data Handler (applicable for connectors that use JMS messaging with guaranteed event delivery)

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

For connectors running on InterChange Server Express, values for these properties are also required:
- Associated Maps
- Resources
- Messaging (where applicable)
- Security

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

To change the value of a standard property:
1. Click in the field whose value you want to set.
2. Either enter a value, or select one from the drop-down menu if it appears.

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

3. After entering all the values for the standard properties, you can do one of the following:
   - To discard the changes, preserve the original values, and exit Connector Configurator 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.

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

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

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

Setting connector-specific configuration properties

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

1. Right-click in the top left portion of the grid. A pop-up menu bar will appear. Click **Add** to add a property. To add a child property, right-click on the parent row number and click **Add child**.
2. Enter a value for the property or child property.

   Note: If the property has a Type of String, you can select a subtype from the Subtype droplist. This subtype is used for extended validation of the property.
3. To encrypt a property, select the **Encrypt** box.
4. To get more information on a particular property, left-click the entry in the Description column for that property. If you have Extended Help installed, a hot button will appear. When you click on the hot button, a Help window will open and display details of the standard property.

   Note: If the hot button does not appear, no Extended Help was found for that property.
5. Choose to save or discard changes, as described for "Setting standard connector properties" on page 63.

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

Extended Help files located in
<*ProductDir*>\bin\Data\App\Help\<*AdapterHelpName*>\<*RegionalSetting*>\. See the AdapterHelpName property described in the Standard Properties appendix.

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

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

Encryption for connector properties

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

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

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

Update method

Refer to the descriptions of update methods found in the Standard Properties appendix, under "Standard connector properties overview" on page 33.

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 in order to perform event notification or additional configuration (using meta-objects) with their applications.

InterChange Server Express as your broker

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

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

1. Click an empty field in the **Business Object Name** list. A drop list displays, showing all the business object definitions that exist in the System Manager project.
2. Click on a business object to add it.
3. Set the **Agent Support** (described below) for the business object.

4. In the File menu of the Connector Configurator 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 Binding**

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

  Explicit binding is required only when more than one map exists for a particular supported business object. When 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.

Security

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

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

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

- For Windows:
  `<ProductDir>\connectors\security\<connectorname>.jks`
- For Linux and i5/OS:
  `/ProductDir/Connectors/security/<connectorname>.jks`

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

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

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

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

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

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

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

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

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

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

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

Setting trace/log file values

When you open a connector configuration file or a connector definition file, Connector Configurator 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 InterChange Server Express 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.

Changing a configuration file

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

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

To change your broker selection within an existing configuration file (optional):
- Open the existing configuration file in Connector Configurator Express.
- Select the **Standard Properties** tab.
- In the **BrokerType** field of the Standard Properties tab, select the value that is appropriate for your broker.
  When you change the current value, the available tabs and field selections in the

properties window will immediately change, to show only those tabs and fields that pertain to the new broker you have selected.

