IBM WebSphere Business Integration Adapters

Adapter for PeopleSoft (Version 7.x) User Guide

Adapter Version 3.5.x

Note!
Before using this information and the product it supports, read the information in "Notices" on page 67.

30September 2004

This edition of this document applies to the adapter for PeopleSoft 7.x (5724-G98) version 3.5.0.

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

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

Contents

About this document

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

This document describes the installation, configuration, and business object development for the IBM WebSphere Business Integration Adapter for PeopleSoft 7.x.

What this document includes

This document describes installation, connector property configuration, business object development, and troubleshooting for the IBM WebSphere Business Integration Adapter for PeopleSoft 7.x.

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 each 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 IBM WebSphere Business Integration Adapter Framework consultants and customers. To use the information in this document, you should be knowledgeable in the following areas:

- Connector development
- Business object development
- PeopleSoft application architecture
- PeopleSoft Tools

Related documents

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

You can install related documentation from the following sites:

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

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

- For using adapters with WebSphere InterChange Server, see the IBM WebSphere InterChange Server InfoCenters:
  http://www.ibm.com/websphere/integration/wicserver/infocenter
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- For more information about WebSphere message brokers:
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- For more information about WebSphere Application Server:
  http://www.ibm.com/software/webservers/appserv/library.html

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

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, file name, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
italic	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.

New in this release

New in Release 3.5.x

Updated in June, 2004.

The adapter now supports PeopleTools version 7.63.

New in Release 3.4.x

Updated in December, 2003.

Adapter installation information has been moved from this guide. See Chapter 2 for the new location of this information.

The adapter for PeopleSoft 7.x is no longer supported on Microsoft Windows NT.

Updated in March, 2003.

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

The IBM WebSphere Business Integration Adapter for PeopleSoft 7.x includes the connector for PeopleSoft 7.x. This adapter operates with both the InterChange Server (ICS) and WebSphere MQ Integrator integration brokers. An integration broker, which is an application that performs integration of heterogeneous sets of applications, provides services that include data routing.

This adapter includes:

- An application component specific to PeopleSoft
- PeopleSoft ODA
- A sample business object, included in the `\connectors\PeopleSoft7\samples` directory
- IBM WebSphere Adapter Framework, which consists of:
  - Connector Framework
  - Development tools (including Business Object Designer and Connector Configurator)
  - APIs (including ODK, JCDK, and CDK)

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

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

New in Release 3.3.x

Version 3.3.1 of the IBM CrossWorlds Connector for PeopleSoft 7.x includes the following new features and changes:

- Two new field mappings have been added to the Message Agent queues: `keylist` and `poll_delay`.

Chapter 1. Overview of the connector

This chapter describes the connector component of the IBM WebSphere Business Integration Adapter for PeopleSoft 7.x. The connector enables a supported integration broker to exchange business objects with PeopleSoft 7.55 and 7.57 applications.

Connectors consist of two parts: the connector framework and the application-specific component. The connector framework, whose code is common to all connectors, acts as an intermediary between the integration broker and the application-specific component. The application-specific component contains code tailor to a particular application. 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 connector framework and the application-specific component. 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, or the WebSphere Business Integration Adapters Implementation Guide for MQ Integrator.

The connector works with the following databases: Informix, SQLBase, Oracle, Sybase, Microsoft, DB2, DB2ODBC, DB2MDI, DB2400, DB2Unix.

Communication between the connector and PeopleSoft

Business object operations

The connector inserts data into PeopleSoft by using the PeopleSoft Message Agent API and a set of message definitions for each business object.

Event notification

For event notification, the connector uses PeopleCode to populate an event queue table when an event of interest to the integration broker occurs in PeopleSoft. The connector polls the event queue table at a configurable interval, retrieves the events by using the PeopleSoft Message Agent API, processes the events, and archives the events.

Processed events and unsubscribed events are archived by means of PeopleCode on the event work table. The code removes events from the event table and copies them to an archive table. The event and archive tables are installed with database upgrades for each database supported by the connector. See “Performing a selective application upgrade” on page 5 for information on the event queue table.

Chapter 2. Installing and configuring the connector

This chapter describes how to install the IBM WebSphere Business Integration Adapter for PeopleSoft 7.x and how to configure the PeopleSoft application to work with the connector. It contains the following sections:

Adapter environment

Before installing, configuring, and using the adapter, you must understand its environment requirements. They are listed in the following section.

Broker compatibility

The adapter framework that an adapter uses must be compatible with the version of the integration broker (or brokers) with which the adapter is communicating. Version 3.5.0 of the adapter for PeopleSoft 7 is supported on the following adapter framework and integration brokers:

- **Adapter framework**:
  WebSphere Business Integration Adapter Framework versions 2.1, 2.2, 2.3.x, and 2.4.
- **Integration brokers:**
  - WebSphere InterChange Server, versions 4.1.1, 4.2, 4.2.1, 4.2.2
  - WebSphere MQ Integrator, version 2.1.0
  - WebSphere MQ Integrator Broker, version 2.1.0
  - WebSphere Business Integration Message Broker, version 5.0

See *Release Notes* for any exceptions.

Note: For instructions on installing your integration broker and its prerequisites, see the following guides.
For WebSphere InterChange Server (ICS), see *IBM WebSphere InterChange Server System Installation Guide for UNIX* or *for Windows.*
For WebSphere message brokers, see *Implementing Adapters with WebSphere Message Brokers.*

Adapter platforms

The adapter is supported on the following software.

Operating systems:

- Windows 2000

Third-party software

- PeopleTools 7.55 and 7.57, and PeopleTools 7.63

Globalization

This adapter is not DBCS (double-byte character set)-enabled or translated.

Prerequisites

This section describes the software components you must install and the tasks you must perform before installing and running the connector.

User account setup

You must create a user account in PeopleSoft for the connector. The user account can have any valid PeopleSoft username and password. It must have privileges to retrieve, insert, update and delete data from the appropriate application panels and the cw event panel in the PeopleSoft system. For example, if the connector handles customer data, the connector's user account must have privileges to access and modify data in all relevant customer panels.

The user account must be configured so that it never times out. Since the connector uses the Message Agent to insert and extract business objects from the PeopleSoft server, the Message Agent must be running when the connector accesses the PeopleSoft system. If the connector times out, the Message Agent closes. As a result, the connector logs an error message every time it accesses the PeopleSoft system — at every poll, retrieve, create, and update.

To configure the user account so that it never times out:

1. Open the People Tools Security Administrator.
2. Open the class which contains the user account.
3. Select the Never Time-Out radio button in the Time-Out Minutes group box.

Upgrading from a previous version

If you are upgrading from a previous version of the product, the configuration properties in the repository definition file may have changed. See "Configuring the connector" on page 8.

Deleting the previously installed connector

If you have an earlier version of the connector installed, you must delete the connector and as well as its customized components in the PeopleSoft environment. This includes all supporting menus, panels, records, event and archive tables, and the IBM WebSphere business integration adaptor business process.