Completing the configuration

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

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

Using Connector Configurator 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>
```

Appendix C. Overview of XML adapter sample scenarios

Consider the situation in which a company wants to use the WebSphere Business Integration Adapter for XML to read XML documents from and POST XML documents to Web Servers. Below are two different sample scenarios designed to be simple and show the basic points of the XML Adapter's functionality.

- Installation of the XML sample scenario. This fictitious scenario involves two integrations that represent the two directions of data exchange:
  - The first integration starts with an ″XML_REQUEST_Order″ event sent from WebSphere MQ Integrator Broker to an WebSphere MQ queue. The XML Adapter accepts the event from the queue and invokes the XML DataHandler to convert the event to an XML Document. The XML document is POSTed to a Webserver. The Webserver will respond to the XML Adapter with it's own XML.
  - In the second integration, the XML Adapter will poll a URL for an XML document. Once read, the XML Adapter will send the document through the XML DataHandler. The DataHandler will return a response event that will be sent to an WebSphere MQ queue for the WebSphere MQ Integrator Broker to read.
- Installation of the XML sample scenario with InterChange Server Express connectivity. This fictitious scenario involves two integrations that represent the two directions of data exchange:
  - The first integration starts with an ″XML_REQUEST_Order″ object sent from the PortConnector to the XML Adapter via the ″Port_To_XML″ collaboration object. The XML Adapter will invoke the XML DataHandler to convert the request business object to an XML Document. The XML is POSTed to a Webserver. The Webserver will respond to the XML Adapter, which converts the response XML to the response business object sent back to InterChange Server Express.
  - In the second integration, the XML Adapter will poll a URL for an XML document. Once read, the XML Adapter will send the document through the XML DataHandler to convert it to a response business object that will be sent to the InterChange Server Express. Then, via one of two collaborations, ″XML_To_Port_Customer″ or ″XML_To_Port_Manifest″, the events will be sent to the Port Connector.

Installation of the XML sample scenario with InterChange Server Express connectivity

Note: In this sample, the poll will return three business objects:
- XML_Order_Customer
- XML_Order_Manifest
- XML_Order_Receipt

Only two of these business objects have collaborations offering subscriptions (Customer and Manifest), so the third business object (Receipt) will be archived by the XML Adapter to a specified location.

The following are the steps for installing and verifying the sample scenario:

- Pre-installation notes and assumptions.
- Installation of the sample scenario.
- Running the service call request scenario.
- Running poll scenario.
- Summary.

Pre-installation notes and assumptions.

1. You have installed and are experienced with WebSphere Business Integration Server Express Adapters.
2. You have installed and are experienced with IBM WebSphere MQ
3. You have installed the WebSphere Business Integration Server Express Adapter for XML.
4. You have a Webserver set up to handle java serlvets.
5. Whenever %WBIA_RUNTIME% is mentioned in this document, it refers to the folder containing your current adapters installation.
6. All environment variables and file separators are specified in the Windows format. Please make the appropriate changes if running on Linux platforms. (example: %WBIA_RUNTIME%\connectors would be ${WBIA_RUNTIME}/connectors)

Installation of the sample scenario.

1. **Load business objects into repository:**

   Start InterChange Server Express, and using System Manager, select the ″Open From File″ menu item from the ″File″ menu of Business Object Designer Express.

   - Windows: Load the repository file labeled ″Sample_XML_Order_Objects.in″ located in the %CROSSWORLDS%\connectors\XML\Samples \WebSphereICS directory.
   - Linux:
     Load the repository file labeled ″Sample_XML_Order_Objects.in″ located in the %CROSSWORLDS%/connectors/XML/Samples/WebSphereICS directory.
   - i5/OS:
     Load the repository file labeled ″Sample_XML_Order_Objects.in″ located in the /QIBM/UserData/WBIServer44/InterChangeServerName /connectors/XML/Samples/WebSphereICS directory.
2. **Configure XML connector and Port connector CFG file:**

   Open the file: %WBIA_RUNTIME%\connectors\XML\samples\ WebSphereICS\XMLConnector.cfg

   The following properties in the file may need to be modified for your particular setup. See the ″Guide to the WebSphere Business Integration Server Express Adapter for XML″ for detailed information. If the property represents a path or filename that does not already exist, it needs to be created:

- QueueManager
- RepositoryDirectory
- ArchiveDirectory

Open the file: %WBIA_RUNTIME%\connectors\XML\samples\WebSphereICS\PortConnector.cfg

The following properties in the file may need to be modified for your particular setup. If the property represents a path or filename that does not already exist, it needs to be created:

- QueueManager
- RepositoryDirectory

3. **Specifying the XML connector CFG file:**
   - ***For Windows:***

     Open the properties of the shortcut for the XML CONNECTOR. As the last argument in the target, add ″-c″ + <full path and filename for the XMLConnector.cfg file>.

     For example:

     ```
     c%WBIA_RUNTIME%\connectors\XML\samples\WebSphereICSBroker\
     XMLConnector.cfg.
     ```
   - ***For Linux:***

     Open the file: %WBIA_RUNTIME%/bin/connector_manager_XML. Set the value of the AGENTCONFIG_FILE property to ″-c″ + <full path and filename
     for the XMLConnector.cfg file>.

     For example:

     ```
     AGENTCONFIG_FILE=c${WBIA_RUNTIME}/connectors/XML/samples/
     WebSphereICS/XMLConnector.cfg.
     ```
4. **Modify the servlets for use in your envirnment**

   The following changes must be made to the servlets:
   - For PollXMLOrder.java:

     On line 41 of the source file, the value of the outFileName String must be changed to be the name of the file on the local system to which the servlet should log the incoming XML message. On line 56 of the source file, the value passed to the FileInputStreamconstructor should be the exact file name on the local system for the delivered SamplePollingInput.xml file.
   - For MirrorXMLOrder.java:

     No Changes Necessary.
5. **Configure the XML Poll business object:**

   Start the Business Object Designer Express. Select File->Open From File, and choose the file named ″XML_POLL_Order.xsd″. This will open the business object labeled″XML_POLL_Order″. The first attribute of this business object should be named ″URL″.Change the URL attribute's default value to the location to which the XML Adapter will listen for XML responses (ie. PollXMLOrder.java). Save the business object by replacing the file XML_POLL_Order.xsd with the modified business object.
6. **Configure the web server:**

   Compile the delivered servlets. The generated class files should be moved to the correct directory for the web server to pick them up and run them. Also, any additional steps to register the servlets should be done (the exact requirements will depend entirely on the web server being used).

Running the service call request scenario.

1. **Start:**
   - Start XML Adapter
   - Start the WebServer.
   - Start one instance of the Visual Test Connector.
2. **Simulate Port connector:**

   Using the Visual Test Connector, define a profile for the ″PortConnector″. Select FILE -> OPEN PROFILE from the Test Connector's menu. Click the ADD button, and specify the connector configuration file. Open the file: %WBIA_RUNTIME%\connectors\XML\samples\WebSphereICS\XMLConnector.cfg. Click OK on the following two windows, and connect your agent.
3. **Load test data:**

   Using the Test Connector simulating the ″PortConnector″, Select EDIT -> LOAD BO from the menu. Specify the connector configuration file, %WBIA_RUNTIME%\connectors\XML\samples\WebSphereICS\XMLConnector.cfg.
4. **Set the URL:**

   Open the test data that was loaded into the Test Connector. Modify the value of the URL attribute to the location to which the XML Adapter should POST it's xml request.
5. **Send test data:**

   Using the Test Connector simulating the ″PortConnector″, click on the loaded Test BO. Select REQUEST -> SEND from the menu.
6. **Check for successful processing:**

   To ensure successful processing, check that the XML Adapter received an event, converted the event to an XML document, POSTed it to the WebServer, received a response and successfully parsed the response.

Running the Poll scenario

1. **Start:**
   - XML Adapter
   - WebServer
   - One instance of the Visual Test Connector
2. **Simulate Port connector:**

   Using the Visual Test Connector, define a profile for the ″PortConnector″. Select FILE -> OPEN PROFILE from the Test Connector's menu. Click the ADD button, and specify the connector configuration file, %WBIA_RUNTIME%\connectors\XML\samples\WebSphereICS\XMLConnector.cfg..
   Click OK on the following two windows, and connect your agent.
3. **Poll sample data:**
   - ***Windows:***

     The PollFrequency has already been set to key. From the command window in which.the XML Adapter was started, type the letter 'p' followed by the Enter key. Stop the XML Adapter. Open the XMLConnector.cfg file, change the

PollFrequency to a number like 30000 (polls/millisecond) and save the file. Restart the connector, and wait 30 seconds for the connector to poll.

- ***Using Linux:***

  Stop the XML Adapter. Open the XMLConnector.cfg file, change the PollFrequency to a number like 30000 (polls/millisecond) and save the file. Restart the connector, and wait 30 seconds for the connector to poll.

4. **Accept the request using the Port connector:**

   The XML Adapter will receive an XML document, convert it to a message that will be placed on an WebSphere MQ queue. Accept the request using the Test Connector, and reply with a successful response.

5. **Check for successful processing:**

   To ensure successful processing, check that the data in the accepted request in the
   Test Connector and the Archive Directory correspond to the events from the SamplePollingInput.xml file provided in the sample