Installing the connector

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

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

Required modifications

The connector installation includes a set of Data Mover data files containing menu definitions, panel definitions, record definitions, and PeopleCode required to support the connector. These files are used to create the following:

- Event queue tables that store event notifications, and an archive table that stores processed events.
- An Application Engine used to move events between event tables.
- A derived work record for storing temporary values and PeopleCode.
- A set of PeopleSoft message definitions for each business object/verb combination. Message definitions are stored either in the `Connector Integration` activity in the `IBM WebSphere business integration adaptor` business process or, when ICS is the integration broker, in the activities and business processes associated with supported collaborations. For example, message definitions for Employee object/verb combinations are found in the `CWEmployee` activity in the `CWEMPLOYEE` business process.

Performing a selective application upgrade

To copy the required business-integration objects into the PeopleSoft database, you must perform a selective application upgrade. The upgrade process involves copying the objects from a source database, the application upgrade database (AUDB), to the target database in the PeopleSoft environment. For detailed information on Application Upgrader, see the PeopleSoft Administration Tools guide.

Before performing the application upgrade you must create and prepare the AUDB by performing the following steps:

1. Create a new database to use as the upgrade database.
2. Launch the PeopleTools Data Mover by double clicking the `PSDMT.EXE` file.The Data Mover window appears.
3. Import the file *%ProductDir%*`\connectors\PeopleSoft\PeopleSoft7\dependencies\DataMover\CWUpgPrj.dat` into the upgrade database.
4. In Data Mover, open the script `%PeopleSoft%\script\auimport.dms`.
5. Run the script by selecting File>Run Script, or by clicking the Run Script button. The message definition data is imported. Ensure that the `Successful Completion` message appears.

At this point, it is a good idea to run a comparison report between the AUDB and the target database. However, if you are certain that there are no business-integration objects in the target database, you can skip this report and perform the application upgrade.

Perform the application upgrade as follows:

1. Log on to the AUDB.

2. In the Application Designer, open the CWUpgPrj project.
3. From the Tools menu, select Copy>Upgrade.
4. Sign on to the target database by clicking Signon. The Signon screen is displayed. Enter the name of the target database, operator ID, and password.
5. In the Copy dialog box, select the Export Project check box. Make sure that all other items in the list box are also selected, then click Copy.

   If PeopleSoft is the source application, you may skip the next two steps.
6. To complete the installation of the event tables, go to Data Designer and perform a SQL Table Create against XR_EVENT, XR_FUTURE_EVENT, TMP_FUTURE_EVENT, and XR_ARCHIVE.
7. Go to Data Designer and perform a SQL View Create against XR_SRCH.
8. In the PeopleSoft Security Administrator, open the Operator class for the connector and give this class the privileges to use all items in the EVENT_NOTIFICATION menu group.

To test that the application upgrade executed successfully:

1. Close all PeopleSoft windows.
2. Log out of the application, then log back in.
3. Select the Start Menu, and confirm that the Event Notification menu is displayed.

Importing the application engine

Note: The Application Engine is not required if the connector does not support PeopleSoft as a target application.

The Application Engine is a batch process that moves events from the future events table to the event table. To import the Application Engine follow these steps:

1. In your Data Mover window, import the file *%ProductDir*S%\connectors\PeopleSoft\PeopleSoft7\dependencies\DataMover\CWAE.dat into the production database.
2. In Data Mover, open the script CWAEIn.dms.
3. Run the script by selecting File>Run Script or by clicking the Run Script button.
4. When the script runs successfully, open the Process Scheduler and schedule XR_MOVE to run at midnight every night. For information on using the Process Scheduler, see the PeopleSoft Administration Tools manual.

Installing message definitions

Next you must import message definitions to support WebSphere Business Integration Adapter Framework business objects. The message definitions enable the connector to communicate with the PeopleSoft system via the Message Agent API. The import script file, CWBPIn.dms, is located in *%ProductDir*s%\connectors\PeopleSoft\PeopleSoft7\dependencies\DataMover. This file contains the following scripts:

Table 1. Message definition scripts

Script	Message definitions
CWBP.dat	Event notification message definitions
CWEMPL.dat	Employee message definitions
CWDEPT.dat	Department message definitions

Table 1. Message definition scripts (continued)

Script	Message definitions
CWEMPL_757.dat	Employee message definitions for use with PeopleTools 7.57

To import the message definitions into the PeopleSoft system, follow these steps:

1. In the Data Mover window, open the .dms scripts for the business objects you need to support.
2. Run the scripts by selecting File>Run Script or by clicking the Run Script button. Ensure that the Successful Completion message is displayed. The message definition data is imported.

Note: If the message definition import fails, use delbp.sql to delete all WebSphere Business Integration Adapter Framework message definitions, then repeat the import steps. The delbp.sql file is located in *%ProductDir*s%\connectors\PeopleSoft\PeopleSoft7 \dependencies\DataMover.

Embedding PeopleCode into panels

To enable triggering for Department and Employee business objects, you must embed fields in certain panels. Table 2 details which embedded fields are required for each business object.

Table 2. Business object triggering fields

Business object	Use for	Field name	Panel	Level
Department	Event notification	DEPT_TRIGGER	DEPT_TBLGBL_SBP	Level 0
Employee	Event notification	EMPL_PERS_TRIGGER	PERSONAL_DATA2	Level 0
Employee	Event notification	EMPL_JOB_TRIGGER	JOB_DATA1	Level 0
Employee	Request	EMPL_OTHERPHONE	PERSONAL_DATA2	Level 1 (phone scrollbar)

To embed fields in the appropriate panels:

1. In Panel Designer, open the panel for the appropriate business object listed in Table 2.
2. On the menu bar, select Add >Edit Box.
3. Click on the panel to drop an edit box on the panel.
4. Right click on the field and select Record from the popup menu.
5. In the Edit Box Record dialog box, enter XR_EVENT_WRK for Record Name, and enter the related Field Name from the table above.
6. Click OK and right click on the Edit Box again.
7. Select Use from the popup menu.
8. Select the Invisible check box, and click OK.

Event table structure

When an event occurs in the PeopleSoft application, the event is placed in an event queue table named XR_EVENT. The connector polls this table at a configurable interval and processes the events sequentially.

The event queue table and archive queue table have the following structure. However, only the archive table has the PROCESS_KEY field.

Table 3. Event queue and archive queue table structure

Name	Type	Constraint	Description
PROCESS_KEY	Datetime	Not NULL	Unique key for the event result set. The key is the date and time stamp when the event was processed. Used by the archive table only.
CWPRIORITY	char 3		Event priority.
EVENT_KEY	Datetime	Not NULL	Unique key for the event. The key is the date and time stamp when the event was processed.
OBJ_SETID	char 5		TableSetID for the PeopleSoft object.
OBJECT_ID	char 18		Unique ID for the PeopleSoft object.
OBJECT	char 18		Name of the PeopleSoft object.
DOVERB	char 15		Name of the action taken on the PeopleSoft object.
KEYLIST	char 51		List of non-primary object keys. Keys are listed in name=value format and delimited by a colon (:).
PRCS_FLG	char 3		Process flag to indicate the state of the event. The values can be: • N = Not processed • P = Successfully processed • NS = Event not subscribed to • E = Error

Configuring the connector

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

A connector obtains its configuration values at startup. During a run-time session, you might want to change the values of one or more connector properties. Changes to some connector configuration properties, such as AgentTraceLevel, take effect immediately. Changes to other connector properties require component restart or system restart after a change. To determine whether a property is dynamic (taking effect immediately) or static (requiring either connector component restart or system restart), refer to your integration broker's administration utility. For instance, if you are using ICS, see the Update Method column in the Connector Properties window of the System Manager.

Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, "Standard Configuration Properties for Connectors," on page 25 for documentation of these properties.

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

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

Connector-specific properties

Connector-specific configuration properties provide information needed by the connector at run time. 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 4 lists the connector-specific configuration properties for the connector. See the sections that follow for explanations of the properties.

Table 4. Connector-specific configuration properties

Name	Possible values	Default value
ApplicationPassword		PS
ApplicationUserName		PS
AppServerMachineNameOrIP		
ConnectErrors		Unable to connect to destination:ORACLE not available:TNS:listener failed:Could not connect to application server:Database access is not allowed:Failed to establish MsgAPI service context for operator
disableCrossReferencing	True/False	Enables or disables cross referencing. The default is false, cross referencing is enabled.
PollQuantitiy	1 to 500 (maximum)	25
PortNumber		7000
PeopleToolsVersion		7.57
Priority	0-n	0
UseDefaults		false

ApplicationPassword

The Password for the connector's user account.

ApplicationUserName

The Name of the connector's user account. ApplicationUserName is the OperatorID in PeopleSoft.

AppServerMachineNameOrIP

The Machine name or IP address of the machine name.

ConnectErrors

The List of strings that cause the connector to terminate when the connection to the PeopleSoft application or database is lost. These strings are written to the connector log when the connection is lost. Additional strings can be specified; separate strings with colon (:) delimiters.

disableCrossReferencing

Turns cross referencing on and off. The default value is False, cross-referencing is enabled.

PollQuantitiy

The number of events in the database table that the connector retrieves per polling interval.

PortNumber

The Port number for Message Agent requests.

PeopleToolsVersion
The version of PeopleTools, for example, 7.52.

Priority
The priority level from 0-n, with 0 being the highest level. At each poll interval, the connector picks up events starting with the highest priority events and continuing with lower priority events until the number of events specified in the PollQuantity configuration property is met, or until there are no events in the event table. The connector currently does not decrement priority.

UseDefaults
On a Create operation, if UseDefaults is set to `true` or not set, 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 succeeds; otherwise, it fails. If the parameter is set to `false`, the connector checks only for valid values; the Create operation fails if valid values are not provided.

Creating multiple versions of the connector

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

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

Create a new directory

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

```
ProductDir\connectors\connectorInstance
```

where `connectorInstance` uniquely identifies the connector instance.

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

```
ProductDir\repository\connectorInstance
```

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

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

   ```
   ProductDir\repository\initialConnectorInstance
   ```

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

Create a connector definition

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

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

Create a start-up script

To create a startup script:

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

   dirname
2. Put this startup script in the connector directory you created in "Create a new directory" on page 10.
3. Create a startup script shortcut (Windows only).
4. Copy the initial connector's shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

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

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

Starting the connector

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

ProductDir\connectors*connName*

where *connName* identifies the connector.

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

The name of the startup script depends on the operating-system platform, as Table 5 shows.

Table 5. Startup scripts for a connector

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

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

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

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

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

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

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

    `connector_manager -start connName brokerName [-cconfigFile ]`

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

  Note: For a WebSphere message broker or WebSphere Application Server on a Windows system, you *must* include the `-c` option followed by the name of the connector configuration file. For ICS, the `-c` is optional.
- From Adapter Monitor (available only when the broker is WebSphere Application Server or InterChange Server), which is launched when you start System Manager

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

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

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

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

Stopping the connector

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

- If you started the connector from the command line, with its connector startup script:

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

  `connector_manager_connName -stop`

  where *connName* is the name of the connector.

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

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.

- From System Monitor (WebSphere InterChange Server product only)

  You can load, activate, deactivate, pause, shutdown or delete a connector using this tool.

- On Windows systems, you can configure the connector to start as a Windows service. In this case, the connector stops when the Windows system shuts down.

Loss of connection to the application

If the connector determines that the connection with the application has been lost, the connector terminates and returns a status to the integration broker. The ConnectErrors property specifies which error strings cause the connector to terminate. For more information see "Connector-specific properties" on page 9.

Chapter 3. Developing business objects for the connector

This chapter describes how the WebSphere Business Integration Adapter for PeopleSoft 7.x processes business objects, and provides suggestions for developing and modifying business objects for the connector. It contains the followign sections:

Business object application-specific text

The application-specific text in a business object is used to identify where data is located in the application database. This section describes the application-specific text format that the connector expects.

Object application-specific text format

The connector uses application-specific text at the business-object level to determine which PeopleSoft activity the connector uses for that business object. To specify this in application-specific text, use the name-value pair `Activity=` *`name`*. The naming convention for activities is `CW`*`business object name`*. For example, for the Employee business object, the PeopleSoft activity is named `CWEmployee`. The application-specific text for this is:

```
[BusinessObjectDefinition]
Name = Psft_Employee
Version = 1.0.0
AppSpecificInfo = Activity=CWEmployee
...
```

The name parameter in the application-specific text must match the name of the activity in PeopleSoft. This name-value pair is needed only for a top-level business object. The connector also uses application-specific text at the business-object level to determine whether a business object will be processed by the connector. If the AppSpecificInfo field for the business object contains the string `NoOp`, the business object is not processed.

When using the `NoOp` string, the Activity name-value pair also needs to be specified. In this case, the Activity name-value pair precedes the `NoOp` string and is delimited by a colon. For example:

```
[BusinessObjectDefinition]
Name = Psft_Department
Version = 1.0.0
AppSpecificInfo = Activity=CWDepartment:NoOp
...
```

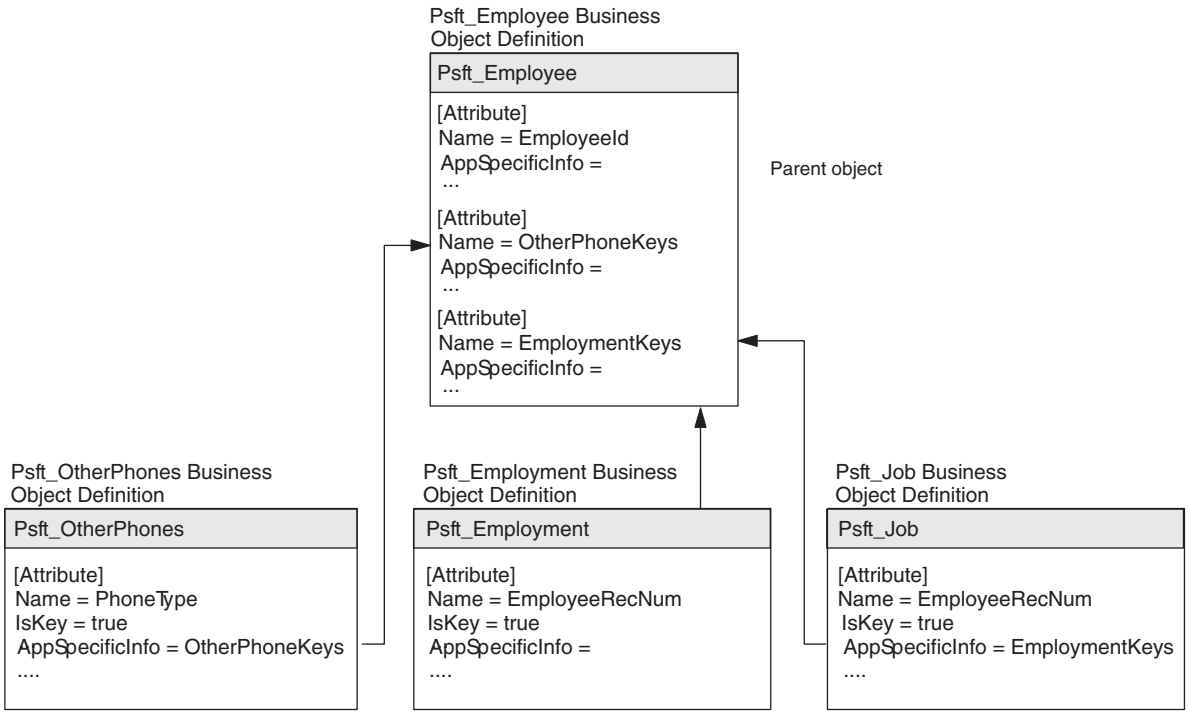
Attribute application-specific text format

The connector uses attribute application-specific text to retrieve keys for child objects in hierarchical business objects.

As part of the connector installation, message definitions that support business object/verb combinations are imported into the PeopleSoft system. These message definitions can contain one or more special fields that correspond to attributes in a top-level business object. Each special field is associated with PeopleCode that gathers the keys to child objects.

When the connector receives an event notification and queries for the complete set of data for a business object, the PeopleCode executes to dynamically assemble the child keys in the special fields. The connector then retrieves the value of each special field, which is a list of child keys, into the attribute corresponding to the special field. The connector can parse the list to identify the specific key for a particular attribute in a child object. The content of the special fields is not stored in the PeopleSoft database.

The following illustration shows example parent and child business objects and the format of the application-specific text. The `Psft_Employee` parent business object includes the attribute `OtherPhoneKeys`, which corresponds to the special field defined in the message definition for this business object. The child object includes the name of the special attribute in application-specific text for the key attribute. The connector identifies the key for the child object attribute from within the application-specific text.



When modifying business objects for the connector, key attributes in child objects must have application-specific text that points to a special attribute in the parent object.

Verb application-specific text format

The connector does not currently use the application-specific property for verbs. Leave this field blank when creating business object definitions.

Modifying business objects for the connector

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

When modifying or extending a business object for the connector, if PeopleCode is used to assemble keys, you must modify the PeopleCode in the XR_EVENT_WRK

table and the message definitions that were installed with the product. For example, to add a child business object to a hierarchical business object, you must perform the following general steps:

1. Modify the top-level business object to add two new attributes: the container attribute for the child business object and an attribute to hold the child keys.
2. Add the child business object. The key in the child business object must have application-specific text that points to the attribute at the top-level object. For example, in the `Psft_Employee` business object, the child object `Psft_OtherPhones` has the `AppSpecificInfo` field for the PhoneType attribute set to `OtherPhoneKeys` to point back to the top-level business object attribute OtherPhoneKeys.
3. Modify the PeopleCode in the XR_EVENT_WRK table to add a section that concatenates all the keys for the child objects. For the `Psft_Employee` business object, the PeopleCode for `Psft_OtherPhones` creates the string CELL:FAX:HOME, where the keys are separated by colons.
4. Add a message definition for the child business object and modify the message definition for the top-level business object. For example, the `Psft_Employee` business object has the following message definitions for the child object:
   - Retrieve OtherPhones
   - Create OtherPhones
   - Update OtherPhones
5. Add `OtherPhonesKeys` to the Retrieve Employee message definition for the top-level business object.

Rules for modifying business objects

Follow these rules when modifying business objects for the connector:

Objects spanning multiple panel groups

If a top-level business object or child business object spans two different panel groups, split the business object or child object into two business objects.

Effective dated and effective sequenced rows

1. PeopleSoft application objects are designed to model the way the panel processes data. A panel with effective dated information identifies the database record that the effective dated information is stored in. This table/record must have the EFFDT as one of its keys. For an example of this, compare the Psft_EffDtEmployee object to the record PS_PERS_DATA_EFFDT. This is the main record referenced on the effective dated subpanel within the employee panel groups.
2. EffectiveSequence is part of the Job record key, which enables you to track events by sequence on the same effective date. Effective Sequence is a manual input field used to track the multiple actions that occur on the same effective date. It does not have any automated database sequencing. For example, if you entered a “Posn Chg” action for an employee on 4/25/00, and no other action had been performed on that date, leave the Effective Sequence field at `0`. If you later entered a “Pay Rt Chg” action for the same employee on 4/25/00, set the Effective Sequence field to 1.
3. For all effective dated and effective sequenced rows, you must repeat effective date field or effective sequence twice in the message definition. This is necessary only for Create operations in the message definition--for example, `CreateJob`.
4. For effective dated rows, list all fields that show up on the panel as attributes in the business object. When inserting new rows into the database, the

connector follows the same behavior as if the user is online inserting a row by pressing the F7 key. In both cases, all the attributes in the scroll are copied into the new scroll. If you do not specify all the attributes in the business object, you run the risk of unintentionally copying data from a previous row into the new row.

5. For effective dated rows, you need to specify the attribute application-specific text to be EFFDT in the Business Object.
6. For effective dated rows, you need an extra message definition. The name of the message definition has to be the business object name + EFFDT. The attributes are the keys (input) and EFFDT (output). For example:

```
Message Definition = RetrieveJobEFFDT
Attributes =   EmployeeId (input)
          EmployeeRecNum (input)
          JobEffectiveOnDate (output)
```

Message definitions

1. Because the Message Agent processes data in the same order as the online panels, the order of fields in the message definition is important only for fields requiring validation from previously entered data. Otherwise, the order of the fields listed on the Message Definition is not important.
2. In the message definitions, the message `Output All Occurrences` works only for Level 1 fields. See Create Message Definitions in the Peoplebooks Development Tools for more information. If you need to retrieve a Level 2 or Level 3 row, you must create a new panel to move the Level 2 or Level 3 fields to Level 1.

Business object attributes

Attributes must exist in both the business object and the message definition. Attribute names must be spelled consistently. The order of attributes in the business object is not important.

Child business object keys

Child business objects do not need to repeat the keys that already exist in the parent business objects. When the connector processes a hierarchical business object, the connector first obtains the keys that are stored in the parent business objects, then processes the child business objects. However, the keys must be repeated in the Message Agent definition.

Attribute names

Do not use EFFDT or EFFSEQ as the attribute name.

AppSpecificInfo

You can have only two identifiers for AppSpecificInfo. They must be delimited by a colon (:). For example:

```
AppSpecificInfo = EFFDT:Additional PayKeys
```

Event notifications

To process event notifications, the connector takes the value of `OBJECT_ID` in the event table and stores it in a field called [*business object name*]Id. This field name might differ from the actual name of the key in PeopleSoft. For example, in PeopleSoft 7.5, the key for the Item table is `InvItemId`, but the connector requires that the name of the key in the Item business object be `ItemId`.