Summary:

If you've performed all the above steps successfully, you should have a working sample scenario that uses the XML Adapter and XML DataHandler to exchange XML documents between a WebSphereICS and a Webserver.

Installation of the XML sample scenario with InterChange Server Express connectivity

Note: In this sample, the poll will return three business objects:

- XML_Order_Customer
- XML_Order_Manifest
- XML_Order_Receipt

Only two of these business objects have collaborations offering subscriptions (Customer and Manifest), so the third business object (Receipt) will be archived
by the XML Adapter to a specified location.

The following are the steps for installing and verifying the sample scenario.

- Pre-installation notes and assumptions.
- Installation of the sample scenario.
- Running the service call request scenario.
- Running the poll senario.
- Summary.

Pre-installation notes and assumptions.

1. You have installed and are experienced with WebSphere Business Integration Server Express Adapters.
2. You have installed and are experienced with WebSphere Business Integration Server Express.
3. You have installed the WebSphere Business Integration Server Express Adapter for XML.
4. You have a Webserver set up to handle java serlvets.

5. Whenever %WBIA_RUNTIME% is mentioned in this document, it refers to the folder containing your current adapter installation.
6. All environment variables and file separators are specified in the Windows 2003 format. Please make the appropriate changes if running on Linux and i5/OS platforms. Example:
%WBIA_RUNTIME%\connectors would be %WBIA_RUNTIME%/connectors).

Installation of the sample scenario.

1. **Load business objects into repository:**

   Start InterChange Server Express, and using System Manager, select the ″Open From File″ menu item from the ″File″ menu of the Business Object Designer Express.

   Note: After loading the repository file, confirm that all of the business objects have been loaded. There should be a total of 12.
   Windows:
   Load the repository file labeled ″Sample_XML_Order_Objects.in″ located in the %WBIA_RUNTIME%\connectors\XML\Samples\WebSphereICS directory. Confirm that the business objects have been loaded. There should be 12 in total.
   Linux:
   Load the repository file labeled ″Sample_XML_Order_objects.in″ located in the %CROSSWORLDS%/connectors/XML/Samples/ in WebSphereICS directory.
   i5/OS:
   Load the repository file labeled ″Sample_XML_Order_Objects.in″ located in the /QIBM/UserData/WBIServer44/InterChangeServerName /connectors/XML/Samples/WebSphereICS directory.
   You must map a network drive from a Windows system to the i5/OS system (in the Map Network Drive dialog box, type \\os400Name\root for the folder field where *os400Name* is the name or IP address of the i5/OS system). Then navigate to the drive to find the file.
2. **Load connectors into repository:**

   Using System Manager, select the ″Open From File″ menu item from the ″File″ menu of Connector Configurator Express .

   Note: After loading the connectors into the repository, confirm that the XMLConnector and PortConnector definitions have been loaded.
   - **Windows**:

     Load the repository file labeled ″Sample_XML_Order_Connectors.in″ located in the %CROSSWORLDS%\connectors\XML\Samples of the WebSphere InterChange Server Express directory.
   - **Linux**:

     Load the repository file labeled ″Sample_XML_Order_Connectors.in″ located in the %CROSSWORLDS%/connectors/XML/Samples of the WebSphere InterChange Server Express directory.
   - **i5/OS**:

     Load the repository file labeled ″Sample_XML_Order_Connectors.in″ located in /QIBM/UserData/WBIServer44/InterchangeServerName /connectors/XML/samples of the WebSphereICS directory.
3. **Configure XML connector:**

   Using System Manager, double-click on the XML CONNECTOR definition so that the Connector Designer is launched. The following Application

Config Property values need to be modified for your specific file structure. These paths and/or files need to be created if they don't already exist:

- ArchiveDirectory.

4. **Load collaboration templates and objects into repository:**

   Using System Manager, select the ″Open From File″ menu item from the ″File″ menu.

   Note: After loading the collaboration templates and objects into the repository, confirm that three template definitions and three collaboration objects have been loaded.

   - **Windows**:

     Load the repository file labeled ″Samle_XML_Order_Collaborations.in″ located in the %CROSSWORLDS%\connectors\XML\samples\WebSphereICS directory.
   - **Linux**:

     Load the repository file labeled″Sample_XML_Order_collaborations.in″ located in the %CROSSWORLDS%/connectors/XML/Sample of the WebSphere Interchange Server Express directory.
   - i5/OS:

     Load the repository filed labeled ″Sample_XML_Order_Collaborations.in ″ located in the /QIBM/UserData/WBIServer44/InterChangeServerName /connectors/XML/Samples/WebSphereICS directory.
     You must map a network drive from a Windows system to the i5/OS system (in the Map Network Drive dialog box, type \\os400Name\root for the folder field where *os400Name* is the name or IP address of the i5/OS system). Then navigate to the drive to find the file.