Attribute names in business objects and message definitions

The connector requires that there be an exact match in attribute names between the business object and the message definitions for each verb for that object. For

example, there must be an exact match between the attribute names in Psft_Item and the attribute names in the following message definitions:

- Create Item
- Retrieve Item
- Update Item

As you update message definitions, make sure that the spelling and capitalization of the attribute names matches the attributes in the business object. These message definitions can be found in the IBM WebSphere business integration business process and the Connector Integration activity.

Transaction processing for hierarchical business objects

If a failure occurs when the connector is processing a hierarchical business object with a Create or Update verb, one or more child business objects may not be processed.

The connector does not define a transaction as an operation on a complete hierarchical business object. Instead, the connector defines a separate transaction for each operation on individual parent or child business objects in a hierarchical business object. When the connector receives a business object with a Create verb, it first processes the parent business object and then processes each individual child business object. Under normal conditions, it begins a transaction for each business object in the hierarchy, creates the entity in the application, and commits the transaction.

If the connector succeeds in creating one or more child entities, and a failure occurs during the creation of another child entity, the connector rolls back the Create operation for this particular business object by backing out the entity. The connector continues processing child business objects until all the remaining entities have been created. Because there was a failure, the connector returns BON_FAIL to the integration broker for the overall operation on the business object. In addition, the Create or Update operation in the application may be incomplete. This scenario is similar for an Update transaction.

When the connector returns BON_FAIL, the complete hierarchical business object is placed in the unprocessed events queue. A system administrator needs to check the destination application and resolve the problem that caused the failure. Then the system administrator needs to resubmit the business object to the connector. Because the connector checks for the existence of application entities before creating them, the connector does not create duplicate data on a business object resubmission for those child business objects that have already been created.

Sample business object definition file

There are three sample business object definition files included with the product:

- BO_Psft_DEPT
- BO_PsftEmployee
- SavePostChange

BO_Psft_DEPT business object

The following example is the BO_Psft DEPT business object.

```
[BusinessObjectDefinition] Name = DeptTbl Version = 1.0.0 AppSpecificInfo =
CiName=DEPT     [Attribute]     Name = Company     Type = String
MaxLength = 255     IsKey = true     IsForeignKey = false     IsRequired =
false     AppSpecificInfo = get=getCompany:set=setCompany
IsRequiredServerBound = false     [End]     [Attribute]     Name =
BudgetLvl     Type = String     MaxLength = 1     IsKey = false
IsForeignKey = false     IsRequired = false     AppSpecificInfo =
get=getBudgetLvl:set=setBudgetLvl     IsRequiredServerBound = false
[End]     [Attribute]     Name = Descr     Type = String     MaxLength = 1
    IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getDescr:set=setDescr     IsRequiredServerBound =
false     [End]     [Attribute]     Name = DescrShort     Type = String
MaxLength = 1     IsKey = false     IsForeignKey = false     IsRequired =
false     AppSpecificInfo = get=getDescrshort:set=setDescrshort
IsRequiredServerBound = false     [End]     [Attribute]     Name =
ObjectEventId     Type = String     MaxLength = 255     IsKey = false
IsForeignKey = false     IsRequired = false     IsRequiredServerBound =
false     [End]     [Verb]     Name = Create     [End]     [Verb]     Name
= Delete     [End]     [Verb]     Name = Retrieve     [End]     [Verb]
Name = Update     [End] [End] [BusinessObjectDefinition] Name = Psft_dept
Version = 1.0.0 AppSpecificInfo = CiName=DEPT     [Attribute]     Name =
Deptid     Type = String     MaxLength = 255     IsKey = true
IsForeignKey = false     IsRequired = true     AppSpecificInfo =
get=getDeptid:set=setDeptid:GetKey=true     IsRequiredServerBound = false
    [End]     [Attribute]     Name = Setid     Type = String     MaxLength
= 1     IsKey = true     IsForeignKey = false     IsRequired = true
AppSpecificInfo = get=getSetid:set=setSetid:GetKey=true
IsRequiredServerBound = false     [End]     [Attribute]     Name = DptTbl
    Type = DeptTbl     ContainedObjectVersion = 1.0.0     Relationship =
Containment     Cardinality = n     MaxLength = 1     IsKey = false
IsForeignKey = false     IsRequired = true     AppSpecificInfo =
get=getDeptTbl:KEEPRELATIONSHIP=true     IsRequiredServerBound = false
[End]     [Attribute]     Name = ObjectEventId     Type = String
MaxLength = 255     IsKey = false     IsForeignKey = false     IsRequired =
false     IsRequiredServerBound = false     [End]     [Verb]     Name =
Create     [End]     [Verb]     Name = Delete     [End]     [Verb]     Name
= Retrieve     [End]     [Verb]     Name = Update     [End] [End]
```

BO_PsftEmployee business object

The following example is the BO_PsftEmployee business object.

```
[BusinessObjectDefinition] Name = PSFTEmployee Version = 1.0.0
AppSpecificInfo = cIName=Emp     [Attribute]     Name = EMPID     Type =
String     Cardinality = 1     MaxLength = 255     IsKey = true
IsForeignKey = false     IsRequired = true     AppSpecificInfo =
get=getEmplid:set=setEmplid:keepRelationship=false:uid=true:
findKey=true:getKey=true:createKey=true     IsRequiredServerBound = false
    [End]     [Attribute]     Name = EMPL_RCD     Type = String
Cardinality = 1     MaxLength = 1     IsKey = true     IsForeignKey = false
    IsRequired = true     AppSpecificInfo =
get=getEmplRcd:set=setEmplRcd:keepRelationship=false:uid=true:
findKey=true:getKey=true:createKey=true     IsRequiredServerBound = false
    [End]     [Attribute]     Name = NAME     Type = String     Cardinality
= 1     MaxLength = 1     IsKey = true     IsForeignKey = false
IsRequired = true     AppSpecificInfo =
```

```
get=getName:set=setName:keepRelationship=false:uid=false:
findKey=true:getKey=false:createKey=false     IsRequiredServerBound = false
    [End]     [Attribute]     Name = LAST_NAME_SRCH     Type = String
Cardinality = 1 MaxLength = 1     IsKey = true     IsForeignKey = false
IsRequired = true     AppSpecificInfo =
get=getLastNameSrch:set=setLastNameSrch:keepRelationship=false:uid=false:
    findKey=true:getKey=false:createKey=false     IsRequiredServerBound =
false     [End]     [Attribute]     Name = NAME_AC     Type = String
Cardinality = 1     MaxLength = 1     IsKey = true     IsForeignKey = false
    IsRequired = true     AppSpecificInfo =
get=getNameAc:set=setNameAc:keepRelationship=false:uid=false:
findKey=true:getKey=false:createKey=false     IsRequiredServerBound = false
    [End]     [Attribute]     Name = PER_STATUS     Type = String
Cardinality = 1     MaxLength = 1     IsKey = true     IsForeignKey = false
    IsRequired = true     AppSpecificInfo =
get=getPerStatus:set=setPerStatus:keepRelationship=false:uid=false:
findKey=true:getKey=false:createKey=false     IsRequiredServerBound = false
    [End]     [Attribute]     Name = EMPLID_0     Type = String
Cardinality = 1     MaxLength = 1     IsKey = false     IsForeignKey =
false     IsRequired = false     AppSpecificInfo =
get=getEmplid0:set=setEmplid0:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false     IsRequiredServerBound =
false     [End]     [Attribute]     Name = ORIG_HIRE_DT     Type = String
    Cardinality = 1     MaxLength = 1     IsKey = false     IsForeignKey =
false     IsRequired = false     AppSpecificInfo =
get=getOrigHireDt:set=setOrigHireDt:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false     IsRequiredServerBound =
false     [End]     [Attribute]     Name = SEX     Type = String
Cardinality = 1     MaxLength = 1     IsKey = false     IsForeignKey =
false     IsRequired = true     AppSpecificInfo =
get=getSex:set=setSex:keepRelationship=false:uid=false:findKey=false:
getKey=false:createKey=false     IsRequiredServerBound = false     [End]
    [Attribute]     Name = BIRTHDATE     Type = String     Cardinality = 1
    MaxLength = 1     IsKey = false     IsForeignKey = false     IsRequired
= false     AppSpecificInfo =
get=getBirthdate:set=setBirthdate:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name =
FT_STUDENT     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = true
AppSpecificInfo = get=getFtStudent:set=setFtStudent:keepRelationship=false:
    uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name =
BENEFIT_RCD_NBR     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getBenefitRcdNbr:set=setBenefitRcdNbr:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
HOME_HOST_CLASS     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = true
AppSpecificInfo = get=getHomeHostClass:set=setHomeHostClass:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
HIRE_DT     Type = String     Cardinality = 1     MaxLength = 1     IsKey =
false     IsForeignKey = false     IsRequired = false     AppSpecificInfo =
get=getHireDt:set=setHireDt:keepRelationship=false:
```

```
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name =
CMPNY_SENIORITY_DT     Type = String     Cardinality = 1     MaxLength = 1
    IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getCmpnySeniorityDt:
set=setCmpnySeniorityDt:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false     IsRequiredServerBound =
false     [End]     [Attribute]     Name = SERVICE_DT     Type = String
Cardinality = 1     MaxLength = 1     IsKey = false     IsForeignKey =
false     IsRequired = false     AppSpecificInfo =
get=getServiceDt:set=setServiceDt:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
PROF_EXPERIENCE_DT     Type = String     Cardinality = 1     MaxLength = 1
    IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getProfExperienceDt:
set=setProfExperienceDt:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false     IsRequiredServerBound =
false     [End]     [Attribute]     Name = LAST_VERIFICATN_DT     Type =
String     Cardinality = 1     MaxLength = 1     IsKey = false
IsForeignKey = false     IsRequired = false     AppSpecificInfo =
get=getLastVerificatnDt:set=setLastVerificatnDt:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
EXPECTED_RETURN_DT     Type = String     Cardinality = 1     MaxLength = 1
    IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getExpectedReturnDt:set=
setExpectedReturnDt:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name =
LAST_DATE_WORKED     Type = String     Cardinality = 1     MaxLength = 1
    IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getLastDateWorked:
set=setLastDateWorked:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false     IsRequiredServerBound =
false     [End]     [Attribute]     Name = LAST_INCREASE_DT     Type =
String     Cardinality = 1     MaxLength = 1     IsKey = false
IsForeignKey = false     IsRequired = false     AppSpecificInfo =
get=getLastIncreaseDt:     set=setLastIncreaseDt:keepRelationship=false:
    uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name =
OWN_5PERCENT_CO     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = true
AppSpecificInfo = get=getOwn5percentCo:
set=setOwn5percentCo:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name =
BUSINESS_TITLE     Type = String     Cardinality = 1     MaxLength = 1
IsKey = true     IsForeignKey = false     IsRequired = true
AppSpecificInfo = get=getBusinessTitle:
set=setBusinessTitle:keepRelationship=false:
uid=false:findKey=false:getKey=true:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name =
REPORTS_TO     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getReportsTo:set=setReportsTo:
```

```
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
SUPERVISOR_ID     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getSupervisorId:set=setSupervisorId:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
PROBATION_DT     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getProbationDt:set=setProbationDt:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
SECURITY_CLEARANCE     Type = String     Cardinality = 1     MaxLength = 1
    IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getSecurityClearance:
set=setSecurityClearance:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name = PHONE
    Type = String     Cardinality = 1     MaxLength = 1     IsKey = false
    IsForeignKey = false     IsRequired = false     AppSpecificInfo =
get=getPhone:set=setPhone:     keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false     IsRequiredServerBound =
false     [End]     [Attribute]     Name = TIME_RPT_LOCK     Type = String
    Cardinality = 1     MaxLength = 1     IsKey = false     IsForeignKey =
false     IsRequired = false     AppSpecificInfo =
get=getTimeRptLock:set=setTimeRptLock:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
JOB_REPORTING     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = true
AppSpecificInfo = get=getJobReporting:set=setJobReporting:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
DED_TAKEN     Type = String     Cardinality = 1     MaxLength = 1     IsKey
= false     IsForeignKey = false     IsRequired = true     AppSpecificInfo
= get=getDedTaken:set=setDedTaken:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
DED_SUBSET_ID     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getDedSubsetId:set=setDedSubsetId:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
CAN_ABORIGINAL     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getCanAboriginal:set=setCanAboriginal:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
CAN_VISBL_MINORITY     Type = String     Cardinality = 1     MaxLength = 1
    IsKey = false     IsForeignKey = false     IsRequired = true
AppSpecificInfo = get=getCanVisblMinority:set=setCanVisblMinority:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
CURRENT_SEQ     Type = String     Cardinality = 1     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getCurrentSeq:set=setCurrentSeq:
```

```
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false     [End]     [Attribute]     Name =
PERS_DATA_EFFDT     Type = PERS_DATA_EFFDT     ContainedObjectVersion =
1.0.0     Relationship = Containment     Cardinality = n     MaxLength = 1
    IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getPersDataEffdt:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name =
EMAIL_ADDRESSES     Type = EMAIL_ADDRESSES     ContainedObjectVersion =
1.0.0     Relationship = Containment     Cardinality = n     MaxLength = 1
    IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getEmailAddresses:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name =
PERSONAL_PHONE     Type = PERSONAL_PHONE     ContainedObjectVersion = 1.0.0
    Relationship = Containment     Cardinality = n     MaxLength = 1
IsKey = false     IsForeignKey = false     IsRequired = false
AppSpecificInfo = get=getPersonalPhone:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false     [End]     [Attribute]     Name = PERS_NID
    Type = PERS_NID     ContainedObjectVersion = 1.0.0     Relationship =
Containment     Cardinality = n     MaxLength = 1     IsKey = false
IsForeignKey = false     IsRequired = false     IsRequiredServerBound =
false     [End]     [Attribute]     Name = JOB     Type = JOB
ContainedObjectVersion = 1.0.0     Relationship = Containment
Cardinality = n     MaxLength = 1     IsKey = false     IsForeignKey =
false     IsRequired = false     IsRequiredServerBound = false     [End]
    [Attribute]     Name = ObjectEventId     Type = String     MaxLength =
255     IsKey = false     IsForeignKey = false     IsRequired = false
IsRequiredServerBound = false     [End]     [Verb]     Name = Create
[End]     [Verb]     Name = Delete     [End]     [Verb]     Name = Retrieve
    [End]     [Verb]     Name = Update     [End] [End]
```

Appendix A. Standard Configuration Properties for Connectors

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

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

If your adapter supports DB2 Information Integrator, use the WMQI options and the DB2 II standard properties (see the Notes column in Table 6 on page 27.)