5. **Compile the collaboration templates:**

   Using System Manager, right click on the folder labeled *Collaboration Templates* and select **Compile all** from the drop down list.

6. **Modify the servlets for use in your environment:**

   The following changes must be made to the servlets:

   - For PollXMLOrder.java:

     On line 41 of the source file, the value of the outFileName String must be changed to be the name of the file on the local system to which the servlet should log the incoming XML message. On line 56 of the source file, the value passed to the FileInputStream constructor should be the exact file name on the local system for the delivered SamplePollingInput.xml file.
   - For MirrorXMLOrder.java:

     No Changes Necessary.

7. **Configure the XML Poll business object:**

   From System Manager, open the business object labeled ″XML_POLL_Order″. The first attribute of this business object should be named ″URL″. Change the URL attribute's default value to the location to which the XML Adapter will listen for XML responses (ie. PollXMLOrder.java).
   Save the business object to the server.

8. **Configure the Web server:**

   Compile the delivered servlets. The generated class files should be moved to the correct directory for the web server to pick them up and run them. Also, any additional steps to register the servlets should be done (the exact requirements will depend entirely on the web server being used).

9. **InterChange Server Express restart:**

   Reboot Interchange Server Express to ensure that all changes take effect. Use the System View from System Manager to ensure that all of the collaboration objects and connector controllers are in a green state.

Running the service call request scenario.

1. **Start:**
   - Interchange Server Express, if not already running.
   - XML Adapter.
   - Web server.
   - One instance of the Visual Test Connector.
2. **Simulate the Port connector:**

   Using the Test Connector, define a profile for the ″PortConnector″. Select FILE -> CONNECT AGENT from the Test Connector's menu to begin simulating the agent.
3. **Load test data:**

   Using the Test Connector simulating the ″PortConnector″, Select EDIT -> LOAD BO from the menu. Load the *sampleOrderData.bo* file from one of the following paths:
   - **Windows**: %CROSSWORLDS%\connectors\XML\samples\WebShereICS
   - **Linux**: %CROSSWORLDS%/connectors/XML/samples/WebSphereICS
   - **i5/OS**: /QIBM/UserData/WBIServer44/InterchangeServerName /connectors/XML/samples/WebSphereICS
4. **Set the URL:**

   Open the test data that was loaded into the Test Connector. Modify the value of the URL attribute to the location to which the XML Adapter should POST it's xml request.
5. **Send test data:**

   Using the Test Connector simulating the ″PortConnector″, click on the loaded Test BO. Select REQUEST -> SEND from the menu.
6. **Check for successful processing:**

   To ensure successful processing, check that the XML Adapter received an event, converted the BO to an XML document, POSTed it to the WebServer, received a response, parsed the response, and sent it back to the collaboration.

Running the Poll scenario

1. **Start**
   - Interchange Server Express, if not already running.
   - XML Adapter.
   - Web server.
   - Start one instance of the Visual Test Connector.

2. **Simulate the Port connector:**

   Using the Test Connector, define a profile for the "PortConnector". Select FILE -> CONNECT AGENT from the Test Connector's menu to begin simulating the agent.
3. **Poll sample data:**

   The PollFrequency has already been set to key. From the command window in which the XML Adapter was started, type the letter 'p' followed by the Enter key.

   Stop the XML Adapter. Open the XML Adapter Controller, change the PollFrequency to a number like 30000 (polls/millisecond). Restart the connector, and wait 30 seconds for the connector to poll.
4. **Accept the request using the Port connector:**

   The XMLConnector will receive an XML document, convert it to a CrossWorlds business object and pass it to InterChange Server Express. InterChange Server Express passes the event to two different collaborations that have subscriptions for this event. The collaborations will pass the events to the PortConnector. Accept the requests using the Test Connector, and reply with a successful response to both of the events.
5. **Check for successful processing:**

   To ensure successful processing, check that the data in the accepted request in the Test Connector and the Archive Directory correspond to the events from the SamplePollingInput.xml file provided in the sample.

Summary:

If you've performed all the above steps successfully, you should have a working sample scenario that uses the XML Adapter and XML DataHandler to exchange XML documents between InterChange Server Express and a Webserver.

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:

© Copyright IBM Corp. 2000, 2004

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 Eclipse Project
(http://www.eclipse.org/).

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

Index

Printed in USA