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

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

New properties

These standard properties have been added in this release:

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

Standard connector properties overview

Connectors have two types of configuration properties:

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

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

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

Starting Connector Configurator

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

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

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

Configuration property values overview

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

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

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

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

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

There are four update methods for standard connector properties:

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

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

To determine how a specific property is updated, refer to the **Update Method** column in the Connector Configurator window, or see the Update Method column in Table 6 on page 27.

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

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

Note: In the Notes column in Table 6, the phrase "RepositoryDirectory is set to <REMOTE>" indicates that the broker is InterChange Server. When the broker is WMQI or WAS, the repository directory is set to <*ProductDir*>\repository

Table 6. Summary of standard configuration properties

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

Table 6. Summary of standard configuration properties (continued)

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

Table 6. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
CommonEventInfrastructure	true or false	false	Component restart	
CommonEventInfrastructureURL	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.
ConcurrentEventTriggeredFlows	1 through 32,767	1	Component restart	This property is valid only if the value of RepositoryDirectory is set to <REMOTE> and the value of BrokerType is ICS.
ContainerManagedEvents	Blank or JMS	Blank	Component restart	This property is valid only when the value of Delivery Transport is JMS.
ControllerEventSequencing	true or false	true	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ControllerStoreAndForwardMode	true or false	true	Dynamic	This property is valid only if the value of Repository Directory is set to <REMOTE> and the value of BrokerType is ICS.
ControllerTraceLevel	0 through 5	0	Dynamic	This property is valid only if the value of RepositoryDirectory is set to <REMOTE> and the value of BrokerType is ICS.
DeliveryQueue	Any valid JMS queue name	*<CONNECTORNAME>* /DELIVERYQUEUE	Component restart	This property is valid only when the value of Delivery Transport is JMS.
DeliveryTransport	MQ, IDL, or JMS	IDL when the value of RepositoryDirectory is <REMOTE>, otherwise JMS	Component restart	If the value of RepositoryDirectory is not <REMOTE>, the only valid value for this property is JMS.
DuplicateEventElimination	true or false	false	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
EnableOidForFlowMonitoring	true or false	false	Component restart	This property is valid only if the value of BrokerType is ICS.
FaultQueue	Any valid queue name.	*<CONNECTORNAME>* /FAULTQUEUE	Component restart	This property is valid only if the value of DeliveryTransport is JMS.
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory, CxCommon.Messaging.jms.SonicMQFactory, or any Java class name	CxCommon.Messaging.jms.IBMMQSeriesFactory	Component restart	This property is valid only if the value of DeliveryTransport is JMS.

Table 6. Summary of standard configuration properties (continued)

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

Table 6. Summary of standard configuration properties (continued)

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

Table 6. Summary of standard configuration properties (continued)

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

Table 6. Summary of standard configuration properties (continued)

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

Standard properties

This section describes the standard connector configuration properties.

AdapterHelpName

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

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

AdminInQueue

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

The default value is <*CONNECTORNAME*>/ADMININQUEUE

AdminOutQueue

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

The default value is <*CONNECTORNAME*>/ADMINOUTQUEUE

AgentConnections

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

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

The default value of this property is 1.

AgentTraceLevel

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

The default value is `0`.

ApplicationName

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

The default is the name of the connector.

BiDi.Application

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

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

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

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

BiDi.Broker

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

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

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

BiDi.Metadata

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

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

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

BiDi.Transformation

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

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

The default value is `false`.

BrokerType

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

CharacterEncoding

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

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

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

CommonEventInfrastructure

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

The default value is `false`.

CommonEventInfrastructureContextURL

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

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

The default value is a blank field.

ConcurrentEventTriggeredFlows

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

Setting this property to a value greater than 1 allows a connector for a source application to map multiple event business objects at the same time and deliver

them to multiple collaboration instances simultaneously. This speeds delivery of business objects to the integration broker, particularly if the business objects use complex maps. Increasing the arrival rate of business objects to collaborations can improve overall performance in the system.

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

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

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

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

The default value is 1.

ContainerManagedEvents

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

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

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

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

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

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

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

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

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

There is no default value.

ControllerEventSequencing

The ControllerEventSequencing property enables event sequencing in the connector controller.

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

The default value is `true`.

ControllerStoreAndForwardMode

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

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

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

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

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

The default value is `true`.

ControllerTraceLevel

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

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

The default value is `0`.

DeliveryQueue

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

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

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

DeliveryTransport

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

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

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

The default value is JMS.

WebSphere MQ and IDL

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

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

JMS

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

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

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

The default value is `false`.

EnableOidForFlowMonitoring

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

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

The default value is `false`.

FaultQueue

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

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

jms.FactoryClassName

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

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

jms.ListenerConcurrency

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

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

The default value is 1.

jms.MessageBrokerName

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

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

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

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

jms.NumConcurrentRequests

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

The default value is `10`.

jms.Password

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

There is no default value.

jms.TransportOptimized

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

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

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

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

The default value is `false`.

jms.UserName

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

There is no default value.

JvmMaxHeapSize

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

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

The default value is 128m.

JvmMaxNativeStackSize

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

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

The default value is 128k.

JvmMinHeapSize

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

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

The default value is 1m.

ListenerConcurrency

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

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

The default value is 1.

Locale

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

A locale name has the following format:

ll_TT.codeset

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

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

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

The default value is `en_US`.

LogAtInterchangeEnd

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

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

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

The default value is `false`.

MaxEventCapacity

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

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

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

The default value is 2147483647.

MessageFileName

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

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

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

The default value is `InterchangeSystem.txt`.

MonitorQueue

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

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

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

OADAutoRestartAgent

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

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

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

The default value is `false`.

OADMaxNumRetry

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

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

The default value is `1000`.

OADRetryTimeInterval

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

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

The default value is `10`.

PollEndTime

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

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

If the adapter runtime detects:

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

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

PollFrequency

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

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

The following values are valid for this property:

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

The default is `10000`.

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

PollQuantity

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

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

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

- When it is polled once, the connector detects the body of the message, which it reads as an attachment. Since no data handler was specified for this mime type, it will then ignore the message.

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

PollStartTime

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

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

If the adapter runtime detects:

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

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

RepositoryDirectory

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

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

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

RequestQueue

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

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

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

ResponseQueue

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

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

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

RestartRetryCount

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

The default value is 3.

RestartRetryInterval

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

Possible values for the property range from 1 through 2147483647.

The default value is 1.

ResultsSetEnabled

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

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

The default value is false.

ResultsSetSize

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

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

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

RHF2MessageDomain

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

This is an example header:

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

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

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

SourceQueue

The SourceQueue property designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see "ContainerManagedEvents" on page 36.

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

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

SynchronousRequestQueue

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

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

The default value is <*CONNECTORNAME*>/SYNCHRONOUSREQUESTQUEUE

SynchronousRequestTimeout

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

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

The default value is 0.

SynchronousResponseQueue

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

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

The default is <*CONNECTORNAME*>/SYNCHRONOUSRESPONSEQUEUE

TivoliMonitorTransactionPerformance

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

The default value is false.

WireFormat

The WireFormat property specifies the message format on the transport:

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

WsifSynchronousRequestTimeout

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

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

The default value is 0.

XMLNameSpaceFormat

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

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

The default value is short.

Appendix B. Connector Configurator

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

You use Connector Configurator to:

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

The topics covered in this appendix are:

Overview of Connector Configurator

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

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

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

You use Connector Configurator to:

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

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see "Running Configurator in stand-alone mode" on page 50).

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

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

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

For **connector-specific properties**, however, you need first to define the properties and then set their values. You do this by creating a connector-specific property template for your particular adapter. There may already be a template set up in your system, in which case, you simply use that. If not, follow the steps in "Creating a new template" on page 51 to set up a new one.

Running connectors on UNIX

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

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

Starting Connector Configurator

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

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

Running Configurator in stand-alone mode

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

To do so:

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

You may choose to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see "Completing a configuration file" on page 56.)

Running Configurator from System Manager

You can run Connector Configurator from System Manager.

To run Connector Configurator:

1. Open the System Manager.
2. In the System Manager window, expand the **Integration Component Libraries** icon and highlight **Connectors**.
3. From the System Manager menu bar, click **Tools>Connector Configurator**. The Connector Configurator window opens and displays a **New Connector** dialog box.
4. When you click the pull-down menu next to **System Connectivity Integration Broker**, you can select ICS, WebSphere Message Brokers or WAS, depending on your broker.

To edit an existing configuration file:

- In the System Manager window, select any of the configuration files listed in the Connector folder and right-click on it. Connector Configurator opens and displays the configuration file with the integration broker type and file name at the top.
- From Connector Configurator, select **File>Open**. Select the name of the connector configuration file from a project or from the directory in which it is stored.
- Click the Standard Properties tab to see which properties are included in this configuration file.

Creating a connector-specific property template

To create a configuration file for your connector, you need a connector-specific property template as well as the system-supplied standard properties.

You can create a brand-new template for the connector-specific properties of your connector, or you can use an existing connector definition as the template.

- To create a new template, see "Creating a new template" on page 51.
- To use an existing file, simply modify an existing template and save it under the new name. You can find existing templates in your `\WebSphereAdapters\bin\Data\App` directory.

Creating a new template

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

To create a template in Connector Configurator:

1. Click **File>New>Connector-Specific Property Template**.
2. The **Connector-Specific Property Template** dialog box appears.
   - Enter a name for the new template in the **Name** field below **Input a New Template Name.** You will see this name again when you open the dialog box for creating a new configuration file from a template.

- To see the connector-specific property definitions in any template, select that template's name in the **Template Name** display. A list of the property definitions contained in that template appears in the **Template Preview** display.

3. You can use an existing template whose property definitions are similar to those required by your connector as a starting point for your template. If you do not see any template that displays the connector-specific properties used by your connector, you will need to create one.
   - If you are planning to modify an existing template, select the name of the template from the list in the **Template Name** table below **Select the Existing Template to Modify: Find Template.**
   - This table displays the names of all currently available templates. You can also search for a template.

Specifying general characteristics

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

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

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

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

Specifying values

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

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

To create a new property value:

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

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

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

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

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

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

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

Setting dependencies

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

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

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

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

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

Setting pathnames

Some general rules for setting pathnames are:

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

- Queue names may not have leading or embedded spaces.

Creating a new configuration file

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

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

To start Connector Configurator:

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

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

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

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

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

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

Creating a configuration file from a connector-specific template

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

1. Set the operating system for extended validation of the configuration file using the **Target System:** droplist on the menu bar (see "Creating a new configuration file" above).
2. Click **File>New>Connector Configuration**.
3. The **New Connector** dialog box appears, with the following fields:
   - **Name**

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

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

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

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

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

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

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

Using an existing file

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

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

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

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

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

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

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

- All files (*.*)

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

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

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

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

Completing a configuration file

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

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

1. Under the **Standard Properties** tab, select the value field for the `BrokerType` property. In the drop-down menu, select the value `ICS`, `WMQI`, or `WAS`.
2. The Standard Properties tab will display the connector properties associated with the selected broker. The table shows **Property name**, **Value**, **Type**, **Subtype** (if the Type is a string), **Description**, and **Update Method**.
3. You can save the file now or complete the remaining configuration fields, as described in "Specifying supported business object definitions" on page 59..
4. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

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

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

Setting the configuration file properties

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

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

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

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

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

- Associated Maps
- Resources
- Messaging (where applicable)
- Security

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

Standard properties differ from connector-specific properties as follows:

- Standard properties of a connector are shared by both the application-specific component of a connector and its broker component. All connectors have the same set of standard properties. These properties are described in 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, click **File>Exit** (or close the window), and click **No** when prompted to save changes.
   - To enter values for other categories in Connector Configurator, select the tab for the category. The values you enter for **Standard Properties** (or any other category) are retained when you move to the next category. When you close the window, you are prompted to either save or discard the values that you entered in all the categories as a whole.
   - To save the revised values, click **File>Exit** (or close the window) and click **Yes** when prompted to save changes. Alternatively, click **Save>To File** from either the File menu or the toolbar.

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

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

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

Setting connector-specific configuration properties

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

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

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

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

   Note: If the hot button does not appear, no Extended Help was found for that property.
5. Choose to save or discard changes, as described for "Setting standard connector properties" on page 58.

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

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

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

Encryption for connector properties

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

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

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

Update method

Refer to the descriptions of update methods found in the Standard Properties appendix, under "Configuration property values overview" on page 26.

Specifying supported business object definitions

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

Note: Some connectors require that certain business objects be specified as supported in order to perform event notification or additional configuration

(using meta-objects) with their applications. For more information, see the *Connector Development Guide for C++* or the *Connector Development Guide for Java*.

If ICS is your broker

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

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

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

To delete a business object from the supported list:

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

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

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

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

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

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

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

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

If a WebSphere Message Broker is your broker

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

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

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

If WAS is your broker

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

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

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

Associated maps (ICS)

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

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

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

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

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

- **Business Object Name**

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

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

- **Explicit Binding**

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

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

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

  To explicitly bind a map:

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

Resources (ICS)

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

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

Messaging (ICS)

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

Validating messaging queues

Before you can validate a messaging queue, you must:

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

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

Security (ICS)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Setting trace/log file values

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

To change the logging and tracing values:

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

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

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

Data handlers

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

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

Saving your configuration file

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

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

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

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

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

Changing a configuration file

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

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

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

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

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

COPYRIGHT LICENSE

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

IBM
the IBM logo
AIX
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.

IBM WebSphere Business Integration Adapter FrameworkV2.6.